A coupled view of the physical abilities of human-robot dyad for the online quantitative evaluation of assistance needs

Antun Skuric

université de BORDEAUX

université de BORDEAUX

THÈSE PRÉSENTÉE

POUR OBTENIR LE GRADE DE

DOCTEUR
DE L'UNIVERSITÉ DE BORDEAUX

ÉCOLE DOCTORALE DES SCIENCES PHYSIQUES ET DE L'INGÉNIEUR

AUTOMATIQUE, PRODUCTIQUE, SIGNAL ET IMAGE, INGÉNIERIE COGNITIQUE

Par **Antun Skuric**

A coupled view of the physical abilities of human-robot dyad for the online quantitative evaluation of assistance needs

Une vue couplée des capacités physiques de la dyade humain-robot pour l'évaluation quantitative en ligne des besoins d'assistance

Sous la direction de : **David Daney** et **Vincent Padois**

Soutenance le 6 novembre 2023

Membres du jury :

Mme. Christine Chevallereau	Directrice de Recherche	CNRS	Rapportrice
M. Darwin Lau	Professeur Agrégé	The University of Hong Kong	Rapporteur
M. Nicolas Mansard	Directeur de Recherche	LAAS-CNRS	Examinateur
M. Stéphane Caron	Charge de Recherche	INRIA Paris - Willow team	Examinateur
M. Philip Long	Maître de Conférences	Atlantic Technological University	Examinateur
M. David Daney	Directeur de Recherche	INRIA Université de Bordeaux	Directeur
M. Vincent Padois	Directeur de Recherche	INRIA Université de Bordeaux	Directeur

Une vue couplée des capacités physiques de la dyade humain-robot pour l'évaluation quantitative en ligne des besoins d'assistance

Résumé : Cette thèse repose sur une vision de l'avenir où la robotique et l'industrie sont centrées autour des humains, mettant l'accent sur la collaboration entre les humains et les robots plutôt que sur une stricte automatisation. Dans ce futur collaboratif, les robots servent de collaborateurs actifs, coexistant étroitement avec les humains et participant à des interactions physiques pour exécuter des tâches. De tels systèmes symbiotiques exploitent les capacités uniques des humains et des robots, améliorant l'efficacité et donnant la priorité à la sécurité et au bien-être des humains grâce à une assistance robotique personnalisée.

La réalisation de cette vision nécessite la capacité de quantifier les diverses capacités des humains et des robots dans une façon unifiée, ainsi que leurs capacités conjointes lors de la collaboration. De plus, cela nécessite la capacité de mesurer l'assistance requise par les opérateurs afin de garantir leur sécurité et leur bien-être. Par conséquent, cette thèse préconise l'utilisation de mesures de capacité physique, en particulier de leurs représentations par des polytopes, pour répondre à ces questions. Les polytopes sont particulièrement adaptées aux scénarios de collaboration, car les différents outils efficaces de l'algèbre permettent de réaliser des opérations sur les polytopes (telles que la somme de Minkowski, l'intersection et l'union) et de caractériser les capacités physiques de plusieurs robots et humains sous forme de polytopes.

Cette thèse propose une vue structurée des polytopes de capacité physique communs pour les humains et les robots, ainsi qu'un aperçu des méthodes d'evaluation applicables. Deux nouveaux algorithmes d'évaluation de polytopes sont proposés, particulièrement adaptés à l'évaluation des polytopes de force (wrench) des manipulateurs robotiques et des humains basés sur des modèles musculosquelettiques. Ces algorithmes réduisent considérablement la complexité des méthodes de l'état de l'art et permettent des applications en temps réel.

Ensuite, la thèse explore l'utilisation des polytopes comme source d'informations en temps réel sur les capacités physiques changeantes des humains et des robots et leur potentiel pour améliorer différents aspects de la collaboration homme-robot.

Dans le contexte de la collaboration physique homme-robot, la thèse présente l'utilisation des informations en temps réel sur les capacités changeantes des humains et des robots pour créer des stratégies de contrôle de robot adaptables. Les stratégies développées sont validées expérimentalement dans le cadre du transport collaboratif d'objets.

De plus, la thèse explore la visualisation des polytopes de capacité physique comme source d'information en temps réel pour les opérateurs. Elle introduit une nouvelle formulation de polytope, l'espace atteignable dans un horizon temporel, qui offre des informations intuitives sur l'état actuel d'un robot et ses capacités physiques.

De plus, une nouvelle approche de planification de trajectoire basée sur les polytopes dans l'espace cartésien est introduite. Cette approche exploite l'algèbre des polytopes pour évaluer efficacement la capacité de mouvement du robot dans la direction de la trajectoire et exploite la pleine capacité de mouvement du robot en mettant à jour la trajectoire planifiée en temps réel.

Enfin, cette thèse présente le package 'pycapacity', un logiciel efficace et intitif pour calculer les mesures de capacité physique des humains et des robots, ouvrant la voie à leur utilisation dans la communauté élargie.

Mots-clés : interaction humain robot, robotique collaborative, analyse des performances, géométrie computationnelle, commande

A coupled view of the physical abilities of human-robot dyad for the online quantitative evaluation of assistance needs

Abstract: This thesis is based on vision of the future where robotics and industry are centred around humans, emphasising collaboration between humans and robots rather than mere automation. In this collaborative future, robots serve as active assistants, coexisting closely with humans and engaging in physical interactions to execute tasks. Such symbiotic systems leverage the unique abilities of both humans and robots, enhancing efficiency and prioritising human safety and well-being through personalised robotic assistance.

Realising this vision requires being able to quantify the diverse abilities of humans and robots in a unified view, as well as their joint abilities when collaborating. Additionally, it requires being able to measure the assistance required by operators in order to ensure their safety and well-being. Therefore, this thesis advocates for the use of physical ability metrics, particularly their polytope representations, to address these questions. Polytope representations are particularly well suited for collaborative scenarios, as the different efficient tools from polytope algebra permit making operations over polytopes (such as Minkowski sum, intersection and unions) and characterising the physical abilities of multiple robots and humans in the polytope form as well.

This thesis proposes a structured view on common physical ability polytopes for humans and robots along with an overview of their applicable evaluation methods. Two new polytope evaluation algorithms are proposed, particularly well suited for evaluating wrench polytopes of robotic manipulators and human's based on the musculoskeletal models. The algorithms significantly reduce the complexity of the state of the art methods and enable real-time applications.

The thesis then explores the use of polytopes as a source of real-time information about human's and robot's changing physical abilities and their potential to enhance different aspects of human-robot collaboration.

In the context of human-robot physical collaboration, the thesis showcases the use of the real-time information about human's and robot's changing abilities for creating adaptable robot control strategies. The developed strategies are experimentally validated in the collaborative object carrying setting.

Furthermore, the thesis explores the visualisation of physical ability polytopes as real-time feedback for operators. It introduces a novel polytope formulation, the reachable space within a time horizon, which offers intuitive insights into a robot's current state and physical abilities.

Additionally, a new polytope based Cartesian Space trajectory planning approach is introduced. This approach leverages the polytope algebra to efficiently evaluate the robot's movement capacity in the trajectory direction and exploits the robot's full movement capacity by updating the planned trajectory in real-time.

Finally, this thesis presents the 'pycapacity' package, an efficient and user-friendly framework for calculating physical ability metrics of humans and robots, opening doors to their use in the wider community.

Keywords: human robot interaction, collaborative robotics, performance analysis, computational geometry, control

Contents

Acronyms

AAN Assist-As-Needed 10, 89, 90, 94, 99, 155

AR Augmented Reality 12, 101, 102, 122–124, 167, 170

CHM Convex-Hull Method 54, 55, 66, 67, 74

CS Cartesian Space 4, 5, 12, 16, 20, 21, 32, 57, 64, 67, 71, 72, 74, 75, 97, 99, 102–105, 109–113, 116, 117, 124–136, 138, 139, 142–147, 150–152, 158, 159, 163, 167, 168

DDM Double-Description Method 45

DMP Dynamic Motion Primitives 127

ESP Equality-Set Projection 51, 53, 109

FME Fourier-Motzkin Elimination 51, 53, 109

GTA Graph Traversal Algorithms 45

HPSM Hyper-Plane Shifting Method 53, 71, 72, 160

INA Incremental Algorithms 45

JS Joint Space 5, 20, 21, 23, 24, 27, 32, 59, 102, 104, 113, 118, 119, 126–131, 135, 138, 139, 142, 144–146, 150–152, 155, 158–160

LP Linear Program 10, 47, 54, 55, 67–69, 73, 74, 132–134, 144

MCDR Multi-link Cable-Driven Robot 23–28

MPC Model Predictive Control 117, 127, 150, 151, 168

MR Mixed Reality 101

PIM Pivoting Method 45, 71

QP Quadratic Program 10, 81, 89, 96, 143, 151

REBA Rapid Entire Body Assessment 7

ROS Robot Operating System 83, 91, 113, 122, 144, 148, 153, 154

RSM Ray Shooting Method 25, 54, 55, 57, 67, 70–72

RULA Rapid Upper Limb Assessment 7

SVD Singular Value Decomposition 61, 68, 156

TAP Trapezoidal Acceleration Profile 127, 128, 130, 133, 135–144, 146, 147, 150–152, 168

VR Virtual Reality 101, 102, 124

Chapter 1

Introduction

In recent years, the field of robotics is undergoing a profound transformation, with a growing emphasis on human-centered and autonomous systems. The traditional perception of robotics as a means to replace human labor with automated machines is gradually giving way to a more human-oriented approach. Robotics is no longer solely about exploiting robots to remove humans from various tasks; instead, it is becoming a means to enhance human capabilities through collaboration with autonomous machines. This paradigm shift is driven by the expectation that the true potential of robots lies in their ability to interact with humans and operate effectively within human environments.

One of the key areas where this shift is particularly evident is in industrial settings. Industrial processes, especially those requiring specialised expertise, have historically relied on human operators who have developed their skills and knowledge over years of experience. While complete automation of such processes might seem appealing, it often proves to be impractical or even infeasible due to the complexity and variability of tasks. This is where the concept of human-robot collaboration comes into play.

This thesis is developed in the context of the LiChIE project, a project funded by BPI France and coordinated by Airbus Defence and Space. The primary objective of the LiChIE project is to improve the efficiency of the assembly processes of mini satellites. These processes demand a high degree of precision and expertise, which human operators have traditionally provided. Due to the high complexity and variability of the satellite assembly and the small scale of the production, full automating is not a viable solution. Therefore, the project aims to leverage robotics to support and enhance the capabilities of human operators, rather than replace them entirely.

The industry of the future, as described by recent movements like Industry 5.0 [1] and Society 5.0 [2], promises to go even a step further. It puts human workers in the central position and aims to create the workflows that ensure their well-being as well as the long term sustainability of the industrial practices in general [3]. The industry of the future relies on the flexibility and adaptability of the human workers, by embracing their cognitive and physical skills as well as their talents and different levels of expertise. The role of the automation is no longer purely optimisation of the industrial processes, but providing the support and assistance to the human operator with the aim to reach both machines' and humans' full potential, establishing human-automation symbiosis [4] on the industry floor.

Such symbiosis relies on scenarios where humans and robots work in a close proximity and interact physically to execute different tasks, such as human-robot collaborative workstations [5]. On the one hand, the symbiosis enables improving the overall efficiency by leveraging the abilities of both humans (flexibility, adaptability, cognitive capacity, expertise, etc.) and robots

(repeatability, precision, tirelessness, etc.). On the other hand, it enables improving the operator's well-being, remove the unnecessary strain and ensure their safety when executing tasks.

When it comes to putting this ideas to practice, such future collaborative scenarios require having a set of tools for characterising different abilities of robots and humans, as well as different notions of human well-being and safety. These tools are necessary for creating new task allocation strategies, permitting to quantify if different tasks are more suitable for human's or robot's abilities or potentially for their collaboration. Furthermore, they could facilitate the development of more human-centred robotic assistance strategies, by quantifying the extent of assistance the operator requires from the robot, both for task execution and for ensuring well-being and safety.

In order to achieve such advanced collaboration behaviours in practice, such tools should be able to fulfil a set of requirements

Requirement 1 (RQ1)**.** *Enable quantifying the abilities required to execute a task*

Requirement 2 (RQ2)**.** *Enable quantifying human's and robot's individual abilities*

Requirement 3 (RQ3)**.** *Enable quantifying their common abilities when collaborating*

Requirement 4 (RQ4)**.** *Enable quantifying human's need of assistance or their well-being*

Even though human-robot collaboration is still a relatively recent field, there are numerous metrics, measures and indicators proposed in the literature that quantify different aspects of the quality of their collaboration and their individual abilities [6]. However, it's important to note that metrics assessing the abilities of humans and robots often differ significantly in their nature and focus. On the one hand, metrics for robots typically concentrate on their physical capabilities (maximum speed, force, etc.), safety considerations (stopping time, force measurement capacity, etc.), and usability factors (programming ease, flexibility, etc.). On the other hand, metrics for humans tend to concentrate on cognitive abilities (situation awareness, mental workload, etc.), ergonomic indicators (physical and cognitive), as well as emotional states (satisfaction, acceptance, trust, etc.). Such separate set of metrics are well suited for evaluating their individual abilities, addressing RQ2 and RQ4. However, due to their different nature, characterising the necessary abilities to execute a certain task (RQ1), as well as characterising human's and robot's contribution to their common abilities when collaborating (RQ3), becomes challenging.

The focus of this thesis is put on characterising human's and robot's physical abilities, arguing that they provide a unified set of tools capable of addressing all four requirements RQ1-RQ4.

Characterising physical abilities of humans and robots consists in describing how different properties of robotic systems, such as the actuator limits and kinematic structure, as well as different biomechanical and biological limitations of human bodies, influence their respective capacity to execute movements, generate forces, achieve different precision levels or generate other physical quantities necessary to execute a certain task (RQ2).

Traditionally, different characterisation of the physical abilities required to execute a certain set of tasks are important tools for the analysis and design of robotic manipulators [7], as well as for defining the tasks they are capable of executing [8], by comparing the task requirements to the robot's abilities (RQ1). In case of humans, they are commonly used to ensure human safety by studying the ergonomics of different tasks and workspaces [9] (RQ4). Additionally, some preliminary works from Chiacchio *et al.* [10] and Lee [11] have shown that physical ability metrics can be used to characterise common physical abilities of a collaborative system as well (RQ3). Although these works characterise the collaboration of multiple robots rather than human-robot collaboration, similar approach can be used for the human-robot case as well.

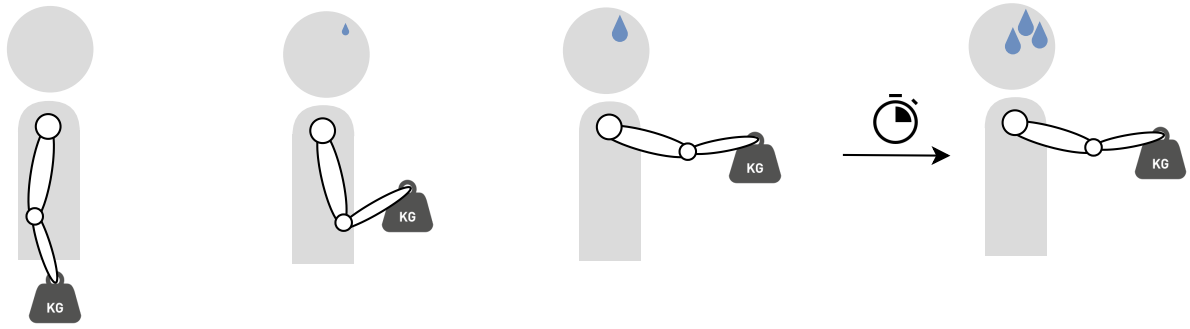

FIGURE 1.1: Illustration of the intuition about the changing nature of humans' physical abilities. When carrying a heavy object, putting it close to the body results in lower physical effort than if the object is further from the body. Moreover, even if the arm posture does not change, the effects such as muscular fatigue can cause that, over time, the level of necessary physical effort increases substantially.

Moreover, human's and robot's physical abilities are not constant, they vary with time and with their state, and can evolve significantly during the task execution. This is something that we all know intuitively. For example, if we carry a heavy object in our hand, as illustrated on Figure 1.1. We all know that having our arm close our body makes carrying much easier than if our arm is stretched in front of us. Additionally, even if our arm does not move, the effects of muscle fatigue can decrease our ability to carry the object and ultimately make us unable to do it after certain time. Therefore, even though the weight of the object does not change, our capacity (physical ability) of carrying it can change significantly with the changing state (posture) of the arm, as well as different physiological factors such as muscular fatigue. Similar parallel can be made for robotic manipulators as well.

Having an accurate online information about changing physical abilities of humans, enables assessing if the they are apt to execute a certain task and quantify if they need robot's assistance, due to their lacking physical abilities. On the other hand, accurate and online information about robot's changing physical abilities, enables creating robot control strategies that adapt to the their changes and exploit the robot's full physical potential. Therefore, to better assist humans and fully exploit robot's physical potential, tools capable of capturing their changing physical abilities in real-time are needed, adding an additional requirement

Requirement 5 (RQ5)**.** *Enable capturing human's and robot's changing abilities online*

To go beyond qualitative characterisation of human-robot collaboration, the main focus of this thesis is therefore put on developing tools capable of capturing the changing physical abilities of humans and robots in real-time. In the long term this is anticipated as a necessary steps towards the enhancement of the collaboration and, overall, should contribute to the safety and well-being of humans at work.

As metrics characterising different physical abilities, for humans and robots, are numerous in the literature, Section 1.1 brings an overview of commonly used ones and discusses their potential to address the requirements RQ1-RQ5.

1.1 Characterising physical abilities

Many different metrics for characterising physical abilities of humans and robots are developed in the literature. They have a varying degree of complexity and physical interpretation, as well as different scope and accuracy. Therefore, this section brings an overview of different

physical ability metrics for robots and humans and discusses their potential use for human-robot collaboration.

1.1.1 Metrics in robotics

In robotics, physical abilities are characterised as different performance indicators, establishing the relationship between the robot's actuator limits (joint positions, velocities, torques, etc.), its kinematics and dynamics equations, and the achievable sets of different task related physical quantities, such as achievable positions, velocities, external wrenches and similar. The metrics quantifying different physical abilities are usually divided in two groups with respect to their scope: *Global* and *Local* metrics [12].

Global metrics present robot state independent metrics evaluated by taking in consideration all the possible robot states. One example of such metric is the robot's reachable workspace [13–15], characterising the set of Cartesian Space (CS) positions the robot can reach given its geometry and the limitations of its actuator's positions. More generally, workspace analysis based tools find the set of robot's reachable CS positions given the limitations of its actuators, its kinematics and dynamics and different task related variables. Examples of such metrics are constant orientation workspace [16] representing robot's reachable positions with a given fixed orientation, singularity free workspace [17] representing the reachable positions without any singular configurations, or wrench closure workspace [18, 19] which corresponds to robot's reachable positions guaranteeing the ability to apply certain set of wrenches. One traditional application for such performance metrics is robot dimensional design, as they enable verifying and guaranteeing that the robotic system, being designed, is compliant with all the requirements of the tasks. However, these reachable workspace analysis tools have relatively complex geometry which can be challenging to exploit when it comes practical applications.

A different set of global metrics, often specified in manufacturer's data-sheets, are different scalar metrics representing the robot's physical abilities based on finding the *worst-case* (minimal) or *best-case* (maximal) values of different task related values within the workspace. These metrics guarantee certain physical ability values for a given robot, such as robot's payload, its *worst-case* carrying capacity, robot's *worst-case* positioning accuracy or positioning repeatability [12]. Therefore, they are often used to evaluate if a certain robotic system is suitable for a given task, and potentially compare between the robots from different manufacturers. However, such simplified metrics evaluating the robot's physical abilities, based on *worst-case* scenarios, are in many cases largely underestimating robot's true capacity.

Furthermore, global metrics are computationally expensive as their evaluation requires sweeping through all the robot states, and they are typically calculated only once for a particular robot design or a given task. Moreover, as they are evaluated for entire set of robot's states, they are by design not capable of characterising the changing nature of the robot's physical abilities, that are state dependent (RQ5).

Local metrics, on the other hand, are robot state dependant performance metrics which can be evaluated relatively efficiently for any given state. They characterise robot's task related physical abilities for a single robot state, as opposed to the global metrics which are calculated for all robot states. Additionally, as they enable quantitative comparison of the performance of different robot's states, they can be used as a base for different global metrics, in order to characterise robot's abilities in the whole workspace [20]. The local metrics are often represented as scalar values indexes for a given robot's state [21], where some of the most well known ones are: manipulability index [22] and condition number [23] which characterise the movement and dexterity capacity of the robot as well as its accuracy [24], dynamic manipulability [25] that characterises the task related acceleration capacity or the robot stiffness [26] characterising the

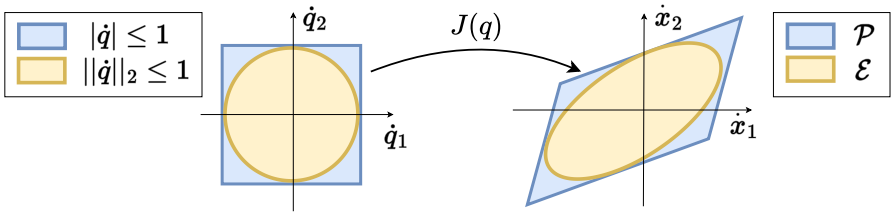

FIGURE 1.2: An example manipulability polytope and ellipsoid geometry for a planar $m = 2$ robot with $n = 2$. The difference between the Joint Space (JS) limits for ellipsoid described with $||\dot{\boldsymbol{q}}||_2 \leq 1$ (orange) and the range limits $-\mathbf{1} \leq \dot{\boldsymbol{q}} \leq \mathbf{1}$ (blue) is shown on the left. The difference in obtained achievable task space velocity $\dot{\boldsymbol{x}}$ polytope $\mathcal{P}$ (blue) and ellipsoid $\mathcal{E}$ (orange) is shown on the right plot. The plots show that both in joint and task space the ellipsoid metric is an underestimation of the true robot's capacity.

robot's tasks related load and deflection relationship. Having a scalar metric is beneficial when it comes their use in different optimisation scenarios, for example in robot design refinement [15] or in robot trajectory generation [27].

However having scalar representation of robot's capabilities can be constraining in certain applications, especially if the task has more than one dimension, for example movement in the 3D space. Different representations of robot's local physical abilities have been developed over the years that are capable of characterising the robot's task related capacity in the multi-dimensional task space. One particularly common representation is in ellipsoid form. Ellipsoids represent the ease of generating different task related variables (velocity, acceleration, inertia, stiffness etc.) in different directions of task space. One of the first ellipsoid shaped robot's physical ability representations was developed by Yoshikawa [22] and is called the manipulability ellipsoid. Manipulability ellipsoid describes the capacity of a robotic system to generate task space velocities and can be described mathematically as

$$\mathcal{E}(\boldsymbol{q}) = \{\dot{\boldsymbol{x}} \mid \dot{\boldsymbol{x}} = J(\boldsymbol{q})\dot{\boldsymbol{q}}, \quad ||\dot{\boldsymbol{q}}||_2 \leq 1\} \tag{1.1}$$

where $J(\boldsymbol{q})$ is robot's state $\boldsymbol{q}$ dependant Jacobian matrix relating its joint velocity $\dot{\boldsymbol{q}}$ and the task space velocity $\dot{\boldsymbol{x}}$[1]. This metric evaluates the capacity of generating different task space velocities $\dot{\boldsymbol{x}}$ by considering that the robot can generate all the joint velocities $\dot{\boldsymbol{q}}$ with equal ease, represented by the condition $||\dot{\boldsymbol{q}}||_2 \leq 1$. Similar ellipsoid based representations can be used to characterise the force capacity of robotic systems [10], acceleration capacity [25], inertia [28], stiffness [29], etc.

The hypothesis of equal capacity (ease of generating motion, forces, accelerations etc.) for all the robot's joints, in general case, does not reflect the true nature of robot actuation capacity. Robots have different actuators for different joints that can have substantially different characteristics. Therefore, in order to evaluate more accurate robot's task space capacity (motion, force, acceleration etc.) their true limits have to be taken in consideration. The ellipsoid metrics can be extended to considering non-uniform robot actuation limits, by introducing scaling

$$||W\dot{\boldsymbol{q}}||_2 \leq 1, \quad W = \text{diag}\left([\dot{q}_{1,max},\ \dot{q}_{2,max},\ \ldots]\right)^{-1} \tag{1.2}$$

where scaling matrix W normalises the robot joint velocity capacity. However, they still make the assumption that the limits can be represented in the euclidean norm $||.||_2$ form, implying

[1]Task space velocity $\dot{\boldsymbol{x}}$ in general case can include both translational and rotational components, corresponding to the CS twist. In that case $\dot{\boldsymbol{x}}$ is an abuse of notation, as there is no representation of a CS pose $\boldsymbol{x} \in SE(3)$ which time derivative is a twist. Yet, for the sake of conciseness this notation is used throughout this manuscript.

that the robot's actuator limits are mutually interdependent, which in a general case is not true [30]. Furthermore, as the real robot limits are usually expressed in a form of independent ranges $\dot{\boldsymbol{q}}_{min} \leq \dot{\boldsymbol{q}} \leq \dot{\boldsymbol{q}}_{max}$, the euclidean norm (1.2) based representation is an underestimation of the true robot's capacities. This effect is shown on the example of manipulability ellipsoid metric on Figure 1.2.

Once the true robotic actuator limits are considered, the representation of physical abilities becomes convex polytope shaped. As opposed to the manipulability ellipsoid $\mathcal{E}$, manipulability polytope can be defined as

$$\mathcal{P}(\boldsymbol{q}) = \{\dot{\boldsymbol{x}} \mid \dot{\boldsymbol{x}} = J(\boldsymbol{q})\dot{\boldsymbol{q}}, \quad \dot{\boldsymbol{q}}_{min} \leq \dot{\boldsymbol{q}} \leq \dot{\boldsymbol{q}}_{max}\} \tag{1.3}$$

The manipulability polytope $\mathcal{P}$ represents an exact solution of the robot's task space velocity capacity, given robot's state $\boldsymbol{q}$ and its joint velocity limits $\dot{\boldsymbol{q}}$. The difference between the manipulability ellipsoid $\mathcal{E}$ and polytope $\mathcal{P}$ is shown for a simple 2D example on Figure 1.2.

Furthermore, as discussed by Finotello *et al.* [31], polytopes allow for calculating many other local performance metrics, such as Maximum Available Value (MAV) corresponding to the highest achievable magnitude of the task space variable (velocity, acceleration, force, static error, etc.), Maximal Isotropic Value (MIV) corresponding to the maximal value achievable in all the directions in space or Directional Index (DI) [32] corresponding to the maximum achievable task space variable in certain direction in space. Additionally, polytope volume and major (preferred) axis can be used as performance metrics as well [10, 33].

Therefore, polytope based physical ability metrics are the most complete local representation of task space physical abilities of robotic systems and they are used to characterise many task related physical values such as velocities [30, 34], forces [35], accelerations [36], stiffness [29], positioning accuracy [37], etc. The main downside to the polytope based metrics, with respect to the ellipsoid based metrics, is their computational complexity, making their use somewhat limited to non time critical applications.

1.1.2 Metrics for humans

Evaluating physical abilities of humans is a much more challenging than for robotic systems, as human bodies are much more complicated systems that depend both on biological and biomechanical, as well as cognitive and psychological factors.

Measuring different physical abilities of human subjects is traditionally done experimentally. The ability of human subjects to achieve different physical quantities such as generate forces [38, 39], reach positions [40], execute movements [41] or generate stiffnesses [42, 43], is tested in laboratory environment for a set of predefined human postures or movements. These experiments enable finding precise physical abilities of one or multiple subjects, for the tested motions and postures. Such experimental analysis are common tools for evaluation of physical abilities in sports [41], where the motions and postures of interest are usually well defined, and in rehabilitation [44] where the aim is to measure the progress of the human subject's recovery. However due to the large variability in human bodies and movements, it is challenging to generalise these results to the human subjects who were not part of the experiments, as well as to different postures and movements, of the same human subjects, that were not tested in the experiments.

In more varied and unstructured settings, where the human postures and movements are harder to characterise, such as in factory setting, the experimental metrics are impractical. For these scenarios, instead of characterising the physical ability of human subject directly, different metrics are developed aiming to evaluate the ergonomics of human postures and movements

when executing tasks. These metrics are able to account for variations in human body anthropomorphic structure as well as for different movement and forces generation requirements of the tasks. Such metrics are often defined in a form of standards, such as Rapid Entire Body Assessment (REBA) [45], Rapid Upper Limb Assessment (RULA) [46] or DULA and DEBA [47] (their differentiable equivalents), and manuals, such as Great Britain's *Manual Handling Operation Regulations* [48] or NASA's *Man-systems integration standards* [49]. These metrics are designed to give a certain score or a recommendation for the human postures, movements or force generation requirements of the task, in order to evaluate if the task is well suited for human's physical abilities. Common applications of these metrics, in factory settings, have for a goal to evaluate and improve the ergonomics of human's tasks [50] and workplace layouts [51, 52]. However, these metrics, in order to be task and human subject agnostic, make coarse approximations of human physical abilities which are constraining when it comes to the online human-robot collaboration scenarios [53].

In order to characterise human's physical abilities more finely, without the need for time consuming and impractical experimental evaluation, different metrics based on human musculoskeletal models are proposed. Human musculoskeletal models are relatively complete models of human bodies capable of describing different body's biological and biomechanical parameters as well as its rigid body dynamics. Musculoskeletal models are actuated by muscle-tendon units, capable of applying a range of contracting forces, accelerations and velocities. Similar to robotic systems, several global and local physical ability metrics, based on musculoskeletal models, can be calculated by evaluating how different limits of the muscles as actuators effect human's task related physical abilities, such as to generate forces or motions.

Several global physical ability metrics based on musculoskeletal models are introduced in literature, such as human upper extremity reachable workspace [54, 55] or comfortable reachable workspace [56], the comfortablility map of the reachable workspace of human's upper limbs evaluated using RULA and REBA ergonomics scores.

In addition to the global metrics many robotics inspired local, state dependent, metrics have been developed as well. Different authors have used their ellipsoid based representations to asses the velocity (manipulability) [57], force [58, 59], acceleration [60] and stiffness [43] capacity of humans based on their musculoskeletal models. Several of these ellipsoids have been extend to the more complete polytope representation as well, such as force polytope [58, 59, 61] and acceleration capacity polytope [60, 62]. As in the case of robotic systems, polytopes represent the exact solution to the physical ability evaluation for any given state of the musculoskeletal model.

However, as the musculoskeleral models are only an approximation of the true physics of human bodies, the accuracy of calculated physical ability metrics relies entirely on the correspondence between the model and the human subject. Multiple experimental studies were conducted comparing the experimental data against the polytopes and ellipsoids obtained using musculoskeletal models, namely for the human upper limb force generation capacity [59, 63, 64]. Their results confirm that the polytopes correspond better to the experimentally obtained capacity of the human subjects. However, the results further underline the importance of having an appropriate musculoskeletal model of the human subject. Even though many strategies have been developed for fitting the musculoskeletal models to human subjects in the biomechanics literature mostly based on different forms of scaling [65, 66], it is still an open field of research.

With the assumption of an appropriate and well adapted human musculoskeletal model of the human subject, polytopes and ellipsoids present an accurate estimation of the human's task related physical abilities. Even though polytopes are more accurate estimation, ellipsoid based representations, due to their computational efficiency, are still a preferred choice when it comes

to evaluating human's physical ability, especially in real-time applications. The difference in computational complexity between polytope and ellipsoid evaluation for musculoskeletal models is even more exaggerated than for the robotic systems. As the musculoskeletal models often have many muscles (often more than 50) final polytope geometry becomes complex which ultimately has a negative impact of their computation time.

1.1.3 Physical abilities in human-robot collaboration

Human's and robot's individual physical ability metrics have been used in different human-robot collaboration scenarios. Robot's physical abilities are often calculated in order to make sure that the task requirements comply with the robot's capacity. On the other hand, human's physical ability and ergonomics metrics are then used to design suitable robot behaviors in order to make the human's task and posture more ergonomic. Such collaboration strategies consider the robot to be an action vector with limited resources whose job is to improve the task performance and the human ergonomics at the same time. Examples of such human-centered collaborative scenarios are described by Kim *et al.* [67], where the robot adapts the position of the manipulated object in space in order to improve human's ergonomics, or in the context of exoskeleton control by Carmichael and Liu [68, 69] and Petrič *et al.* [70], where the robot adapts to different assistance needs of the human operator.

However, when it comes to the collaboration scenarios where a human and a robot interact physically in order to execute a certain task, characterising their joint physical abilities is still an open research question. Human and robot physical abilities are often expressed in different ways, with different units and with fundamentally different metrics, making the procedure of combining them challenging. Therefore, in order to characterise their joint physical capacity, the first step is to express their individual physical abilities need in the same unified form.

When it comes to multi-robot physical collaboration, Chiacchio *et al.* [10] have shown that ellipsoid based metrics can be used to calculate their combined joint velocity (manipulability) capacity. The resulting joint capacity has an ellipsoid form as well. However, in their formulation, the collaboration of multiple robots is essentially seen as one larger robot, combining all the joints of all the robots, resulting in the robot with $n = n_1 + n_2 + \dots$ joints.

$$\boldsymbol{q} = \begin{bmatrix} \boldsymbol{q}_1 \\ \boldsymbol{q}_2 \\ \dots \end{bmatrix}, \quad \dot{\boldsymbol{q}} = \begin{bmatrix} \dot{\boldsymbol{q}}_1 \\ \dot{\boldsymbol{q}}_2 \\ \dots \end{bmatrix}, \quad J(\boldsymbol{q}) = \mathrm{diag}([J_1(\boldsymbol{q}_1),\ J_2(\boldsymbol{q}_2),\ \dots]) \tag{1.4}$$

Where n_i is the number of joints, while $\boldsymbol{q}_i$ is the vector of joint positions and $J_i(\boldsymbol{q}_i)$ is the Jacobian matrix for each one of the robot's involved.

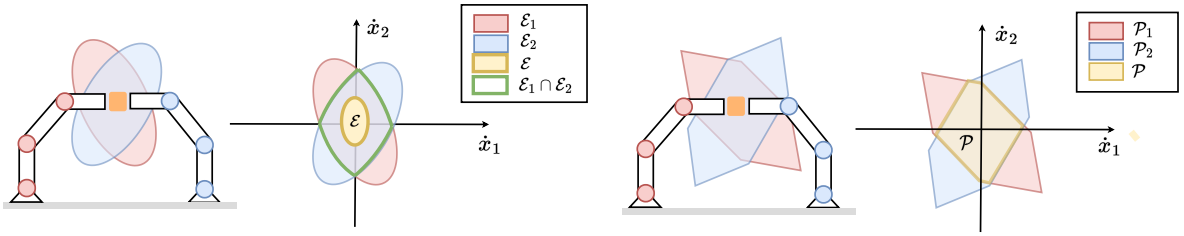

FIGURE 1.3: Comparison of using ellipsoids (left) and polytopes (right) for collaborative physical ability calculation. Using polytopes, the common velocity capacity of the two robots is calculated as the intersection of their polytopes $\mathcal{P} = \mathcal{P}_1 \cap \mathcal{P}_2$. In ellipsoid case the intersection of the ellipsoids (green) is no longer an ellipsoid and it is hard to characterise, while the collaborative ellipsoid $\mathcal{E}$ (orange) introduced by Chiacchio *et al.* [10] is a large underestimation of the true joint capacity.

With such large number of joints, the euclidean norm $||.||_2$ limits present a large under-approximation of the real robot's limits, resulting in large under-approximation of the true capacity of the collaboration. As shown on the example on Figure 1.3, a more precise approach would be to intersect the ellipsoids calculated independently for all the robots

$$\mathcal{E} = \mathcal{E}_1 \cap \mathcal{E}_2 \cap \dots$$

However, the intersection operation over ellipsoids is hard to characterise and to evaluate, as the intersection of two ellipsoid is no longer an ellipsoid, and even if the intersection of the ellipsoids is obtained, it is just an approximation of true common capacity.

Expressing the robot's physical ability in the polytope form however is not just the exact solution, but enables using the polytope algebra to do different operations, such as sum, intersection and Convex-Hull. These operations are well defined and can be calculated efficiently. The work from Jihong Lee [11] shows that polytopes metrics can be used to describe the common velocity capacity of multi-arm collaborative robotic system. The work describes an efficient way of calculating the joint velocity capacity by intersecting the individual polytopes of each one of the robots involved, resulting in a convex polytope shaped joint velocity capacity

$$\mathcal{P} = \mathcal{P}_1 \cap \mathcal{P}_2 \cap \dots$$

By exploiting different polytope algebra operations it is possible to express different physical collaboration scenarios as well. For example, if now the robots would be re-arranged from the parallel to the serial configuration, where the robot would be stacked one on top of the other, their joint velocity capacity would correspond to the Minkowski sum of their individual capacity

$$\mathcal{P} = \mathcal{P}_1 \oplus \mathcal{P}_2 \oplus \dots$$

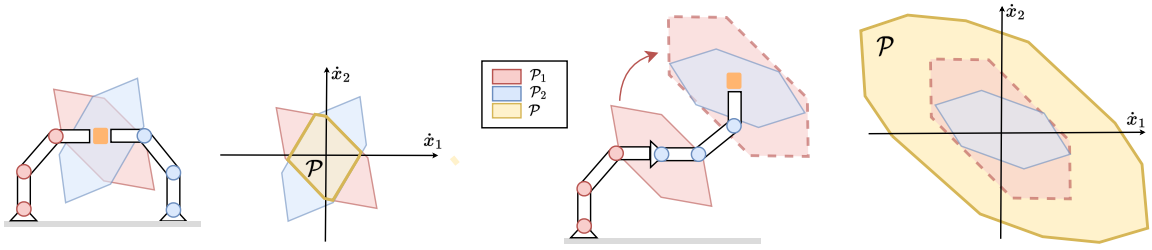

FIGURE 1.4: Two examples of using polytope based metrics for joint achievable velocity capacity calculation of multiple robots collaborating. Figure on the left shows the parallel collaboration where the common capacity is calculated with the intersection. Figure on the right shows the serial collaboration where the common capacity is a Minkowski sum of their individual capacity. As the capacity is calculated at the end-effector of the second robot, the velocity polytope $\mathcal{P}_1$ of the first robot has to be transformed the same position (in red dashed border).

Figure 1.4 shows a simplified plannar example of the joint velocity capacity calculation for parallel and serial robot arrangement.

In summary, the polytope representation enables the physical abilities of robots, humans, and their collaboration to be expressed through a unified format. Furthermore, as many different physical abilities of humans and robots can be expressed in polytope form, it has the potential to provide an important set of tools for evaluating different collaboration tasks. For example, by evaluating different physical abilities relevant to the task, it enables characterising if the task is more suitable for humans, robots or for different modes of their physical collaboration.

1.1.4 Why polytopes?

As described in the previous sections, polytope representations of physical abilities accurately characterise the physical abilities of robotic manipulators, at the same time being the most accurate approximation of human physical abilities based on musculoskeletal models (RQ2). Additionally, since most of the well known polytope formulations for robotic systems can be formulated as polytopes for musculoskeletal models as well, they enable evaluating and comparing human's and robot's physical abilities in a unified manner. Furthermore, the polytope algebra enables operations over multiple polytopes and, in that way, has a potential to enable characterising common physical abilities of humans and robots involved in different physical collaboration scenarios (RQ3).

Moreover, polytopes are local metrics, accurately representing different task related physical abilities for any robot's or human's state. Therefore, they enable capturing their state dependent changes in their physical abilities (RQ5).

Polytopes enable comparing the operator's changing physical abilities with the physical abilities required by the task. Such real-time information has a potential to improve the operator's safety and well-being, by enabling to create assistive robot control strategies that adapt to the operator's changing capacity and make sure that they are never surpassed (RQ4). Similar human-centred and assistive robot control strategies are used in the context of assistive exoskeletons, as a part of so called Assist-As-Needed (AAN) [68] control paradigm. Using the AAN control strategy, the robot provides assistance to the operator adapted to the operator's current physical ability, providing the assistance only where the operator needs it. In their work Carmichael and Liu [69] have proposed to quantify the need of assistance of the operator as the lacking physical ability to execute a certain task.

When it comes to using polytopes in practical applications, there are two standard ways of representing them: as a set of vertices (vertex representation) or as a set of inequality constraints corresponding to their faces (half-plane representation) [71]. Transforming a polytope into these standard representations enables using different efficient tools from computational geometry and polytope algebra in order to do operations over polytopes, for example find minimal distance between polytopes [72], polytope volume [73] or calculate the Convex-Hulls [74], Minkowski sums [75], intersections and unions of multiple polytopes [76]. Additionally, it opens doors for different simplification strategies of polytopes, by finding the inner and outer approximations of polytopes using spheres [77] or cubes [78].

When characterising physical abilities, transforming the polytopes to the vertex representation enables using efficient triangulation algorithms from computational geometry. Most of the standard visualisation tools can visualise such triangulated meshes in very efficient manner, even in real-time. In the context of human-robot collaboration, an example application of polytope visualisation is proposed by Zolotas *et al.* [79], where the robot's velocity polytopes are used to inform the operator about the robot's movement capacity during the teleportation task. A similar application was recently proposed by Weistroffer *et al.* [80], where the polytopes of robot's force capacity were displayed to the operator designing the collaborative workspace with the aim to improve its safety and ergonomics.

Polytopes can be expressed as a set of linear inequalities $A\boldsymbol{x} \leq \boldsymbol{b}$, that can be integrated into different optimisation problems in a form of linear inequality constraints. Most of the standard Quadratic Program (QP) [81] and Linear Program (LP) [82] solvers support the inequality constraints in an efficient manner. In robotics, typical examples of applications often based on different optimisation strategies are robot control and trajectory planning. In the context of human-robot collaboration, being able to express both robot's and human's physical abilities in the polytope form, opens doors for creating more flexible robot control (or trajectory planning)

strategies, which take in consideration both robot's and human's physical abilities.

Therefore, polytope representation provides an accurate representation of both humans' and robots' physical abilities in a unified manner (RQ2). Leveraging the efficient tools from polytope algebra, polytopes have a potential to enable characterising the common physical abilities of humans and robots interacting physically when executing certain task (RQ3). Furthermore, being a local metric, polytopes enable capturing the changing physical abilities of humans and robots evolving with their state (RQ5). Finally, by capturing human's changing physical abilities online, polytopes enable evaluating if the operator lacks the physical ability to execute certain task, and in that way evaluate his safety and well-being (RQ4).

Furthermore, representing robot's and human's capacities in the polytope form enables using many efficient tools form the polytope algebra and computational geometry to perform operations over polytopes and to extract useful information concerning the task, for example different performance indicators. Additionally, polytopes can be transformed to more standard forms such as a set of vertices or set of inequalities which can then be used with standard visualisation tools and integrated in different optimisation problems, opening many doors in human-robot collaborative applications.

However, due to the considerable computational complexity of polytope evaluation, their applications in real-time (time-critical) applications is still relatively limited.

1.2 Thesis overview

This thesis focuses on exploiting the polytope representation of human's and robot's physical abilities, as the most accurate characterisation of their abilities that is at the same time capable of fulfilling all the requirements RQ1-RQ5.

The first two chapters (Chapters 2 and 3) concentrate on the common formulations of the physical ability polytopes and their efficient evaluation.

In the effort to unify the vision of different physical abilities of humans are robots, Chapter 2 provides an overview of common polytope representations of their physical abilities. Furthermore, the chapter explores the combination of their individual polytopes in order to characterise their joint abilities as one single system. Finally, the chapter proposes a synthesis of the common polytope formulations of humans and robots in a generic unified form.

Chapter 3 focuses on transforming the introduced polytope formulations to their standard representations. The chapter introduces the generic families of polytope formulations derived from the unified generic formulation described in the previous chapter. The chapter then brings the overview of the standard polytope evaluation algorithms suitable for each one of the proposed families and discusses briefly their computation complexity. The chapter then introduces two new efficient algorithms (VEPOLI^2 and ICHM) developed in the context of this thesis, evaluates their computational efficiency and compares them to the state of the art methods. The experimental validation shows that the proposed algorithms have significantly lower computational complexity with respect to the standard approaches and have the potential to be used in online applications.

The following three chapters (Chapters 4 to 6) present applications of the real-time evaluation of human's and robot's physical abilities in a polytope form, with the aim to improve different aspects of the human-robot collaboration.

Chapter 4 presents the application of the real-time polytope evaluation for enhancing human-robot physical collaboration. The chapter aims to demonstrate that the real-time evaluation of

the physical ability polytopes enables creating the robot control strategies capable of adapting to their changes on the fly. The proposed collaborative robot control strategies are experimentally validated on a task consisting in collaborative carrying of a heavy object, inspired by the LiChiE project. Two experiments are presented: dual robot collaborative carrying and human-robot collaborative carrying. In the dual robot collaborative carrying task, two Franka Emika Panda robots jointly carry 12kg object, largely above their rated capacity (6kg). The results show that by having real-time information about both robots' carrying capacities, the proposed collaborative control strategy adapted to their changes in real-time and successfully distributed the weight of the object during the whole duration of the experiment. In the human-robot collaborative carrying experiment, a human operator and a Franka robot carry 7kg object. The experiment showed that, having online information about their changing capacity enabled to employ the robot's and human's physical potential without compromising their safety. Furthermore, even though neither the robot nor the human would have been able to carry the entire object's weight on their own, by collaborating they were able to accomplish the task.

Chapter 5 explores the idea of using polytope representations of robot's physical abilities as real-time visual feedback to operators. The chapter discusses the potential of such visualisation to provide the operator with real-time insight into the robot's current state and its current physical abilities. The chapter introduces a new polytope formulation developed particularly with the visualisation in mind: the approximation of the robot's reachable space within a time horizon. This polytope formulation represents the Cartesian Space (CS) the robot can reach in a certain horizon time, from its current position, while respecting all the robot's actuator limits, its kinematics and dynamics. By representing the reachable space of CS positions, once visualised, this polytope's interpretation is more intuitive in comparison to the other common polytope formulation representing abstract physical quantities (forces, accelerations, velocities, etc.). Moreover, in the effort to evaluate the effectiveness of the real-time polytope visualisation to the operators, Chapter 5 presents the preliminary work on the development of the testing setup based on the Augmented Reality (AR) tools. This setup presents the foundation for the future work on evaluating the information sharing potential of different polytope formulations and different visualisation modalities, in the context of the human-robot collaboration.

Chapter 6 introduces the application of robot's movement capacity polytopes in the CS trajectory planning. The chapter shows that real-time evaluation of polytopes enables adapting to the changes in the robot's movement capacity on the fly and in that way to exploit its full movement potential. The chapter proposes a CS trajectory planning strategy that evaluates robot's movement capacity in each step of the trajectory execution and recalculates the optimal trajectory using the updated values in real-time. By re-planning the trajectory in real-time, the proposed method is able to react on environmental changes as well. The method's efficiency is confirmed using an experimental benchmark comparison with the state of the art methods. Finally, a practical utility of the proposed method is demonstrated in the experiment of the human-robot collaborative waste sorting, where the proposed method is used to generate efficient and reactive robot's movement on the fly.

Chapter 7 presents the `pycapacity` package, an efficient framework for calculating different physical ability metrics for both humans and robots, based on polytopes and ellipsoids. The package implements several state of the art algorithms for polytope evaluation and manipulation, including VEPOLI^2 and ICHM developed in the context of this thesis, bringing many of them to the interactive online capable execution times. The package is open-source, easy to install, and has relatively extensive documentation. Furthermore, the package is written in an user-friendly way with the aim to facilitate building applications and potentially bring these metrics to the wider community

Chapter 8 brings the thesis conclusion and the discussion on the perspectives of the physical ability polytopes in human-robot collaboration.

1.3 List of publications

This section lists the publications published during the thesis period and the corresponding sections in the manuscript.

[A1] **A. Skuric**, V. Padois, and D. Daney, "On-line force capability evaluation based on efficient polytope vertex search," in *IEEE International Conference on Robotics and Automation*, Xi'an, China, May 2021. [Online]. Available: https://hal.science/hal-02993408, [Video] [GitLab]
MENTIONED ON PAGES 13, 74, 98

[A2] **A. Skuric**, N. Rezzoug, D. Daney, and V. Padois, "Common wrench capability evaluation of a human-robot collaborative system," in *46ème Congrès Société Biomécanique*, vol. 24, Saint Etienne, France: Taylor & Francis, Oct. 2021, S242–S245. DOI: 10.1080/10255842.2021.1978758. [Online]. Available: https://inria.hal.science/hal-03396009
MENTIONED ON PAGE 13

[A3] **A. Skuric**, V. Padois, N. Rezzoug, and D. Daney, "On-line feasible wrench polytope evaluation based on human musculoskeletal models: an iterative convex hull method," *IEEE Robotics and Automation Letters*, 2022. DOI: 10.1109/LRA.2022.3155374. [Online]. Available: https://inria.hal.science/hal-03369576, [Video] [GitLab]
MENTIONED ON PAGES 13, 75, 99

[A4] **A. Skuric**, N. Rezzoug, D. Daney, and V. Padois, "Dynamics aware Cartesian wrench polytope estimation based on human musculoskeletal models," in *48ème Congrès de la Société de Biomécanique*, Grenoble, France: Taylor & Francis, Oct. 2023. [Online]. Available: https://inria.hal.science/hal-04190087
MENTIONED ON PAGES 13, 75

[A5] **A. Skuric**, V. Padois, and D. Daney, "Approximating robot reachable space using convex polytopes," in *15th International Workshop on Human-Friendly Robotics*, Delft, Netherlands, Sep. 2022. [Online]. Available: https://inria.hal.science/hal-03719885, [Web] [GitLab]
MENTIONED ON PAGES 13, 124

[A6] **A. Skuric**, V. Padois, and D. Daney, "Pycapacity: An efficient task-space capacity calculation package for robotics and biomechanics," *Journal of Open Source Software*, vol. 8, no. 89, p. 5670, Sep. 2023. DOI: 10.21105/joss.05670. [Online]. Available: https://joss.theoj.org/papers/10.21105/joss.05670, [Docs] [GitHub]
MENTIONED ON PAGES 14, 164

Parts of the work proposed in the articles [A1] and [A2], discussing methods of calculating common physical abilities of human-robot collaborations, are included in Section 2.3 of Chapter 2. Furthermore, articles [A1, A3, A4] introduce two novel polytope evaluation algorithms (VEPOLI^2 and ICHM), described in Chapter 3, as well as their experimental validation in the context of human-robot physical interaction, described in Chapter 4.

The paper [A5] introduces a new polytope formulation intended for the interactive visualisation to the operator: the convex polytope of the robot's reachable space within a horizon. This work is included in Section 5.1 of Chapter 5.

The open-source Python package `pycapacity`, introduced in [A6], is described in Chapter 7 of the manuscript.

Other publications during the thesis

These publications were published in the thesis period, but correspond to side research projects.

[B1] Z. Song, **A. Skuric**, and K. Ji, "A Recursive Watermark Method for Hard Real-Time Industrial Control System Cyber-Resilience Enhancement," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 2, pp. 1030–1043, 2020. DOI: 10.1109/TASE.2019.2963257, [IEEE Spectrum] [GitHub]

[B2] **A. Skuric**, H. S. Bank, R. Unger, O. Williams, and D. González-Reyes, "SimpleFOC: A Field Oriented Control (FOC) Library for Controlling Brushless Direct Current (BLDC) and Stepper Motors," *Journal of Open Source Software*, vol. 7, no. 74, p. 4232, 2022. DOI: 10.21105/joss.04232. [Online]. Available: https://doi.org/10.21105/joss.04232, [Web] [GitHub]

Chapter 2

Polytope representation of physical abilities

As described in the introduction chapter, polytope based representations of different physical abilities (physical ability polytopes) of humans and robots enable an accurate characterisation of their individual abilities. Furthermore, by expressing their individual abilities in the polytope form, their common abilities when collaborating can be computed using efficient polytope algebra operations and represented in polytope form as well. Numerous polytope-based physical ability characterisations have been developed in the literature, both for humans and robots. Therefore, the aim of this chapter is to present an overview of the common physical ability polytopes, applicable to both humans and robots, in the attempt to propose an unified vision of their individual abilities.

Section 2.1 brings an overview of the common polytope formulations for robotic manipulators, while Section 2.2 presents the common metrics for humans, based on their musculoskeletal models. Section 2.3 then discusses using different polytope algebra operators to calculate the joint physical abilities of humans and robots when engaged in physical interactions to accomplish specific tasks. Finally, the Section 2.4 concludes the chapter and provides a structured synthesis of common polytope formulations, proposing a single generic polytope formulation encompassing all the common formulations for both humans and robots.

2.1 Physical ability polytope formulations for robotic manipulators

Physical ability characterisations based on polytopes are most accurate local, state dependent, metrics for the performance analysis of robotic systems [31, 37]. They describe a mapping between the robot's actuator limits (torques, accelerations , velocities, etc.) and the limits of different task space variables (wrenches, accelerations, velocities, etc.) .

The aim of this section is to provide an overview of common polytope formulations for robotic systems in an unified view. Therefore, Section 2.1.1 bringing a quick overview of the dynamics and kinematics equations for robotic systems, used for different polytope definitions. Definitions of common polytope characterisations of physical abilities are then described in subsequent sections (Section 2.1.2 - 2.2.6).

2.1.1 Dynamics and kinematics relationships

The actuation limits, of a robotic manipulator with n joints, are commonly specified as independent ranges of achievable actuator positions $\boldsymbol{q} \in \mathbb{R}^n$, velocities $\dot{\boldsymbol{q}} \in \mathbb{R}^n$, actuator torques

$\boldsymbol{\tau} \in \mathbb{R}^n$ and sometimes even torque derivatives $\dot{\boldsymbol{\tau}} \in \mathbb{R}^n$.

$$\boldsymbol{q} \in [\boldsymbol{q}_{min}, \boldsymbol{q}_{max}], \quad \dot{\boldsymbol{q}} \in [\dot{\boldsymbol{q}}_{min}, \dot{\boldsymbol{q}}_{max}], \tag{2.1a}$$

$$\boldsymbol{\tau} \in [\boldsymbol{\tau}_{min}, \boldsymbol{\tau}_{max}], \quad \dot{\boldsymbol{\tau}} \in [\dot{\boldsymbol{\tau}}_{min}, \dot{\boldsymbol{\tau}}_{max}] \tag{2.1b}$$

The dynamical model, of a standard serial robotic manipulator with a fixed base, is commonly defined as

$$M(\boldsymbol{q})\ddot{\boldsymbol{q}} + C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) + J^T(\boldsymbol{q})\boldsymbol{f} = \boldsymbol{\tau} \tag{2.2}$$

where $M(\boldsymbol{q}) \in \mathbb{R}^{n\times n}$ and $C(\boldsymbol{q}, \dot{\boldsymbol{q}}) \in \mathbb{R}^{n\times n}$ are the inertia and Coriolis matrices, $\boldsymbol{g}(\boldsymbol{q}) \in \mathbb{R}^n$ is the state dependent gravity vector, $J(\boldsymbol{q}) \in \mathbb{R}^{m\times n}$ is the state dependant Jacobian matrix, $\boldsymbol{f} \in \mathbb{R}^m$ is the vector of applied external wrenches and $\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}$ are robot's joint positions, velocities and accelerations respectively.

Furthermore, the mapping between the n dimensional space of robot's actuator (joint) positions $\boldsymbol{q} \in \mathbb{R}^n$ and the m dimensional task space positions $\boldsymbol{x} \in \mathbb{R}^m$ can be found through the robot's forward kinematics

$$\boldsymbol{x} = f_{fk}(\boldsymbol{q}) \tag{2.3}$$

whereas the task space velocity $\dot{\boldsymbol{x}} \in \mathbb{R}^m$, acceleration $\ddot{\boldsymbol{x}} \in \mathbb{R}^m$ and jerk $\dddot{\boldsymbol{x}} \in \mathbb{R}^m$ can be found through the state dependent $\{\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}\}$ and nonlinear mapping

$$\dot{\boldsymbol{x}} = J(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{2.4a}$$

$$\ddot{\boldsymbol{x}} = J(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{J}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} \tag{2.4b}$$

$$\dddot{\boldsymbol{x}} = J(\boldsymbol{q})\dddot{\boldsymbol{q}} + 2\dot{J}(\boldsymbol{q}, \dot{\boldsymbol{q}})\ddot{\boldsymbol{q}} + \ddot{J}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})\dot{\boldsymbol{q}} \tag{2.4c}$$

were $J(\boldsymbol{q})$ is the Jacobian matrix

$$J(\boldsymbol{q}) = \frac{\partial f_{fk}(\boldsymbol{q})}{\partial \boldsymbol{q}} \tag{2.5}$$

while $\dot{J}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ and $\ddot{J}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$ are its first and second time derivatives.

It is worth noting that in general case, m dimensional task space might include positions and orientations, for example if the task space corresponds to the Cartesian Space (CS), where $m = 6$. When the task space includes orientations, the task space pose (including the position and orientation) is often expressed is the homogeneous matrix form $\boldsymbol{x} \in SE(3)$. Then the task space velocity vector $\dot{\boldsymbol{x}} \in \mathbb{R}^6$ becomes the CS *twist vector* $\boldsymbol{\nu}$, expressing the translation and rotation velocity at the same time. However, in such cases, the twist $\boldsymbol{\nu}$ does no longer correspond to the time derivative of the task space pose $\boldsymbol{\nu} \neq \frac{d\boldsymbol{x}}{dt}$. Despite this distinction, for simplicity reasons, this work uses the notation $\dot{\boldsymbol{x}}$, $\ddot{\boldsymbol{x}}$, and $\dddot{\boldsymbol{x}}$ to represent velocity, acceleration, and jerk of the task space variables, respectively.

When considering certain fixed robot state $\{\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}\}$, robot's dynamical model (2.2) and the joint to task space mapping (2.4) become linear, as the state dependent matrices become fixed. Using these linear and affine mappings the achievable sets of different task space variables (ex. velocity $\dot{\boldsymbol{x}}$, accelerations $\ddot{\boldsymbol{x}}$ or wrenches $\boldsymbol{f}$), given the range of robot's actuator limits (2.1), can be found through the equations (2.2) and (2.4). The obtained achievable sets then have a form of convex polytopes as the mapping between the robot limits and the task space variables is linear.

Kinematic joint limits As a precise robot's dynamical model is sometimes hard to obtain, instead of specifying the actuator torque limits (2.1b), the manufacturers often rather specify robot's actuator limits in a kinematic form, considering them to have limited and constant

acceleration $\ddot{\boldsymbol{q}} \in \mathbb{R}^n$ and jerk $\dddot{\boldsymbol{q}} \in \mathbb{R}^n$ limits.

$$\ddot{\boldsymbol{q}} \in [\ddot{\boldsymbol{q}}_{min}, \ddot{\boldsymbol{q}}_{max}], \quad \dddot{\boldsymbol{q}} \in [\dddot{\boldsymbol{q}}_{min}, \dddot{\boldsymbol{q}}_{max}] \tag{2.6}$$

Using these kinematic actuator limits, the linearisation procedure of the mapping (2.4) around given robot state $\{\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}\}$ can then be used to obtain the polytope shaped achievable sets of kinematic task space variables (ex. velocity $\dot{\boldsymbol{x}}$, acceleration $\ddot{\boldsymbol{x}}$, jerk $\dddot{\boldsymbol{x}}$).

Many well known polytope formulations for characterising the physical abilities of robotic manipulators are proposed in the literature, characterising the influence of robots dynamics (2.1) and kinematics joint limits (2.6) on the task space variables through its linearised dynamics (2.2) and kinematics (2.4) equations. Some of the most common polytope formulations are listed in the following sections.

2.1.2 Velocity polytope

Achievable task space velocity capacity is a common physical ability metric for robotic manipulators describing its ability to generate task space velocities $\dot{\boldsymbol{x}}$ in different directions in space, given its joins space velocity $\dot{\boldsymbol{q}}$ limits. Robot's velocity capacity is commonly know as its manipulability or dexterity capacity as well. This metric has been first introduced by Yoshikawa [22] in the ellipsoid form and later extended to the polytope form [10, 30].

For any fixed robot configuration $\boldsymbol{q}$ the ranges of joint velocity limits

$$\dot{\boldsymbol{q}} \in [\dot{\boldsymbol{q}}_{min}, \dot{\boldsymbol{q}}_{max}] \tag{2.7}$$

can be projected to the m dimensional task space using the Jacobian matrix $J(\boldsymbol{q})$, obtaining the convex polytope of achievable task space velocities

$$\mathcal{P}_v(\boldsymbol{q}) = \{\dot{\boldsymbol{x}} \in \mathbb{R}^m \mid \dot{\boldsymbol{q}} \in [\dot{\boldsymbol{q}}_{min}, \dot{\boldsymbol{q}}_{max}], \quad \dot{\boldsymbol{x}} = J(\boldsymbol{q})\dot{\boldsymbol{q}}\} \tag{2.8}$$

Velocity capacity metrics are important tools for evaluating robot's movement capacity in different directions in space for a given configuration $\boldsymbol{q}$, while the polytope form is their exact representation. Different variants of polytope formulations are commonly used for robot redundancy resolution, by finding the robot configurations that maximise the robot's movement capacity and avoid singular positions [83, 84], which can be seen as a zero velocity capacity in certain directions in space [24]. They have also been used for path planning [85, 86] and object grasping [87, 88].

The polytope representation though is less commonly used in time-critical applications, due to their computational complexity. However, Long and Padır [34] have shown that polytopes can be used for whole body humanoid robot control in real-time. Recently, Zolotas *et al.* [79] have shown their potential to be used for real-time visualisation to the user in the case of robot teleportation.

2.1.3 Kinematic Acceleration and Jerk polytope

In a similar manner to the achievable velocity polytope $\mathcal{P}_v$, the achievable task space acceleration $\ddot{\boldsymbol{x}}$ and jerk $\dddot{\boldsymbol{x}}$ sets can be expressed using the kinematic mapping (2.4).

Given the limits of the actuator accelerations $\ddot{\boldsymbol{q}}$ and a given robot state $\{\boldsymbol{q}, \dot{\boldsymbol{q}}\}$, the achievable polytope $\mathcal{P}_a$ of task space accelerations $\ddot{\boldsymbol{x}}$ can be found using the mapping (2.4b).

$$\mathcal{P}_a(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \{\ddot{\boldsymbol{x}} \in \mathbb{R}^m \mid \ddot{\boldsymbol{q}} \in [\ddot{\boldsymbol{q}}_{min}, \ddot{\boldsymbol{q}}_{max}], \quad \ddot{\boldsymbol{x}} = J(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{a}_b(\boldsymbol{q}, \dot{\boldsymbol{q}})\} \tag{2.9}$$

where $\boldsymbol{a}_b(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is considered as a constant bias term for any given robot state $\{\boldsymbol{q}, \dot{\boldsymbol{q}}\}$

$$\boldsymbol{a}_b(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \dot{J}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} \tag{2.10}$$

The achievable task space jerk polytope $\mathcal{P}_j$ can be found for any given robot state $\{\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}\}$, by considering a limited range of actuator jerks $\dddot{\boldsymbol{q}}$ and using the mapping (2.4c)

$$\mathcal{P}_j(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) = \{\dddot{\boldsymbol{x}} \in \mathbb{R}^m \mid \dddot{\boldsymbol{q}} \in [\dddot{\boldsymbol{q}}_{min}, \dddot{\boldsymbol{q}}_{max}], \quad \dddot{\boldsymbol{x}} = J(\boldsymbol{q})\dddot{\boldsymbol{q}} + \boldsymbol{j}_b(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})\} \tag{2.11}$$

where $\boldsymbol{j}_b(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$ is a constant bias term for any fixed robot state $\{\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}\}$

$$\boldsymbol{j}_b(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) = 2\dot{J}(\boldsymbol{q}, \dot{\boldsymbol{q}})\ddot{\boldsymbol{q}} + \ddot{J}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})\dot{\boldsymbol{q}} \tag{2.12}$$

An example application of the jerk and acceleration polytopes is described by Grandguillaume *et al.* [89] where the polytopes are used to find the optimal tool orientation for 5-axis milling.

2.1.4 Precision (maximal positioning error) polytope

Robotic manipulators often have limited positioning precision of their actuators. Actuator precision depends on many different factors, such as position sensor used, motor gearing, motion control strategy etc [31, 37]. However, in most cases the manufacturers are able to provide the limits on the maximal positioning error for each of the actuators within a certain range

$$\delta\boldsymbol{q} \in [\delta\boldsymbol{q}_{min}, \delta\boldsymbol{q}_{min}] \tag{2.13}$$

Depending on the robot's configuration $\boldsymbol{q}$ the actuator position error $\delta\boldsymbol{q}$ will produce different task space positioning errors, their relationship can be calculated through the kinematic mapping (2.4a)

$$\delta\boldsymbol{x} = J(\boldsymbol{q})\delta\boldsymbol{q} \tag{2.14}$$

where $\delta\boldsymbol{x} \in \mathbb{R}^m$, in general case presents both position and orientation error.

The limits of the task space positioning error $\delta\boldsymbol{x}$ in a certain robot configuration $\boldsymbol{q}$, given the limits (2.13) specified by the manufacturer, can be expressed by a convex polytope $\mathcal{P}_\delta$

$$\mathcal{P}_\delta(\boldsymbol{q}) = \{\delta\boldsymbol{x} \in \mathbb{R}^m \mid \delta\boldsymbol{q} \in [\delta\boldsymbol{q}_{min}, \delta\boldsymbol{q}_{max}], \quad \delta\boldsymbol{x} = J(\boldsymbol{q})\delta\boldsymbol{q}\} \tag{2.15}$$

2.1.5 Wrench/Force polytope

Task space force (or more generally wrench) capacity metric is one of the most well known physical capacity metrics for robotics manipulators alongside the velocity (manipulability) capacity. It has been first introduced by Chiacchio *et al.* [35] in a form of an ellipsoid as well as in the polytope form.

This metric establishes the relationship between the robot's actuator torque limits $\boldsymbol{\tau}$ and the achievable set of the task space wrenches $\boldsymbol{f}$. Given the robot's manipulators dynamics equation (2.2)

$$\underbrace{M(\boldsymbol{q})\ddot{\boldsymbol{q}} + C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q})}_{\boldsymbol{\tau}_b(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})} + J^T(\boldsymbol{q})\boldsymbol{f} = \boldsymbol{\tau} \tag{2.16}$$

for any given robot state $\{\boldsymbol{q}, \dot{\boldsymbol{q}}\}$, and given joint acceleration $\ddot{\boldsymbol{q}}$ to be achieved, the effects of the robot's motion and the gravity can be expressed as a constant torque vector $\boldsymbol{\tau}_b(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$.

Then the polytope of achievable task space wrenches $\boldsymbol{f}$ can be expressed as

$$\mathcal{P}_f(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) = \left\{ \boldsymbol{f} \in \mathbb{R}^m \mid \boldsymbol{\tau} \in [\boldsymbol{\tau}_{min}, \boldsymbol{\tau}_{max}], \quad J^T(\boldsymbol{q})\boldsymbol{f} = \boldsymbol{\tau} - \boldsymbol{\tau}_b(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) \right\} \tag{2.17}$$

If achievable wrench capacity $\mathcal{P}_f$ is evaluated in kinetostatic conditions $\dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}} \approx \boldsymbol{0}$, the torque bias vector $\boldsymbol{\tau}_b$ corresponds to the gravity vector $\boldsymbol{g}(\boldsymbol{q})$

$$\boldsymbol{\tau}_b(\boldsymbol{q}, \boldsymbol{0}, \boldsymbol{0}) = \boldsymbol{g}(\boldsymbol{q}) \tag{2.18}$$

As wrench polytopes describe the feasible range of task space wrenches that the robot can apply in certain configuration, they can be used in many different analysis applications as a performance metric, as well as in robot control in order to guarantee that the task requirements are within robot's capacity. An example application of wrench capacity polytopes for performance analysis of robotic excavator is described by Zou *et al.* [90]. Ferrolho *et al.* [91] have described the application of wrench polytopes in offline dynamic trajectory planning. On the other hand, the applications of wrench polytopes in real-time applications are less common, mostly due to their computational complexity. However, Orsolino *et al.* [92] have recently showed the application of wrench polytopes in real-time control of legged robots.

2.1.6 Acceleration polytope

The kinematic acceleration capacity described in Section 2.1.3 considers constant and independent acceleration capacity for each robot's actuator (2.6). However, the true acceleration limits of the robot's actuators depend on their torque capacity (2.1), as well as on the robot's inertia $M(\boldsymbol{q})$ which is highly configuration dependent.

Given the robot's dynamics equations (2.2)

$$M(\boldsymbol{q})\ddot{\boldsymbol{q}} + \underbrace{C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) + J^T(\boldsymbol{q})\boldsymbol{f}}_{\boldsymbol{\tau}_b(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{f})} = \boldsymbol{\tau} \tag{2.19}$$

for any given robot state $\{\boldsymbol{q}, \dot{\boldsymbol{q}}\}$ and applied external wrench $\boldsymbol{f}$, the effects of the robot's motion, the gravity, as well as the applied external wrenches, can be grouped in a constant torque vector $\boldsymbol{\tau}_b(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{f})$. Then the actuator acceleration $\ddot{\boldsymbol{q}}$ can be expressed in relationship to the applied joint torque $\boldsymbol{\tau}$

$$\ddot{\boldsymbol{q}} = M(\boldsymbol{q})^{-1}\boldsymbol{\tau} - M(\boldsymbol{q})^{-1}\boldsymbol{\tau}_b(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{f}) \tag{2.20}$$

Finally, the actuator accelerations $\ddot{\boldsymbol{q}}$ can be transformed to the task space using the mapping (2.4b)

$$\ddot{\boldsymbol{x}} = J(\boldsymbol{q})M(\boldsymbol{q})^{-1}\boldsymbol{\tau} - \underbrace{J(\boldsymbol{q})M(\boldsymbol{q})^{-1}\boldsymbol{\tau}_b(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{f}) + \dot{J}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}}_{\boldsymbol{a}_b(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{f})} \tag{2.21}$$

where $\boldsymbol{a}_b(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{f})$ presents a constant bias vector for any fixed joint state $\{\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{f}\}$. The polytope $\mathcal{P}_a$ of achievable task space accelerations $\ddot{\boldsymbol{x}}$ can then be expressed as

$$\mathcal{P}_a(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{f}) = \left\{ \ddot{\boldsymbol{x}} \in \mathbb{R}^m \mid \boldsymbol{\tau} \in [\boldsymbol{\tau}_{min}, \boldsymbol{\tau}_{max}], \ \ddot{\boldsymbol{x}} = J(\boldsymbol{q})M(\boldsymbol{q})^{-1}\boldsymbol{\tau} - \boldsymbol{a}_b(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{f}) \right\} \tag{2.22}$$

As opposed to the kinematic acceleration polytope (2.9) described in Section 2.1.3, this formulation not only represents more realistic acceleration capacity but it is able to account for the effect of the robot's motion, the effects of the gravity as well as its potential payload etc.

This metric has been initially introduced by Chiacchio [36] in the ellipsoid form and later extended to the polytope form by Rosenstein and Grupen [93]. An application of this metric in off-line analysis has been proposed by Jang and Lee [94], using the acceleration polytopes to analyse and optimise multi-fingered object grasping. Acceleration polytopes are used in real-time robot control applications as well. For example, Jeong and Takahashi [95] proposed an application of acceleration polytopes for robotic manipulators in order to optimise the robot's braking characteristics in real-time and to reduce the impact forces.

2.1.7 Stiffness polytope

A different well known task space physical ability metrics for robotic manipulators is so-called stiffness capacity and is related to the impedance control. Consider that a certain frame of the robotic manipulator (ex end-effector frame) is controlled to have a behaviour of virtual spring with the task space stiffness $K_c \in \mathbb{R}^{m \times m}$

$$\boldsymbol{f} = K_c \Delta \boldsymbol{x} \tag{2.23}$$

where $\boldsymbol{f} \in \mathbb{R}^m$ is the external wrench applied to the robot (by the environment or a human operator) and $\Delta \boldsymbol{x} \in \mathbb{R}^m$ is the resulting displacement corresponding to the virtual spring with stiffness $K_c \in \mathbb{R}^{m \times m}$. In case of standard impedance control for a robotic manipulator in Cartesian Space (CS), controlling both position and orientation ($m = 6$), the displacement vector $\Delta \boldsymbol{x}$ contains both translation and orientation displacements produced by the wrench $\boldsymbol{f}$. Given a certain range of external wrenches that are expected to be applied by the environment (or the human operator)

$$\boldsymbol{f} \in [\boldsymbol{f}_{min}, \boldsymbol{f}_{max}] \tag{2.24}$$

the maximal task space displacement $\Delta \boldsymbol{x}$ (position and orientation) can be expressed as a convex polytope

$$\mathcal{P}_\Delta = \{\Delta \boldsymbol{x} \in \mathbb{R}^m \mid \boldsymbol{f} \in [\boldsymbol{f}_{min}, \boldsymbol{f}_{max}], \quad K_c \Delta \boldsymbol{x} = \boldsymbol{f}\} \tag{2.25}$$

If the stiffness control is done in Joint Space (JS), where the n robot's actuators are controlled in a way to produce a virtual spring behavior with a stiffness $K_j \in \mathbb{R}^{n \times n}$

$$\boldsymbol{\tau} = K_j \Delta \boldsymbol{q} \tag{2.26}$$

This JS stiffness matrix K_j can be used to find the robot state dependent task space stiffness K_c [96]

$$K_c(\boldsymbol{q})^{-1} = J(\boldsymbol{q}) K_j^{-1} J(\boldsymbol{q})^T \tag{2.27}$$

The matrix K_j is a square $n \times n$ matrix, which is in most cases diagonal and has a full rank, while the Jacobian matrix is $m \times n$ shaped and configuration dependent. If the Jacobian matrix $J(\boldsymbol{q})$ is approaching to its singularity, as the singularity can be seen as a reduction of the task space dimension m, the resulting robotic manipulator's stiffness would be approaching infinity in that direction.

Given the same range of expected external wrenches (2.24), the polytope of maximal task space displacements $\Delta \boldsymbol{x}$ can then be expressed as

$$\mathcal{P}_\Delta(\boldsymbol{q}) = \{\Delta \boldsymbol{x} \in \mathbb{R}^m \mid \boldsymbol{f} \in [\boldsymbol{f}_{min}, \boldsymbol{f}_{max}], \quad K_c(\boldsymbol{q}) \Delta \boldsymbol{x} = \boldsymbol{f}\} \tag{2.28}$$

Neither of these formulations takes in consideration the robot's wrench capacity though, they both consider that the robot is capable of generating all the wrenches in the expected range (2.24). In order to make sure that the robot's wrench capacity is not exceeded when calculating

the maximal task space displacement polytope $\mathcal{P}_\Delta$ (both for JS and task space stiffness control), the external wrench range (2.24) needs to be intersected with the robot's wrench polytope $\mathcal{P}_f$ described in the Section 2.1.5. The polytope $\mathcal{P}_\Delta$ of task space displacement $\Delta\boldsymbol{x}$ can be expressed as

$$\mathcal{P}_\Delta(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}}) = \{\Delta\boldsymbol{x} \in \mathbb{R}^m \mid \boldsymbol{f} \in [\boldsymbol{f}_{min},\boldsymbol{f}_{max}] \cap \mathcal{P}_f(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}}),\quad K_c(\boldsymbol{q})\Delta\boldsymbol{x} = \boldsymbol{f}\} \tag{2.29}$$

where the wrenches $\boldsymbol{f} \in [\boldsymbol{f}_{min},\boldsymbol{f}_{max}] \cap \mathcal{P}_f(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}})$ represent all the wrenches in the range (2.24) that the robot can physically generate, given its wrench capacity polytope $\mathcal{P}_f(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}})$.

If the maximal displacement polytope $\mathcal{P}_\Delta$ is calculated only in relationship to the robot's wrench capacity $\mathcal{P}_f$, without specifying the expected range of external wrenches (2.24), this polytope can be expressed in somewhat simplified form

$$\mathcal{P}_\Delta(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}}) = \left\{\Delta\boldsymbol{x} \in \mathbb{R}^m \mid \boldsymbol{\tau} \in [\boldsymbol{\tau}_{min},\boldsymbol{\tau}_{max}],\ J^T(\boldsymbol{q})K_c(\boldsymbol{q})\Delta\boldsymbol{x} = \boldsymbol{\tau} - \boldsymbol{\tau}_b(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}})\right\} \tag{2.30}$$

The stiffness polytope $\mathcal{P}_\Delta$ then represents the maximal task space displacement $\Delta\boldsymbol{x}$ before the robot's torque limits (2.1b) are reached, this polytope formulation is also called *stiffness feasibility region* [29, 97].

2.2 Physical ability polytope formulations for human musculoskeletal models

Polytope based characterisations of humans physical abilities calculated using musculoskeletal models are important tool for local performance and ergonomics analysis tools in biomechanics. In the case of musculoskeletal models, the capacity polytopes establish a relationship between the limitation of muscle-tendon units as the body actuators (muscles' contraction forces, extension velocities, etc), the limits of human's joints (joint positions, velocities, etc.) and the achievable sets of task related variables (task space velocities, accelerations, external wrenches, etc.).

The aim of this section is to provide an overview of common polytope based physical ability formulations for humans in an unified view. Therefore, Section 2.2.1 brings an overview of the dynamics and kinematics equations for musculoskeletal models, used for different metrics definitions. Definitions of common polytope formulations are then described in subsequent sections (Section 2.2.3 - 2.2.6).

2.2.1 Dynamics and kinematics relationships

The dynamical equation of a general musculoskeletal system, expressed in n dimensional, Joint Space (JS) can be formulated as

$$M(\boldsymbol{q})\ddot{\boldsymbol{q}} + C(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) + J^T(\boldsymbol{q})\boldsymbol{f} = \boldsymbol{\tau} \tag{2.31}$$

where $\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}} \in \mathbb{R}^n$ are the joint level generalised positions, velocities and accelerations. $M(\boldsymbol{q}) \in \mathbb{R}^{n\times n}$ is the inertia matrix, $C(\boldsymbol{q},\dot{\boldsymbol{q}}) \in \mathbb{R}^{n\times n}$ is the Coriolis-centrifugal matrix, $\boldsymbol{g}(\boldsymbol{q}) \in \mathbb{R}^n$ is the joint torque produced by gravity, $\boldsymbol{\tau} \in \mathbb{R}^n$ is the joint torque vector generated by the muscle-tendon units, $J(\boldsymbol{q}) \in \mathbb{R}^{m\times n}$ is the end effector Jacobian matrix and $\boldsymbol{f} \in \mathbb{R}^m$ is an m-dimensional external task space wrench vector.

As for the robotic manipulators, the kinematics mapping between human's n dimensional JS and the m dimensional task space (often Cartesian Space (CS)) variables is given through the

forward kinematics equations $f_{fk}(\cdot)$

$$\boldsymbol{x} = f_{fk}(\boldsymbol{q}) \tag{2.32}$$

while the velocity kinematics mapping is described through the Jacobian matrix $J(\boldsymbol{q})$ and its time derivatives

$$\dot{\boldsymbol{x}} = J(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{2.33a}$$

$$\ddot{\boldsymbol{x}} = J(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{J}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} \tag{2.33b}$$

Muscle limitations

Each of the human's n joint torques $\boldsymbol{\tau} \in \mathbb{R}^n$ is generated by a combination of his d muscle-tendon units. One of the most well known muscle models was introduced by Hill [98] and later refined by Zajac [99], where the muscle-tendon unit is approximated by a tensile force composed of an active and a passive component

$$F_i = f_i^A(l_i, \dot{l}_i) F_{iso,i} \alpha_i + \underbrace{f_i^P(l_i) F_{iso,i}}_{F_{p,i}}, \quad \alpha_i \in [0, 1] \tag{2.34}$$

f_i^A and f_i^P are active and passive scaling functions depending on the muscle length l_i and contraction/extension velocity $\dot{l}_i$. $F_{iso,i}$ is the maximal isometric force the muscle can generate and α_i is the muscle activation level. For a set of d muscles, with specified muscle lengths $\boldsymbol{l} \in \mathbb{R}^d$ and contraction velocities $\dot{\boldsymbol{l}} \in \mathbb{R}^d$ the achievable set of muscle tensile forces $\boldsymbol{F} \in \mathbb{R}^d$ has a range

$$\boldsymbol{F} \in [\boldsymbol{F}_p,\ \boldsymbol{F}_m] \tag{2.35}$$

where $\boldsymbol{F}_p \in \mathbb{R}^d$ is the vector of passive forces obtained for $\boldsymbol{\alpha} = \boldsymbol{0}$ and $\boldsymbol{F}_m \in \mathbb{R}^d$ is the maximal muscle force vector obtained for $\boldsymbol{\alpha} = 1$.

$$\begin{aligned} F_{p,i} &= f_i^P(l_i) F_{iso,i} \\ F_{m,i} &= \left(f_i^A(l_i, \dot{l}_i) + f_i^P(l_i) \right) F_{iso,i} \end{aligned} \tag{2.36}$$

The joint torques $\boldsymbol{\tau}$, generated by the muscle tensile force $\boldsymbol{F}$, can then be calculated using the Moment arm matrix [100]

$$\boldsymbol{\tau} = \underbrace{-L^T(\boldsymbol{q})}_{N(\boldsymbol{q})} \boldsymbol{F} \tag{2.37}$$

where $N(\boldsymbol{q}) \in \mathbb{R}^{n \times d}$ is the moment arm matrix, corresponding to the negative transpose of the muscle length Jacobian matrix $L(\boldsymbol{q}) \in \mathbb{R}^{d \times n}$, relating the space of joint and muscle length velocities

$$\dot{\boldsymbol{l}} = L(\boldsymbol{q})\dot{\boldsymbol{q}}, \quad L_{ij} = \frac{\partial l_i}{\partial q_j} \tag{2.38}$$

Finally, the negative sign in equation (2.37) makes the force applied in the length shortening direction of the muscle positive.

Combining equations (2.31) and (2.37), the musculoskeletal model dynamics relationship, taking in consideration d muscle tensile forces $\boldsymbol{F}$, can be expressed as

$$M(\boldsymbol{q})\ddot{\boldsymbol{q}} + C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) + J^T(\boldsymbol{q})\boldsymbol{f} = -L^T(\boldsymbol{q})\boldsymbol{F} \tag{2.39}$$

This state dependant $\{\boldsymbol{q}, \dot{\boldsymbol{q}}\}$ equation establishes the relationship between the muscle tensile forces $\boldsymbol{F}$, the applied external wrenches $\boldsymbol{f}$, the influence of the gravity $\boldsymbol{g}(\boldsymbol{q})$ and different influences of human's movement.

In addition to the limitations of the muscle tensile forces $\boldsymbol{F}$, muscle-tendon units have limited extension velocity $\dot{\boldsymbol{l}}$. These limits are often expresses in a form of independent ranges as well

$$\dot{\boldsymbol{l}} \in [\dot{\boldsymbol{l}}_{min}, \dot{\boldsymbol{l}}_{max}] \tag{2.40}$$

Joint limitations

Human joint torque $\boldsymbol{\tau} \in \mathbb{R}^n$ is generated by the muscle tensile forces $\boldsymbol{F} \in \mathbb{R}^d$ rather than dedicated joint level actuators, as opposed to the robotic manipulator case. However, for different analysis applications it is common to consider human models to be actuated by the set of independent joint torques $\boldsymbol{\tau}$ and considering their limits to have a form of independent ranges [39]

$$\boldsymbol{\tau} \in [\boldsymbol{\tau}_{min}, \boldsymbol{\tau}_{max}] \tag{2.41}$$

This simplification enables studying the musculoskeletal systems as robotic systems and enables the use of different physical ability metrics designed for robotic manipulators like the ones described in Section 2.1.

Similarly to the joint torque limits, that are a simplification of the real limits, studies have shown that the limits of human's n joint positions $\boldsymbol{q} \in \mathbb{R}^n$ can also be approximated in a form of independent ranges [101]

$$\boldsymbol{q} \in [\boldsymbol{q}_{min}, \boldsymbol{q}_{max}] \tag{2.42}$$

even though the real joint range of motion constraints might be non-linearly inter-dependent [102]. In addition to the limited ranges of joint positions $\boldsymbol{q}$, the limited and independent ranges of achievable joint velocity $\dot{\boldsymbol{q}}$ and acceleration $\ddot{\boldsymbol{q}}$ are often considered as well [103]

$$\dot{\boldsymbol{q}} \in [\dot{\boldsymbol{q}}_{min}, \dot{\boldsymbol{q}}_{max}], \quad \ddot{\boldsymbol{q}} \in [\ddot{\boldsymbol{q}}_{min}, \ddot{\boldsymbol{q}}_{max}] \tag{2.43}$$

The independent ranges of joint velocities $\dot{\boldsymbol{q}}$ and accelerations $\ddot{\boldsymbol{q}}$ are a simplification of the real limits which are determined by the limits of muscle extension velocity $\dot{\boldsymbol{l}}$ and limits of the muscle tensile forces $\boldsymbol{F}$. However, making these assumptions enables studying physical abilities of humans using the metrics developed for robotic manipulators, like the ones described in Section 2.1.

Finally, even though all the dynamics and kinematics equations of human musculoskeletal models are highly nonlinear and state dependant, for any given human state (joint positions and velocities) $\{\boldsymbol{q}, \dot{\boldsymbol{q}}\}$ all the matrices become fixed, resulting in affine and linear relationships. These, now linear, equations can be used to evaluate the influence of different muscle and JS limits on the achievable sets of different task space variables, such as task space velocities $\dot{\boldsymbol{x}}$, accelerations $\ddot{\boldsymbol{x}}$, external wrenches $\boldsymbol{f}$ and similar, in a form of convex polytopes.

2.2.2 Parallel with Multi-Link Cable-Driven Robots

Multi-link Cable-Driven Robots (MCDRs) [104] are a family of robotic systems having the serial robotic manipulator structure while being actuated by cables. As the cable actuators are capable of applying unidirectional tensile forces, there is a natural parallel with musculoskeletal models. Lau *et al.* [105] have even proposed a generalised modelling framework for MCDR modelling and demonstrated that it can be used for musculoskeletal models as well. A simplified planar example of a human musculoskeletal model and the MCDR model is shown on Figure 2.1.

One of the main differences comes from the actuator point of view. When it comes to MCDRs, the dynamics of the actuators is often neglected, considering them capable of instantaneously applying a range of d tensile forces $\boldsymbol{t} \in \mathbb{R}^d$.

$$\boldsymbol{t} \in [\boldsymbol{0},\ \boldsymbol{t}_{max}] \tag{2.44}$$

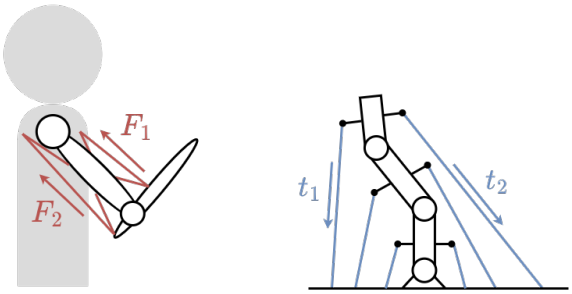

FIGURE 2.1: An illustrative planar example of a human musculoskeletal model and a MCDR. On the left human arm is modeled with two joints and six muscles, where each muscle can apply contraction forces F_i. On the right, a MCDR robot is modeled with 3 joints and 6 cables, which can apply tension forces t_i.

Additionally, this range is often considered constant and is not state dependent. For musculoskeleral models, on the other hand, the ranges of achievable muscle tension forces $\boldsymbol{F}$ are subject to many different biomechanical and biological factors, such as muscle length $\boldsymbol{l}$ and extension velocity $\dot{\boldsymbol{l}}$ as well the effects of fatigue and temperature. Therefore the cables of the MCDRs can be seen as idealised muscles which tensile force capacity stays constant.

Consequently, the physical ability metrics defined for the musculoskeleral models have the same formulation as the ones used to asses the performance of MCDRs.

2.2.3 Wrench/Force polytope

Task space force (or more generally wrench) capacity $\boldsymbol{f} \in \mathbb{R}^m$ is a well established metrics in biomechanics. It has been defined in two manners: a simplified formulation where the torques of n human joints $\boldsymbol{\tau} \in \mathbb{R}^n$ are considered independent and limited (2.41) and in a more complete form where the human's joint torques $\boldsymbol{\tau}$ are generated by d muscle tensile forces $\boldsymbol{F} \in \mathbb{R}^d$ which are limited in their respective ranges (2.35).

Torque based polytopes [58, 106] of achievable task space wrenches $\boldsymbol{f}$ have the same formulation as the force polytopes for robotic manipulators described in Section 2.1.5. Therefore all the considerations for their evaluation are equivalent to those for robotic manipulators.

However, real limits of human joint torques cannot be expressed as independent ranges (2.41) as they are generated by d muscle tensile forces $\boldsymbol{F}$. The true limits of the joint torques $\boldsymbol{\tau}$ are polytope shaped as a consequence of the affine mapping of the muscle forces $\boldsymbol{F}$ to the Joint Space (JS), though the moment arm matrix $N(\boldsymbol{q}) = -L(\boldsymbol{q})^T$.

$$\mathcal{P}_\tau(\boldsymbol{q}) = \{\boldsymbol{\tau} \in \mathbb{R}^n \mid \boldsymbol{F} \in [\boldsymbol{F}_{min}, \boldsymbol{F}_{max}],\ \boldsymbol{\tau} = N(\boldsymbol{q})\boldsymbol{F}\} \tag{2.45}$$

The relationship of the achievable task space wrenches $\boldsymbol{f}$ and the joint torques $\boldsymbol{\tau}$ is described through the dynamics equation

$$\underbrace{M(\boldsymbol{q})\ddot{\boldsymbol{q}} + C(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q})}_{\boldsymbol{\tau}_b(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}})} + J^T(\boldsymbol{q})\boldsymbol{f} = \boldsymbol{\tau} \tag{2.46}$$

where $\boldsymbol{\tau}_b(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}})$ is a state dependant joint torque bias vector including the influence of the gravity $\boldsymbol{g}(\boldsymbol{q})$ and human state $\{\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}}\}$ and the desired acceleration $\ddot{\boldsymbol{q}}$.

Finally the achievable set of task space wrenches $\boldsymbol{f} \in \mathbb{R}^m$ can then be expressed as a set of all the achievable wrenches $\boldsymbol{f}$ given the polytope $\mathcal{P}_\tau$ shaped limits of joint torques $\boldsymbol{\tau}$

$$\mathcal{P}_f(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}}) = \left\{\boldsymbol{f} \in \mathbb{R}^m \mid \boldsymbol{\tau} \in \mathcal{P}_\tau(\boldsymbol{q}),\quad J^T(\boldsymbol{q})\boldsymbol{f} = \boldsymbol{\tau} - \boldsymbol{\tau}_b(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}})\right\} \tag{2.47}$$

Or in its implicit formulation

$$\mathcal{P}_f(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) = \left\{ \boldsymbol{f} \in \mathbb{R}^m \mid \boldsymbol{F} \in [\boldsymbol{F}_{min}, \boldsymbol{F}_{max}], \;\; J^T(\boldsymbol{q})\boldsymbol{f} = N(\boldsymbol{q})\boldsymbol{F} - \boldsymbol{\tau}_b(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) \right\} \tag{2.48}$$

The final polytope formulation $\mathcal{P}_f$ has an implicit form composed of two affine mappings. First the limits of the muscle tension forces $\boldsymbol{F} \in \mathbb{R}^d$ are projected into the limits of the joint torques $\boldsymbol{\tau} \in \mathbb{R}^n$, then the polytope of the joint torques $\mathcal{P}_\tau$ is transformed to the task space wrenches $\boldsymbol{f} \in \mathbb{R}^m$ using the Jacobian transpose matrix $J(\boldsymbol{q})^T$. As the musculoskeletal models often have large number of muscles d, the ending polytope $\mathcal{P}_f$ geometry is complex (having a large number of vertices and faces), both due to the number of muscles considered and the inherent complexity of the affine mapping. Therefore, this metric has been mostly used for static in-advance analysis of human postures [107], due to its long execution times.

However, Carmichael and Liu [61, 108] have proposed a method capable of improving the time efficiency of the polytope resolution that has a potential to be used even in real-time applications. This method is based on Ray Shooting Method (RSM) that samples the polytope in a set of predefined direction in task space. By choosing the directions of interest in advance, the method is capable of approximating the polytope, however it does not give any guarantees on the approximation accuracy.

The achievable wrench polytope $\mathcal{P}_f$ formulation (2.48) have been used for the analysis of the Multi-link Cable-Driven Robots (MCDRs) as well, for example in works by Sheng *et al.* [109] or Muralidharan *et al.* [110].

2.2.4 Acceleration polytope

The task space acceleration polytope $\mathcal{P}_a$ describes the relationship between the limitation of the muscle tensile forces $\boldsymbol{F} \in \mathbb{R}^d$ and the set of achievable task space accelerations $\ddot{\boldsymbol{x}} \in \mathbb{R}^m$.

As in the case of the force polytope, the simplified approach to this metric is possible, where the human joint torque limits $\boldsymbol{\tau} \in \mathbb{R}^n$ are specified as independent ranges, resulting in the acceleration polytope formulation identical to the one for robotic manipulators, as introduced in Section 2.1.6.

However, human's joint torques $\boldsymbol{\tau}$ are not generated by a set of independent actuators, but with a set of d contraction forces $\boldsymbol{F}$ produced by the muscle-tendon units $\boldsymbol{\tau} = N(\boldsymbol{q})\boldsymbol{F}$. Therefore, real limits of achievable torques of human joints are polytope $\mathcal{P}_\tau$ shaped, as described by equation (2.45).

The achievable acceleration polytope $\mathcal{P}_a$ formulation, respecting the muscle tension forces limits $\boldsymbol{F}$, can be derived from the human dynamics equation

$$M(\boldsymbol{q})\ddot{\boldsymbol{q}} + \underbrace{C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) + J^T(\boldsymbol{q})\boldsymbol{f}}_{\boldsymbol{\tau}_b(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{f})} = N(\boldsymbol{q})\boldsymbol{F} \tag{2.49}$$

where $N(\boldsymbol{q}) = -L(\boldsymbol{q})^T$ is the state dependant moment arm matrix. For any given human state $\{\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{f}\}$, the effects of the human's motion, the gravity as well as the applied external wrenches, can be grouped in a constant torque vector $\boldsymbol{\tau}_b(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{f})$. Then, the actuator acceleration $\ddot{\boldsymbol{q}}$ can be expressed was a function of the applied joint torque $\boldsymbol{\tau}$

$$\ddot{\boldsymbol{q}} = M(\boldsymbol{q})^{-1}N(\boldsymbol{q})\boldsymbol{F} - M(\boldsymbol{q})^{-1}\boldsymbol{\tau}_b(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{f}) \tag{2.50}$$

Finally, the joint accelerations $\ddot{\boldsymbol{q}}$ can be transformed to the task space using the mapping (2.33b)

$$\ddot{\boldsymbol{x}} = J(\boldsymbol{q})M(\boldsymbol{q})^{-1}N(\boldsymbol{q})\boldsymbol{F} - \underbrace{J(\boldsymbol{q})M(\boldsymbol{q})^{-1}\boldsymbol{\tau}_b(\boldsymbol{q},\dot{\boldsymbol{q}},\boldsymbol{f}) + \dot{J}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}}}_{\boldsymbol{a}_b(\boldsymbol{q},\dot{\boldsymbol{q}},\boldsymbol{f})} \tag{2.51}$$

where $\boldsymbol{a}_b(\boldsymbol{q},\dot{\boldsymbol{q}},\boldsymbol{f}) \in \mathbb{R}^m$ presents a constant bias vector for any human joint state $\{\boldsymbol{q},\dot{\boldsymbol{q}}\}$ and the generated external wrench $\boldsymbol{f}$. The polytope $\mathcal{P}_a$ of achievable task space accelerations $\ddot{\boldsymbol{x}}$ can then be expressed as

$$\begin{aligned}\mathcal{P}_a(\boldsymbol{q},\dot{\boldsymbol{q}},\boldsymbol{f}) = \{\ddot{\boldsymbol{x}} \in \mathbb{R}^m \mid \boldsymbol{F} \in [\boldsymbol{F}_{min},\boldsymbol{F}_{max}]\,,\\ \ddot{\boldsymbol{x}} = J(\boldsymbol{q})M(\boldsymbol{q})^{-1}N(\boldsymbol{q})\boldsymbol{F} - \boldsymbol{a}_b(\boldsymbol{q},\dot{\boldsymbol{q}},\boldsymbol{f})\}\end{aligned} \tag{2.52}$$

The polytope $\mathcal{P}_a$ is formulated as a linear and affine projection of the muscles tension force $\boldsymbol{F}$ limits using the state dependant projection matrix $J(\boldsymbol{q})M(\boldsymbol{q})^{-1}N(\boldsymbol{q})$. This polytope is much simpler to resolve than the force polytope $\mathcal{P}_f$ as it represents a projection of the d dimensional limits of muscle tension forces $\boldsymbol{F}$ to the lower m dimensional space of achievable task space accelerations $\ddot{\boldsymbol{x}}$.

Polytope $\mathcal{P}_a$ metrics have been used for musculoskeletal model analysis of highly dynamical movements such as football throwing [60] and golf swinging [62]. Additionally this metric has been used for the design analysis of the Multi-link Cable-Driven Robots (MCDRs) as well [109].

2.2.5 Velocity polytope

The simplest form of task space velocity capacity describes the relationship between the limited ranges (2.43) of human joint velocity $\dot{\boldsymbol{q}} \in \mathbb{R}^n$ and achievable task space velocities $\dot{\boldsymbol{x}} \in \mathbb{R}^m$, mapped through the Jacobian matrix $J(\boldsymbol{q}) \in \mathbb{R}^{m\times n}$.

$$\mathcal{P}_v(\boldsymbol{q}) = \{\dot{\boldsymbol{x}} \in \mathbb{R}^m \mid \dot{\boldsymbol{q}} \in [\dot{\boldsymbol{q}}_{min},\dot{\boldsymbol{q}}_{max}]\,,\quad \dot{\boldsymbol{x}} = J(\boldsymbol{q})\dot{\boldsymbol{q}}\} \tag{2.53}$$

This formulation is essentially the same as the one for robotic manipulators described in velocity Section 2.1.2 and all the considerations discussed in that chapter are valid for this formulation as well.

However, taking in account only human's joint velocity $\dot{\boldsymbol{q}}$ limits in a form of independent ranges (2.43) does not take in consideration the limits of muscle extension velocity $\dot{\boldsymbol{l}} \in \mathbb{R}^d$. The true limits of human joint velocities are configuration dependent, as they are a consequence of the limits of muscle extension velocities $\dot{\boldsymbol{l}}$. The relationship between the joint and muscle extension velocities is given through the muscle Jacobian matrix $L(\boldsymbol{q}) \in \mathbb{R}^{d\times n}$, forming polytope shaped limits

$$\mathcal{P}_{\dot{q}}(\boldsymbol{q}) = \left\{\dot{\boldsymbol{q}} \in \mathbb{R}^n \mid \dot{\boldsymbol{l}} \in \left[\dot{\boldsymbol{l}}_{min},\dot{\boldsymbol{l}}_{max}\right],\quad L(\boldsymbol{q})\dot{\boldsymbol{q}} = \dot{\boldsymbol{l}}\right\} \tag{2.54}$$

These polytope shaped limits can then be used to define the achievable task space velocity $\dot{\boldsymbol{x}} \in \mathbb{R}^m$ polytope which respects the limits of the muscle extension velocity $\dot{\boldsymbol{l}}$.

$$\mathcal{P}_v(\boldsymbol{q}) = \{\dot{\boldsymbol{x}} \in \mathbb{R}^m \mid \dot{\boldsymbol{q}} \in \mathcal{P}_{\dot{q}}(\boldsymbol{q}),\quad J(\boldsymbol{q})\dot{\boldsymbol{q}} = \dot{\boldsymbol{x}}\} \tag{2.55}$$

Or in its implicit form

$$\mathcal{P}_v(\boldsymbol{q}) = \left\{\dot{\boldsymbol{x}} \in \mathbb{R}^m \mid \dot{\boldsymbol{l}} \in \left[\dot{\boldsymbol{l}}_{min},\dot{\boldsymbol{l}}_{max}\right],\quad J(\boldsymbol{q})\dot{\boldsymbol{q}} = \dot{\boldsymbol{x}},\quad L(\boldsymbol{q})\dot{\boldsymbol{q}} = \dot{\boldsymbol{l}}\right\} \tag{2.56}$$

This metric has an implicit formulation making its resolution relatively complex, requiring using multiple standard methods for polytope evaluation. It first requires finding the limits of

joint velocities $\dot{\boldsymbol{q}} \in \mathbb{R}^n$ based on the limits of the muscle extension velocities $\dot{\boldsymbol{l}} \in \mathbb{R}^d$ and then projecting them to the task space velocities $\dot{\boldsymbol{x}} \in \mathbb{R}^m$.

To the best of our knowledge, polytope $\mathcal{P}_v$ in its full implicit form (2.56), has not yet been used with musculoskeletal models. However, this implicit formulation has been used for Multi-link Cable-Driven Robots (MCDRs) analysis [110].

Simplified achievable velocity based physical ability metrics, such as manipulability ellipsoids, considering the joint velocity $\dot{\boldsymbol{q}}$ limits [70, 111, 112], have been used in the human motion and ergonomics analysis. As these metrics neglect the muscle extension velocity $\dot{\boldsymbol{l}}$ limits, polytope $\mathcal{P}_v$ could be used as their more complete alternative.

2.2.6 Stiffness polytope

Stiffness metrics for human musculoskeletal models are used to evaluate the limitations of the endpoint displacement $\Delta x \in \mathbb{R}^m$ given the known muscle stiffness matrix $K_m \in \mathbb{R}^{d \times d}$ and a limited range of task space wrenches $\boldsymbol{f} \in \mathbb{R}^m$.

$$\boldsymbol{f} \in [\boldsymbol{f}_{min}, \boldsymbol{f}_{max}] \tag{2.57}$$

With a known muscle stiffness matrix K_m, that can be determined using the Hill's muscle model [113], the joint stiffness matrix $K_j \in \mathbb{R}^{n \times n}$ can be found using the muscle Jacobian matrix $L(\boldsymbol{q}) \in \mathbb{R}^{d \times n}$

$$K_j = L(\boldsymbol{q}) K_m L(\boldsymbol{q})^T \tag{2.58}$$

This Joint Space (JS) stiffness matrix K_j can be used to find the robot state dependent task space stiffness $K_c \in \mathbb{R}^{m \times m}$, through the Jacobian matrix $J(\boldsymbol{q}) \in \mathbb{R}^{m \times n}$ [96, 114, 115]

$$K_c(\boldsymbol{q})^{-1} = J(\boldsymbol{q}) K_j^{-1} J(\boldsymbol{q})^T \tag{2.59}$$

Given the range of expected external wrenches (2.57), the polytope of maximal task space displacements $\Delta\boldsymbol{x}$ can then be expressed as

$$\mathcal{P}_\Delta(\boldsymbol{q}) = \{\Delta\boldsymbol{x} \in \mathbb{R}^m \mid \boldsymbol{f} \in [\boldsymbol{f}_{min}, \boldsymbol{f}_{max}], \quad K_c(\boldsymbol{q})\Delta\boldsymbol{x} = \boldsymbol{f}\} \tag{2.60}$$

However, this formulation considers that the external wrench $\boldsymbol{f}$ range (2.57) is inside the human wrench capacity. In order to make sure that the human's wrench capacity is not exceeded when calculating the maximal task space displacement polytope $\mathcal{P}_\Delta$ the external wrench range (2.57) needs to be intersected with the human's wrench polytope $\mathcal{P}_f$ described in the Section 2.2.3, the polytope $\mathcal{P}_\Delta$ of task space task space displacement $\Delta\boldsymbol{x}$ can then be expressed as

$$\mathcal{P}_\Delta(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) = \{\Delta\boldsymbol{x} \in \mathbb{R}^m \mid \boldsymbol{f} \in [\boldsymbol{f}_{min}, \boldsymbol{f}_{max}] \cap \mathcal{P}_f(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}), \quad K_c(\boldsymbol{q})\Delta\boldsymbol{x} = \boldsymbol{f}\} \tag{2.61}$$

If the maximal displacement polytope $\mathcal{P}_\Delta$ is calculated only in relationship to the human's wrench capacity $\mathcal{P}_f$, without specifying the expected range of external wrenches (2.57), this polytope can be expressed in an implicit form

$$\mathcal{P}_\Delta(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) = \{\Delta\boldsymbol{x} \in \mathbb{R}^m \mid \boldsymbol{F} \in [\boldsymbol{F}_{min}, \boldsymbol{F}_{max}], \quad J^T(\boldsymbol{q}) K_c(\boldsymbol{q}) \Delta\boldsymbol{x} = N(\boldsymbol{q})\boldsymbol{F} - \boldsymbol{\tau}_b(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})\} \tag{2.62}$$

The stiffness polytope $\mathcal{P}_\Delta$ then represents the maximal task space displacement $\Delta\boldsymbol{x} \in \mathbb{R}^m$ that can be achieved given the human's muscle tensile forces $\boldsymbol{F} \in \mathbb{R}^d$ limits (2.35), this polytope formulation is also called *stiffness feasibility region* [97].

The formulation of the polytope (2.60) has relatively low complexity, defined as a projection of the limits of the external task space wrenches $\boldsymbol{f} \in \mathbb{R}^m$ to the same dimensional task space displacements $\Delta\boldsymbol{x} \in \mathbb{R}^m$. However, the implicit polytope formulation (2.62) is much more complex as the limits of muscle tension forces $\boldsymbol{F} \in \mathbb{R}^d$ first need to be projected to the limits of the joint torques $\boldsymbol{\tau} \in \mathbb{R}^n$ which in term determine the set of task space displacements $\Delta\boldsymbol{x} \in \mathbb{R}^m$. While the formulation (2.60) can be resolved with standard polytope resolution algorithms, the formulation (2.62) requires a combination of multiple methods, making it significantly more time consuming.

These polytope based metrics however, to the best of our knowledge, have yet to be used with human musculoskeletal models and Multi-link Cable-Driven Robots (MCDRs). However, their potential has already been shown for human posture analysis [115] and MCDR design [116] applications, in their simplified ellipsoid forms.

2.3 Polytope representation of collaboration abilities

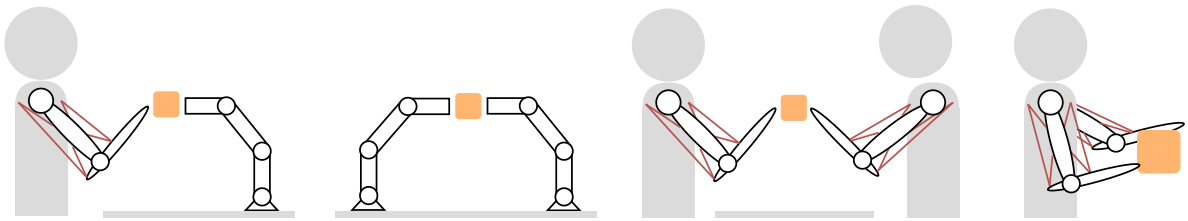

FIGURE 2.2: Illustrative examples of a common collaboration scenario composed of different actors (humans and robots), from left to right, human-robot, robot-robot, human-human and even two arms of a human subject can be seen as a collaboration of two musculoskeletal models.

Physical ability metrics are an important tool for analysing how robots and humans comply with different requirements of tasks. In human-robot collaboration scenarios, these metrics can be used is to decide if the robot, or the human, is better suited to execute a certain task, by evaluating their different physical abilities (movement, forces, precision etc.) to the ones required by the task [117].

However, when it comes to the physical collaboration, where the task is executed jointly by the human and the robot physically interacting, expressing their common physical abilities is much more challenging. As their common physical ability is a combination of their individual abilities, to be able to calculate their common physical ability, they need to be expressed in a unified form.

Many different physical abilities for humans and robots, can be represented in the polytope form, as described in Section 2.1 and Section 2.2. Expressing their physical abilities in this unifying form enables using different efficient tools from the polytope algebra to combine their individual polytopes, such as Minkowski sums, intersections and convex-hulls. Therefore, given their individual polytopes of different physical abilities and the physics of their physical interaction scenario, different polytope algebra operations can be leveraged to express their common physical ability in the polytope form as well.

One example of such characterisation has been developed by Lee [11], showing how polytopes can be used to describe the common velocity capacity of multi-arm collaborative robotic system, by intersecting their individual polytopes. However, this approach is yet to be used for characterising the common capacity of the human-robot interaction, as well as to be extended to other physical abilities and other collaboration scenarios.

Therefore, following sections describe the characterisation of different common physical abilities of the human-robot interaction, based on one classical example of human-robot collaboration scenario, where the two actors interact physically to manipulate an object that is rigidly fixed in the end-effector of the robot and in the hand of the human. First, Section 2.3.1 introduces the physical relationships describing this collaboration scenario, followed by Section 2.3.2 and Section 2.3.3, giving more specific examples of how these relationships are exploited to calculate the common force and velocity capacity of a physical human-robot collaboration leveraging the polytope algebra. Different versions of this scenario are shown on Figure 2.2, ranging from human-robot interaction to the interaction of the two arms of a human subject, that can be seen as two separate musculoskeletal models collaborating.

2.3.1 Human-robot collaboration scenario

A classical human-robot physical interaction scenario is considered, where the human operator and the robot are physically interacting in order to manipulate an object which is rigidly fixed both in the robot's end-effector and the human's hand. A simplified view of this collaboration scenario is shown on Figure 2.3, while a more general view of this scenario, with different actors (humans and robots), is shown on Figure 2.2.

When it comes to commonly manipulating an object held by both human and the robot, the supposition of rigid contact with the object can be used to formulate the force equilibrium equation and the relative motion constraints [118].

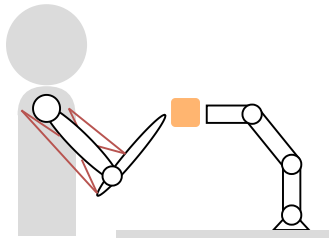

FIGURE 2.3: A collaborative example scenario where a human and a robot interact over an object fixed rigidly in the end-effector of the robot and in the hand of the human.

As they both apply forces on the object, the resulting force $\boldsymbol{f}$ the object will exert on the environment (if in contact with it) or transform into motion follows from the force equilibrium (sum of all forces $\boldsymbol{f}_i$ is equal to zero), and is equal to the sum of the applied forces by the robot $\boldsymbol{f}_r$ and the human $\boldsymbol{f}_h$.

$$\boldsymbol{f} = \boldsymbol{f}_r + \boldsymbol{f}_h, \qquad m_o\ddot{\boldsymbol{x}} = \boldsymbol{f} \tag{2.63}$$

If the object is not in contact with the environment, the force $\boldsymbol{f}$ will generate object's acceleration $\ddot{\boldsymbol{x}}$ in space proportional to its mass m_o.

Furthermore, as the object is rigidly attached to both robot and the human, there is no relative movement possible between the object, human's hand and robot's end-effector. According to this condition, the movement of the object (position $\boldsymbol{x}$, velocity $\dot{\boldsymbol{x}}$, acceleration $\ddot{\boldsymbol{x}}$, jerk $\dddot{\boldsymbol{x}}$, etc.) is equivalent to the movement of both the human's hand $\{\boldsymbol{x}_h, \dot{\boldsymbol{x}}_h, \ddot{\boldsymbol{x}}_h, \dddot{\boldsymbol{x}}_h\}$ and the robot's end-effector $\{\boldsymbol{x}_r, \dot{\boldsymbol{x}}_r, \ddot{\boldsymbol{x}}_r, \dddot{\boldsymbol{x}}_r\}$.

$$\boldsymbol{x}_h = \boldsymbol{x}_r = \boldsymbol{x}, \qquad \dot{\boldsymbol{x}}_h = \dot{\boldsymbol{x}}_r = \dot{\boldsymbol{x}}, \qquad \ddot{\boldsymbol{x}}_h = \ddot{\boldsymbol{x}}_r = \ddot{\boldsymbol{x}}, \qquad \dddot{\boldsymbol{x}}_h = \dddot{\boldsymbol{x}}_r = \dddot{\boldsymbol{x}} \tag{2.64}$$

These two relationships, considering the rigid contact between the robot, human and the object, can be used to describe different physical abilities of the human-robot collaboration in the described scenario. Section 2.3.2 describes exploiting this relationship to calculate the common force capacity while Section 2.3.3 described the calculation of their common velocity capacity.

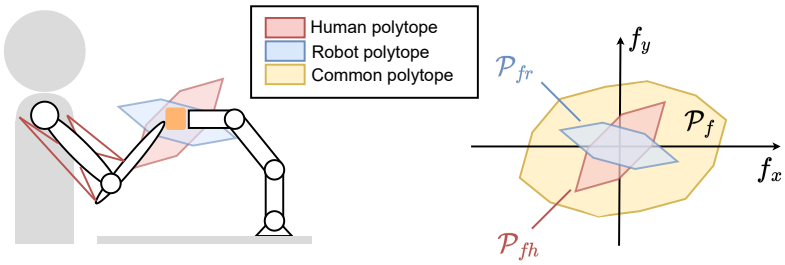

FIGURE 2.4: A collaborative example scenario where a human and a robot interact over an object fixed rigidly in the end-effector of the robot and in the hand of the human. The force polytope of the human (red) and the robot (blue) as well as their collaborative polytope (orange) calculated as a Minkowski sum of their individual polytopes are shown on the right.

2.3.2 Force polytope

When a human operator and a robot are physically interacting in order to manipulate an object, which is rigidly fixed both in the robot's end-effector and the human hand, the final force exerted on the object will be the sum of the forces of the human $\boldsymbol{f}_h$ and the force of the robot $\boldsymbol{f}_r$.

$$\boldsymbol{f} = \boldsymbol{f}_h + \boldsymbol{f}_r \tag{2.65}$$

Their individual force capacity can be expressed in a polytope form, for the human $\mathcal{P}_{fh}$ and the robot $\mathcal{P}_{fr}$ respectively. Finally, as their forces acting on the object are summed, their common force capacity on the object can be expressed as

$$\mathcal{P}_f = \{\boldsymbol{f} \in \mathbb{R}^m \mid \boldsymbol{f} = \boldsymbol{f}_r + \boldsymbol{f}_h, \quad \boldsymbol{f}_r \in \mathcal{P}_{fr},\ \boldsymbol{f}_h \in \mathcal{P}_{fh}\} \tag{2.66}$$

Therefore their common wrench capacity can be expressed as a sum of their individual wrench capacities, corresponding to the Minkowski sum $\oplus$ in polytope algebra

$$\mathcal{P}_f = \mathcal{P}_{fr} \oplus \mathcal{P}_{fh} \tag{2.67}$$

A planar example of this collaboration scenario and the polytopes obtained is in shown on Figure 2.4.

In more general case, if a collaboration is composed of N actors rigidly holding an object, where their force capacity is expressed in a polytope form $\mathcal{P}_{fi}$, then their common force capacity can be calculated as the Minkowski sum of the N polytopes.

$$\mathcal{P}_f = \mathcal{P}_{f1} \oplus \mathcal{P}_{f2} \oplus \ \ldots \ \oplus \mathcal{P}_{fN} \tag{2.68}$$

Furthermore, the polytope formulations that require calculating the wrench/force capacity, for both robot's and human's, will be combined using Minkowski sum $\oplus$. Such metrics are human's and robot's wrench/force polytope and stiffness region polytope, as shown in Table 2.1.

2.3.3 Velocity polytope

When a human operator and a robot are physically interacting in order to manipulate an object which is rigidly fixed both in the robot's end effector and the human hand, there is no relative movement of the object with respect to the human's hand and the robot's end effector

$$\dot{\boldsymbol{x}}_h = \dot{\boldsymbol{x}}_r = \dot{\boldsymbol{x}}, \tag{2.69}$$

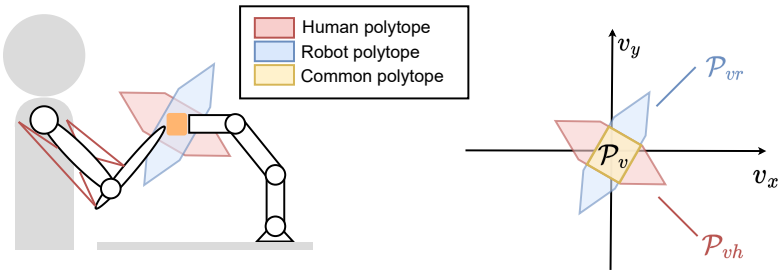

FIGURE 2.5: A collaborative example scenario where a human and a robot interact over an object fixed rigidly in the end-effector of the robot and in the hand of the human. The velocity polytope of the human (red) and the robot (blue) as well as their collaborative polytope (orange) calculated as an intersection of their individual polytopes are shown on the right.

Once their individual velocity capacity is expressed in polytope form $\mathcal{P}_{vh}$ and $\mathcal{P}_{vr}$, their common velocity capacity $\mathcal{P}_v$ (the achievable velocities of the object) can then be calculated as

$$\mathcal{P}_v = \{\dot{\boldsymbol{x}} \in \mathbb{R}^m \mid \dot{\boldsymbol{x}} \in \mathcal{P}_{vr},\ \dot{\boldsymbol{x}} \in \mathcal{P}_{vh}\} \tag{2.70}$$

Geometrically, this operation can be represented as an intersection of their velocity capacities, and the intersection $\cap$ operation can be calculated efficiently using polytope algebra.

$$\mathcal{P}_v = \mathcal{P}_{vh} \cap \mathcal{P}_{vr} \tag{2.71}$$

A planar example of this collaboration scenario and the polytopes obtained is in shown on Figure 2.5.

In a more general case where there are N actors (robots and humans) collaborating by rigidly holding the object, and where their velocity capacity is expressed in a polytope form $\mathcal{P}_{vi}$, the common velocity capacity can be calculated as their intersection

$$\mathcal{P}_v = \mathcal{P}_{v1} \cap \mathcal{P}_{v2} \cap \quad \dots \quad \cap \mathcal{P}_{vN} \tag{2.72}$$

Furthermore, the polytope formulations characterising robot's or human's movement capacity (velocity $\dot{\boldsymbol{x}}$, acceleration $\ddot{\boldsymbol{x}}$, etc.) will be subject to the same condition (2.64) and will be combined using the intersection $\cap$ operation, as shown in Table 2.1.

2.4 Conclusion and synthesis

This chapter brings an overview of the polytope based physical ability characterisations for humans and robots, with the aim to develop a unified view of their abilities and provide a base for characterising their common physical abilities. As described in Section 1.1, the literature proposes many different metrics for quantifying human and robot physical abilities. However, they are often very different in scope, accuracy, physical interpretation and in many cases specific to only robots or only humans. The focus of this chapter, as well as this thesis, is therefore placed on polytope based representations of physical abilities as they are arguably the most complete and the most accurate characterisations for both robots and humans, based on their musculoskeletal models.

This chapter brings an overview of the common polytope based representations of humans' and robots' physical abilities in Section 2.1 and Section 2.2, in the attempt to present a systematic view on their formulations. In this overview, different physical abilities of humans (based on

musculoskeletal models) and robots are characterised by finding the relationships between their actuation limits and the achievable sets of different task-related variables. As these relationships are usually described through their highly nonlinear dynamics and kinematics relationships, in order to obtain the polytope-shaped representations, the dynamics and kinematics equations are linearised around a specific state. This approach yields linear and affine transformations of the actuator limits to the achievable sets of desired task space variables.

Common polytope formulations, described in Section 2.1 for robots and in Section 2.2 for humans, involve different linearised dynamics and kinematics equations, as well as different actuation limits. However, they can be represented using a generic formulation

$$\mathcal{P}_x = \{\boldsymbol{x} \in \mathbb{R}^m \mid A\boldsymbol{x} = B\boldsymbol{y} + \boldsymbol{b},\ \boldsymbol{y} \in \mathcal{P}_y\} \tag{2.73}$$

where the polytope $\mathcal{P}_x$ represents the achievable set of the task space variable $\boldsymbol{x} \in \mathbb{R}^m$, $\boldsymbol{y} \in \mathbb{R}^k$ is the input (actuator) variable limited within input set (actuator limits) $\mathcal{P}_y$. The input set $\mathcal{P}_y$ is often defined in the form of intervals of different input (actuator) variables, however in general case it can have a polytope shape as well. The linearised dynamics and kinematics equations can be represented, in a generic case, using the equation $A\boldsymbol{x} = B\boldsymbol{y} + \boldsymbol{b}$, where the matrix $A \in \mathbb{R}^{k\times m}$ is a linear transformation matrix from the m dimensional output (task) space to the k dimensional intermediate space, the matrix $B \in \mathbb{R}^{k\times d}$ is a transformation matrix from the n dimensional input space to the intermediate space and $\boldsymbol{b} \in \mathbb{R}^k$ is a bias vector expressed in the intermediate space. Furthermore, the dimension of output (task) space m is lower than the intermediate space dimension $k \geq m$, which is in term lower than the input (actuator) space dimension $n \geq k \geq m$. This condition comes from the fact that when characterising the physical abilities, the polytopes map higher dimensional input space (ex. Joint Space (JS) or muscle force space) to the lower dimensional output space (ex. Cartesian Space (CS)).

Table 2.1 brings a list of the polytope formulations for robots and humans, introduced in Section 2.1 and Section 2.2 respectively, in a condensed view. The table shows how different polytope formulations correspond with the generic polytope formulation (2.73), in terms of system matrices A, B, input, output and bias vectors $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{b}$ and the input limits $\mathcal{P}_y$.

Polytope capacity	Eqn.	$\boldsymbol{x}$	A	B	$\boldsymbol{y}$	Input set $\mathcal{P}_y$	$\boldsymbol{b}$	Cond.	Collab.
Robotic manipulators									
Velocity	2.8	$\dot{\boldsymbol{x}}$	$I_{m\times m}$	J	$\dot{\boldsymbol{q}}$	$[\dot{\boldsymbol{q}}_{min}, \dot{\boldsymbol{q}}_{max}]$	-	Kin	$\cap$
Kin. Acceleration	2.9	$\ddot{\boldsymbol{x}}$	$I_{m\times m}$	J	$\ddot{\boldsymbol{q}}$	$[\ddot{\boldsymbol{q}}_{min}, \ddot{\boldsymbol{q}}_{max}]$	$\boldsymbol{a}_b$	Kin	$\cap$
Kin. Jerk	2.11	$\dddot{\boldsymbol{x}}$	$I_{m\times m}$	J	$\dddot{\boldsymbol{q}}$	$[\dddot{\boldsymbol{q}}_{min}, \dddot{\boldsymbol{q}}_{max}]$	$\boldsymbol{j}_b$	Kin	$\cap$
Precision	2.15	$\delta\dot{\boldsymbol{x}}$	$I_{m\times m}$	J	$\delta\boldsymbol{q}$	$[\delta\boldsymbol{q}_{min}, \delta\boldsymbol{q}_{max}]$	-	Kin	$\cap$
Force/Wrench	2.17	$\boldsymbol{f}$	J^T	$I_{n\times n}$	$\boldsymbol{\tau}$	$[\boldsymbol{\tau}_{min}, \boldsymbol{\tau}_{max}]$	$-\boldsymbol{\tau}_b$	Dyn	$\oplus$
Acceleration	2.22	$\ddot{\boldsymbol{x}}$	$I_{m\times m}$	JM^{-1}	$\boldsymbol{\tau}$	$[\boldsymbol{\tau}_{min}, \boldsymbol{\tau}_{max}]$	$-\boldsymbol{a}_b$	Dyn	$\cap$
Stiffness	2.30	$\Delta\boldsymbol{x}$	$J^T K_c$	$I_{n\times n}$	$\boldsymbol{\tau}$	$[\boldsymbol{\tau}_{min}, \boldsymbol{\tau}_{max}]$	$-\boldsymbol{\tau}_b$	Dyn	$\oplus$
Human musculoskeletal models									
Velocity	2.56	$\dot{\boldsymbol{x}}$	$I_{m\times m}$	J	$\dot{\boldsymbol{q}}$	$\mathcal{P}\dot{\boldsymbol{q}}$	-	Kin	$\cap$
Force/Wrench	2.48	$\boldsymbol{f}$	J^T	N	$\boldsymbol{F}$	$[\boldsymbol{F}_{min}, \boldsymbol{F}_{max}]$	$-\boldsymbol{\tau}_b$	Dyn	$\oplus$
Acceleration	2.52	$\ddot{\boldsymbol{x}}$	$I_{m\times m}$	$JM^{-1}N$	$\boldsymbol{F}$	$[\boldsymbol{F}_{min}, \boldsymbol{F}_{max}]$	$-\boldsymbol{a}_b$	Dyn	$\cap$
Stiffness	2.62	$\Delta\boldsymbol{x}$	$J^T K_c$	N	$\boldsymbol{F}$	$[\boldsymbol{F}_{min}, \boldsymbol{F}_{max}]$	$-\boldsymbol{\tau}_b$	Dyn	$\oplus$

TABLE 2.1: The list of common polytope formulations for characterising physical abilities of robots and humans introduced in Section 2.1 and Section 2.2 respectively. The table shows the correspondence of the common formulations to the generic formulation described by the equation (2.73). The table further specifies if the polytope formulations is specified in kinematic or dynamic conditions. Finally, the table shows the necessary polytope algebra operator in order to characterise the common abilities in the collaboration scenario described in Section 2.3.1.

Polytopes, apart from being able to accurately represent both human's and robot's physical abilities, enable combining their individual polytopes for characterising the common physical abilities when interacting physically to execute different task. Given the physical ability of interest and the physics of the human-robot physical interaction scenario, different polytope algebra tools, such as Minkowski sum or intersection, can be used to express the common physical ability of their interaction in the polytope form as well.

Section 2.3, demonstrates the use of different polytope algebra operations to characterise force and movement capacity of the human-robot collaboration in one classical collaboration scenario, where the human and the robot jointly manipulate an object. In the context of the same scenario, Table 2.1 shows the polytope algebra operators (Minkowski sum $\oplus$ and intersection $\cap$) used to combine their individual polytopes to characterise their different physical abilities as one system.

In summary, polytopes enable accurate representation of human's and robot's physical abilities, as well as characterising their common physical abilities when collaborating physically to execute a task. In that way, this representation provides tools to create more advanced collaboration scenarios. Accurate representation of robot's physical abilities enables exploiting its abilities fully when designing its tasks, creating the adapted robot control strategies or planning for its trajectories. On the other hand, accurate representation of human's abilities enables assessing the task ergonomics and making sure that human's capacity is never surpassed. Additionally, it enables creating more human-centered robot behaviours, where the robot adapts its assistance level to the lacking physical abilities of the human operator. Finally, a unified representation of their individual and common physical abilities might enable new task scheduling techniques capable of assessing if different tasks are more suitable for robots, human operators or are they more suitable for their collaboration.

However, polytope formulations, as introduced in Section 2.1 and Section 2.2, need to be transformed to more standard representations in order to be used in practical implementations. The two most common ways to represent a polytope are as a set of vertices, or as a set of inequality constraints corresponding to its faces. Depending on the polytope formulation structure different transformation strategies, to the one of these forms, need to be employed, where the computational efficiency of the transformation operation can vary significantly with respect to the formulation complexity. Therefore, the following chapter (Chapter 3) focuses on providing a generic view on different formulation families present when characterising physical abilities of humans and robots, with the aim to specify different polytope transformation strategies applicable to each of the families as well as to discuss their computational complexity.

Chapter 3

Transforming polytopes to standard representations

As described in Chapter 2, polytope algebra offers efficient means of representing and manipulating the physical abilities of humans and robots. However, using polytopes in practical applications requires transforming them to standard representations, for example into a set of vertices or a set of inequality constraints, corresponding to their faces. The algorithms used to transform the polytopes to these representations, known as *vertex enumeration* and *facet enumeration* methods, have been extensively studied in the literature [71]. However, their applicability often depends on specific polytope formulations and their efficiency is influenced by the complexity of the polytope itself (e.g., the number of vertices and faces).

In the previous chapter, Section 2.4 introduced a generic polytope formulation unifying all the common polytope representations of human and robot physical abilities. However, this formulation is not directly suitable for the standard polytope transformation strategies. Therefore, in this chapter, Section 3.2 proposes a structured view on the different families of polytope formulations derived from the unified formulation, namely the intersection and the projection formulations, as well as their special cases.

Section 3.3 then provides an overview of different polytope transformation strategies in the literature, for the introduced families of polytope formulations. The section starts with the overview of generic strategies for converting between standard (vertex and half-plane) representation, followed by an overview of specialised methods for different families of introduced polytope formulations. In addition to providing an overview of the applicable methods, the section discusses the efficiency of the proposed methods and their limitations. Moreover, Section 3.3 discusses the use of different polytope approximation strategy for improving the transformation efficiency for high-dimensional polytopes that are intractable with standard methods. Finally, Section 3.3.5 brings a condensed view of the proposed overview in a form of Table 3.2.

The final two section of this chapter introduce two theoretic and algorithmic contributions of this work regarding the efficient transformation of polytopes. Section 3.4 introduces an efficient vertex enumeration algorithm for a specific polytope formulation called the intersection formulation with interval input set, introduced in Section 3.2.2. The algorithm exploits the hyperrectangle geometry of the input set which significantly reduces the computational complexity of the algorithm and lowers execution time. The algorithm's performance is compared against the state-of-the-art methods on the use case of the robot's wrench capacity polytope, described in Section 2.1.5.

Section 3.5 introduces a new polytope approximation algorithm developed directly for the unified polytope formulation, introduced in the previous chapter. Therefore, this polytope transformation strategy is suitable for all common polytope based physical ability characterisation

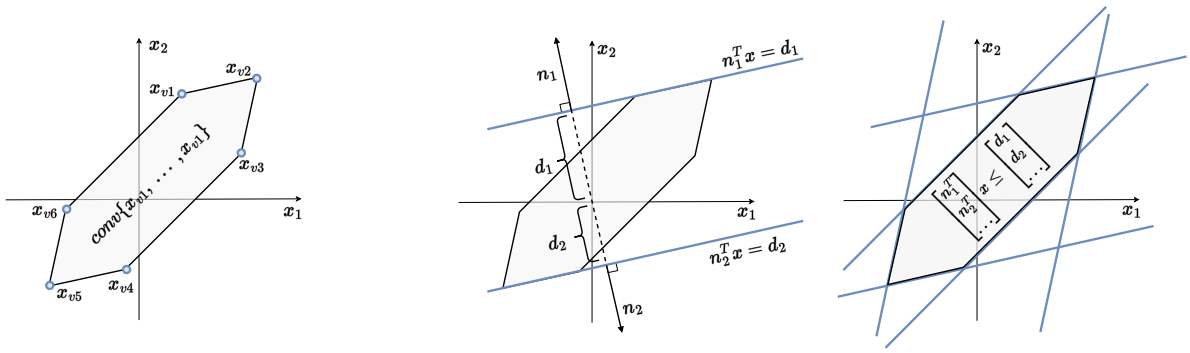

FIGURE 3.1: Figure illustrating the vertex ($\mathcal{V}$) and half-plane ($\mathcal{H}$) representations of a polytope on a simple planar example. Image on the left shows the $\mathcal{V}$-representation as a Convex-Hull of the specified vertices. Images on the right show the $\mathcal{H}$-representation as an intersection of the half-planes (lines in 2D) representing the faces of the polytope $\mathcal{P}_x$.

introduced in the previous chapter. However, it is particularly well suited for high-dimensional problems, where the standard exact methods often have intractable execution times. Such high dimensional problems are common when it comes to characterising human's physical abilities based on musculoskeleral models. The performance of this algorithm is tested on the challenging use case of the human's wrench capacity polytope, and compared to the state-of-the-art methods.

3.1 Common polytope representations

When it comes to polytopes $\mathcal{P}_x$ characterising the convex sets of variables $\boldsymbol{x} \in \mathbb{R}^m$, the most commonly used representations in literature are the so-called vertex or $\mathcal{V}$ and half-plane or $\mathcal{H}$-representation [71, 119].

The vertex or $\mathcal{V}$-representation consists in specifying a list of polytope's vertices $\boldsymbol{x}_{vi} \in \mathbb{R}^m$, such that the polytope is defined as their Convex-Hull Conv $(\cdot)$

$$\mathcal{P}_x = \text{Conv}\left(\boldsymbol{x}_{v1},\ \boldsymbol{x}_{v2},\ \ldots,\ \boldsymbol{x}_{vN_v}\right) \tag{3.1}$$

where N_v is the number of vertices. Figure 3.1 (left) shows a graphical interpretation of the vertex representation on the example of planar ($m=2$) polytope.

The half-plane or $\mathcal{H}$-representation, on the other hand, is defined as the intersection of the half-planes forming the faces of the polytope

$$\mathcal{P}_x = \{\boldsymbol{x} \in \mathbb{R}^m \mid H\boldsymbol{x} \leq \boldsymbol{d}\} \tag{3.2}$$

where the matrix $H \in \mathbb{R}^{N_f \times m}$ is composed of N_f normal vectors $\boldsymbol{n}_i \in \mathbb{R}^m$ of the half-planes corresponding to the N_f faces of the polytope, while vector $\boldsymbol{d} \in \mathbb{R}^{N_f}$ contains their displacement d_i from the origin. Figure 3.1 (right) shows a graphical interpretation of the half-plane representation on the example of planar ($m=2$) polytope.

Several alternative polytope representations were proposed recently, such as the $\mathcal{Z}$-representation [120] where the polytopes are represented as polynomial zonotopes, and the $\mathcal{M}$-representation [121] which is a more compact special case of the $\mathcal{Z}$-representation. These representations offer an improved efficiency of conducting polytope algebra operations, with respect to the traditional $\mathcal{H}$ and $\mathcal{V}$-representations. However, despite these advancements, their practical applications are still quite limited, primarily due to the challenging computational complexity

involved in transforming polytopes into these alternative representations.

Representing polytopes in these standard forms enables using different efficient tools from computational geometry and polytope algebra to perform operations over polytopes and provides tools for exploiting them in practical applications. However, in many cases polytope formulations do not correspond neither to the $\mathcal{V}$ nor $\mathcal{H}$-representation. As a result, many algorithms are developed in the literature transforming different polytope formulations into one of the two standard forms. The algorithms transforming the polytopes to the $\mathcal{V}$-representation are often called *vertex enumeration* algorithms and to the $\mathcal{H}$-representation of are called *facet enumeration* algorithms [122]. The appropriate choice of suitable polytope transformation algorithm, and its complexity, depends on various factors such as the polytope formulation, the desired polytope representation ($\mathcal{V}$ or $\mathcal{H}$), and potentially the time constraints for algorithm execution (online or offline).

Therefore, Section 3.2 brings a structured overview of different polytopes, especially the ones occurring when characterising different physical abilities of humans and robots, and classifies them with respect to their formulation. Section 3.3 then exploits this classification to provide an overview of their applicable polytope transformation strategies and briefly discusses their efficiency.

3.2 Generic view on physical ability polytope formulations

This section aims to present a generic perspective on different families of polytope formulations commonly found when characterising the physical abilities of humans and robots. These polytopes are categorised based on their formulation to facilitate the choice of applicable algorithms.

From a generic standpoint, physical abilities represented in the shape of polytopes $\mathcal{P}_x$ can be seen as feasible sets of output (task space) variables $\boldsymbol{x} \in \mathbb{R}^m$, produced by applying a linear transformation on the input (actuation space) variables $\boldsymbol{y} \in \mathbb{R}^n$, bounded within (actuator limits) $\boldsymbol{y} \in \mathcal{P}_y$. As described in the previous chapter (Section 2.4), the polytope formulations for characterising physical abilities can be expressed in a unified generic form

$$\mathcal{P}_x = \{\boldsymbol{x} \in \mathbb{R}^m \mid A\boldsymbol{x} = B\boldsymbol{y} + \boldsymbol{b},\ \boldsymbol{y} \in \mathcal{P}_y\} \tag{3.3}$$

where matrices $B \in \mathbb{R}^{k\times n}$ and $A \in \mathbb{R}^{k\times m}$ present the linear transformation of the n dimensional input to m dimensional output space, through the k dimensional intermediate space, while $\boldsymbol{b} \in \mathbb{R}^k$ is a bias vector. Furthermore, in the context of this work, the dimension of output space m is considered to be lower or equal to the intermediate space dimension $k \geq m$, which is in term lower or equal to the input space dimension $n \geq k \geq m$. In practice, this condition implies that the human or the robot model considered have at least the same number of degrees of freedom (DOF) as the dimension of the output space m.

This compact and implicit formulation represents a unified view of different physical ability polytope formulations. However, depending on the structure of the matrices A and B, as well as on the form of the input set $\mathcal{P}_y$, different algorithms are required to transform this formulation to its standard representations ($\mathcal{H}$ and $\mathcal{V}$).

One important factor for determining the complexity of the polytope transformation, and the choice of the suitable algorithm, is the shape of the input set, corresponding to the limits of the input variable $\boldsymbol{y} \in \mathbb{R}^n$ limits. There are two most common forms that can be considered: interval form $\mathcal{I}_y$ and polytope form $\mathcal{P}_y$.

The interval shaped $\mathcal{I}_y$ input set is specified in a form of min-max intervals

$$\mathcal{I}_y = \{\boldsymbol{y} \in \mathbb{R}^n | \ \ \boldsymbol{y}_{min} \leq \boldsymbol{y} \leq \boldsymbol{y}_{max}\} \tag{3.4}$$

In a geometric sense, these individual intervals of the input variable $\boldsymbol{y}$ can be visualised as an n-dimensional hyperrectangle, also known as a hyperbox or an orthotope. Each axis i of the hyperrectangle corresponds to an interval $y_i \in [y_{i,\min}, y_{i,\max}]$.

More generally, rather than an interval $\mathcal{I}_y$, the input set can be defined as any convex polytope (ex. set of linear constraints in its $\mathcal{H}$-representation).

$$\mathcal{P}_y = \{\boldsymbol{y} \in \mathbb{R}^n | \ \ H_y\boldsymbol{y} \leq \boldsymbol{d}_y\} \tag{3.5}$$

As polytopes $\mathcal{P}_x$ are then defined as linear transformations $A\boldsymbol{x} = B\boldsymbol{y} + \boldsymbol{b}$ of the input set (intervals $\mathcal{I}_y$ or polytope $\mathcal{P}_y$ shaped) from the input space to the output space, the structure of the linear transformation (matrices A and B) has a direct impact on the complexity of the formulation, as well as on the choice of the algorithm. Therefore, in this work, the distinction between two generic families of polytope formulations is proposed, derived from the unified formulation (3.3): the projection formulation, where matrix A is identity matrix $A = I_{m\times m}$, and the intersection formulation, where $B = I_{n\times n}$.

3.2.1 Projection formulation

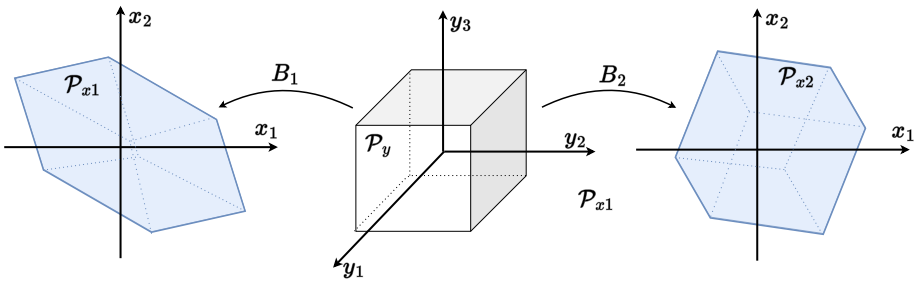

FIGURE 3.2: An example ($m = 2$, $n = 3$) of constructing two different projection formulation polytopes, from the same input set $\mathcal{P}_y$. Polytopes $\mathcal{P}_{x1}$ and $\mathcal{P}_{x2}$ represent linear affine transformations of the input set $\mathcal{P}_y$ using two different transformation matrices B_1 and B_2.

The projection formulation is defined through a linear affine transformation of the n dimensional input space $\boldsymbol{y} \in \mathbb{R}^n$ to the m dimensional output space $\boldsymbol{x} \in \mathbb{R}^m$ using the projection matrix $B \in \mathbb{R}^{m\times n}$

$$\mathcal{P}_x = \{\boldsymbol{x} \in \mathbb{R}^m \mid \boldsymbol{x} = B\boldsymbol{y} + \boldsymbol{b}_x, \ \boldsymbol{y} \in \mathcal{P}_y\} \tag{3.6}$$

where $\boldsymbol{b}_x \in \mathbb{R}^m$ is a constant bias vector, defined in the output space. The name *projection* formulation comes from the fact that the polytope $\mathcal{P}_x$, in this formulation, is a projection of the n dimensional input set $\mathcal{P}_y$ to the m dimensional output space $\mathbb{R}^m$ [123].

A graphical representation of the projection formulation (3.6) is shown on Figure 3.3, where two different output polytopes $\mathcal{P}_x$ are constructed applying different affine transformation matrices B on the same input set $\mathcal{P}_y$.

Zonotope formulation

If the input space has the interval shape $\mathcal{I}_y$, the final polytope $\mathcal{P}_x$, defined by the equation (3.6), has a zonotope form. In that case the polytope $\mathcal{P}_x$ can be represented as a Minkowski sum of n line segments $\mathcal{L}_i$, each one corresponding to one axis range $[y_{i,min}, y_{i,max}]$ of the input

set $\mathcal{I}_y$ projected to the output space using the matrix B. [124]

$$\mathcal{P}_x = \mathcal{L}_1 \oplus \cdots \oplus \mathcal{L}_n \tag{3.7}$$

where the i-th line segment $\mathcal{L}_i$ can be expressed through i-th line vector $\boldsymbol{b}_i$ of the matrix B

$$\mathcal{L}_i = \{\boldsymbol{x} \in \mathbb{R}^m \mid \boldsymbol{x} = \boldsymbol{b}_i y_i + \boldsymbol{b}_x, \quad y_i \in [y_{i,min}, y_{i,max}]\} \tag{3.8}$$

These line segments are often called *components* or *generators* of the zonotope $\mathcal{P}_x$ [125].

More generally, for any zonotope shaped input set $\mathcal{P}_y$, expressed as a Minkowski sum of the generators $\mathcal{L}_{yi}$

$$\mathcal{P}_y = \mathcal{L}_{y1} \oplus \mathcal{L}_{y2} \oplus \cdots \tag{3.9}$$

polytope $\mathcal{P}_x$ is a zonotope as well. The generators $\mathcal{L}_i$ of the polytope $\mathcal{P}_x$ correspond to the generators $\mathcal{L}_{yi}$ of the input set $\mathcal{P}_y$, projected to the output space using the equation (3.6). Therefore, polytope $\mathcal{P}_x$ has the same number of generators $\mathcal{L}_i$ as the zonotope $\mathcal{P}_y$, which can be expressed as

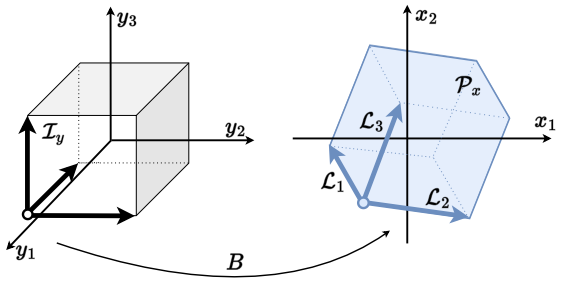

FIGURE 3.3: A graphical example ($m = 2$, $n = 3$) of constructing the zonotope shaped polytope $\mathcal{P}_x$ from the interval input set $\mathcal{I}_y$.

$$\mathcal{L}_i = \{\boldsymbol{x} \in \mathbb{R}^m \mid \boldsymbol{x} = B\boldsymbol{y} + \boldsymbol{b}_x, \quad \boldsymbol{y} \in \mathcal{L}_{yi}\} \tag{3.10}$$

Zonotopes are highly structured and central symmetric shapes, hence transforming them to appropriate representations ($\mathcal{H}$ and $\mathcal{V}$) is in many cases more efficient than for generic polytopes.

3.2.2 Intersection formulation

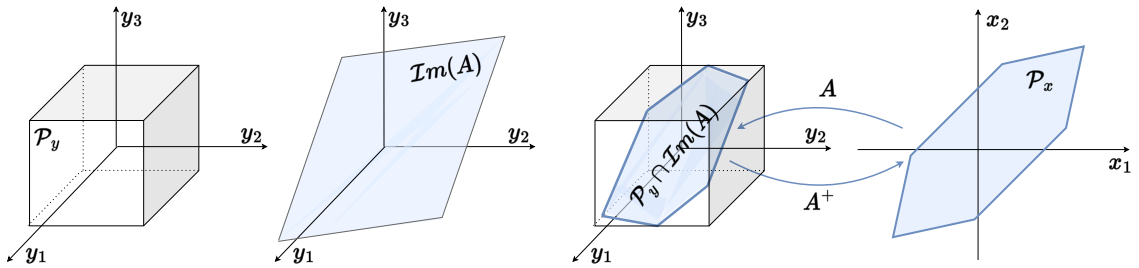

FIGURE 3.4: An example ($m = 2$, $n = 3$) of constructing a intersection formulation polytope $\mathcal{P}_x$ from the input set $\mathcal{P}_y$. First the image $\mathcal{I}m\,(A)$ of the matrix A is intersected with the input set $\mathcal{P}_y$, then this intersection is projected to the output space using the pseudo-inverse A^+.

The intersection formulation, on the other hand, is expressed by a matrix equation $A\boldsymbol{x} = \boldsymbol{y}$. It can be interpreted as a linear transformation in the opposite direction [126], from the m dimensional input space $\boldsymbol{x} \in \mathbb{R}^m$ to the n dimensional output space $\boldsymbol{y} \in \mathbb{R}^n$, using the projection matrix $A \in \mathbb{R}^{n \times m}$

$$\mathcal{P}_x = \{\boldsymbol{x} \in \mathbb{R}^m \mid A\boldsymbol{x} = \boldsymbol{y} + \boldsymbol{b}_y, \ \boldsymbol{y} \in \mathcal{P}_y\} \tag{3.11}$$

where $\boldsymbol{b}_y \in \mathbb{R}^n$ is a constant bias vector, defined in the input space. The name *intersection* formulation comes from the fact that the polytope $\mathcal{P}_x$ is no longer just a projection of the input

set $\mathcal{P}_y$ to the m dimensional output space, but rather a projection of its intersection with the image $\mathcal{I}m\,(A)$ of matrix A. A graphical representation of the intersection formulation (3.11) is shown on Figure 3.4, where the intersection $\mathcal{P}_y \cap \mathcal{I}m\,(A)$ and the final polytope $\mathcal{P}_x$ are denoted in blue.

The main difficulty of the intersection formulation (3.11) is that it is defined as an inverse projection, from the m dimensional output space to the n dimensional input space [126]. In order to inverse this relationship, an equivalent projection formulation polytope can be constructed by characterising the intersection $\mathcal{P}_y \cap \mathcal{I}m\,(A)$.

Equivalent projection formulation

In order to express the intersection formulation in the projection form, it is necessary to inverse the relationship $\boldsymbol{y} = A\boldsymbol{x}$, considering $\boldsymbol{b}_y = \boldsymbol{0}$. If the input and output space have the same dimension $n = m$, A is invertible and the inverse relationship can be easily obtained as $\boldsymbol{x} = A^{-1}\boldsymbol{y}$.

However, in the more general case, the input space is higher dimensional $n > m$, making matrix A not invertible. In such cases the inverse relationship can be obtained using the left pseudo-inverse of matrix A

$$\boldsymbol{x} = (A^T A)^{-1} A^T \boldsymbol{y} = A^+ \boldsymbol{y} \tag{3.12}$$

Due to the fact that, in the general case, the operation $\boldsymbol{x} = A^+\boldsymbol{y}$ is not bijective, the input vector $\boldsymbol{y}' = A\boldsymbol{x}$ corresponding to the output vector $\boldsymbol{x} = A^+\boldsymbol{y}$ does not correspond to the original input vector $\boldsymbol{y}$

$$\boldsymbol{y}' = A\boldsymbol{x} = A(A^+\boldsymbol{y}) \neq \boldsymbol{y} \tag{3.13}$$

making it possible that the input vector $\boldsymbol{y}' = AA^+\boldsymbol{y}$ no longer respects the input set $\mathcal{P}_y$

$$\boldsymbol{y}' = AA^+\boldsymbol{y} \notin \mathcal{P}_y \tag{3.14}$$

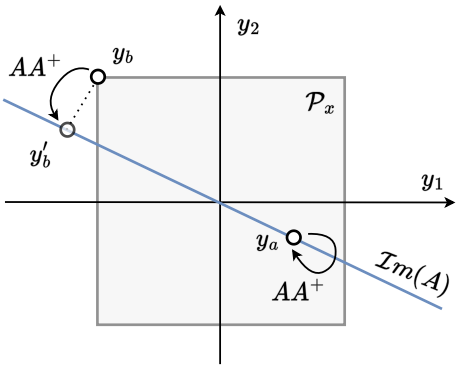

FIGURE 3.5: An example ($n = 2, m = 1$) of applying the image projector AA^+ in the input space, the input set $\mathcal{P}_y$ is shown in grey and the image $\mathcal{I}m\,(A)$ in blue. As the input vector $\boldsymbol{y}_a$ belongs to the image $\mathcal{I}m\,(A)$ its multiplication with AA^+ results in the same vector. $\boldsymbol{y}_b$ does not belong to the image and AA^+ results in $\boldsymbol{y}_b'$, its orthogonal projection onto the image $\mathcal{I}m\,(A)$ is outside the input set $\mathcal{P}_y$.

The matrix AA^+ represents an orthogonal projector matrix to the *image space* $\mathcal{I}m\,(A)$ of the matrix A [127, Chapter 5.5.4]. Therefore, for any input vector $\boldsymbol{y}$ belonging to the image $\mathcal{I}m\,(A)$, the multiplication by AA^+ results in itself [128, Chapeter 1.3.1]

$$\boldsymbol{y} = AA^+\boldsymbol{y}, \qquad \forall \boldsymbol{y} \in \mathcal{I}m\,(A) \tag{3.15}$$

However, for any input vector $\boldsymbol{y}$ not belonging to the image $\mathcal{I}m\,(A)$, multiplying it with the matrix AA^+ will result in its orthogonal projection to the image $\mathcal{I}m\,(A)$. As shown in the graphical example on Figure 3.5, even though the original $\boldsymbol{y}$ respects the input set $\mathcal{P}_y$, its orthogonal projection onto the image $AA^+\boldsymbol{y}$ does not is some cases.

Therefore, having ensured that all the inputs $\boldsymbol{y}$ belong to the image $\mathcal{I}m\,(A)$, the pseudo-inverse A^+ provides a unique inverse solution to the equation $\boldsymbol{y} = A\boldsymbol{x}$. Then, the new input set, respecting the limitations $\boldsymbol{y} \in \mathcal{P}_y$ and ensuring the unique inverse solution, can be found

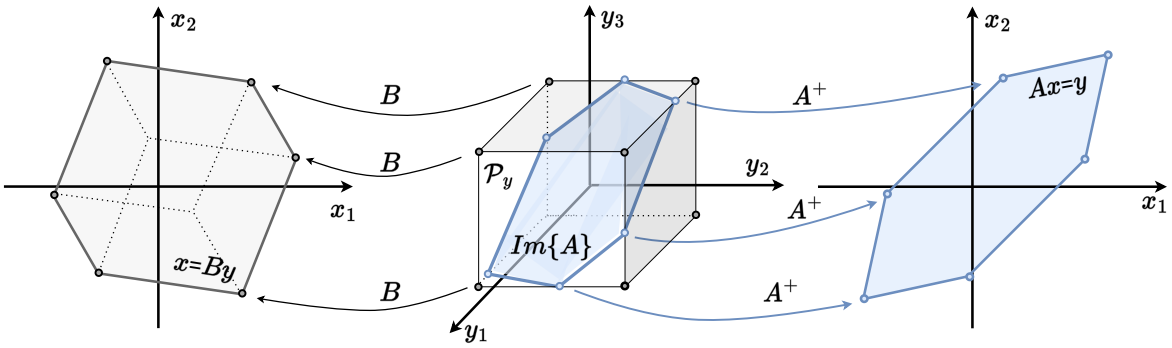

FIGURE 3.6: A comparative view of the projection (left) and intersection (right) polytope formulation using the same input space (middle). In the projection formulation, the whole input space $\mathcal{P}_y$ is projected using the matrix B to the output space to obtain the polytope $\mathcal{P}_x$ (grey left). In the intersection formulation, first the intersection of the input space $\mathcal{P}_y$ with the image of the matrix A is found (blue middle) $\mathcal{I}m\,(A) \cap \mathcal{P}_y$, then this intersection can be projected to the output space using the pseudo-inverse A^+ of the matrix A to obtain the polytope $\mathcal{P}_x$ (blue right).

as the intersection

$$\boldsymbol{y} \in \mathcal{I}m\,(A) \cap \mathcal{P}_y \tag{3.16}$$

By exploiting the new input set and using the pseudo-inverse A^+, the equivalent formulation of the the initial intersection polytope $\mathcal{P}_x$ can be expressed as

$$\mathcal{P}_x = \{\boldsymbol{x} \in \mathbb{R}^m \mid \boldsymbol{x} = A^+\boldsymbol{y} + A^+\boldsymbol{b}_y,\ \boldsymbol{y} \in \mathcal{I}m\,(A) \cap \mathcal{P}_y\} \tag{3.17}$$

Using the formulation (3.17), in practice, often requires characterising the image $\boldsymbol{y} \in \mathcal{I}m\,(A)$ in a set form as well. As shown in expression (3.15), all the vectors $\boldsymbol{y}$ belonging to the image $\mathcal{I}m\,(A)$, when multiplied with the image projector AA^+ result in themselves ($\boldsymbol{y} - AA^+\boldsymbol{y} = \boldsymbol{0}$). This relationship can be exploited to formulate a compact set form of the image $\mathcal{I}m\,(A)$

$$\mathcal{I}m\,(A) = \{\boldsymbol{y} \in \mathbb{R}^n \mid (I_{n\times n} - AA^+)\boldsymbol{y} = \boldsymbol{0}, \quad \boldsymbol{y} \in \mathbb{R}^n\} \tag{3.18}$$

Figure 3.4 provides a graphical representation of the mapping (3.17), showing the intersection $\mathcal{I}m\,(A) \cap \mathcal{P}_y$ and the final polytope $\mathcal{P}_x$, being its projection to the output space $\boldsymbol{x} \in \mathbb{R}^m$.

The intersection formulation is generally more computationally complex to work with than the projection formulation, as it often requires characterising the intersection $\mathcal{I}m\,(A) \cap \mathcal{P}_y$. The graphical comparison of the construction of both intersection and projection polytopes from the same input set is shown on Figure 3.6.

3.2.3 Combined cases

Physical ability polytopes described in Section 2.2 contain two special cases consisting in the combination of intersection and projection polytope formulations: *intersection-projection* and *projection-intersection* formulation.

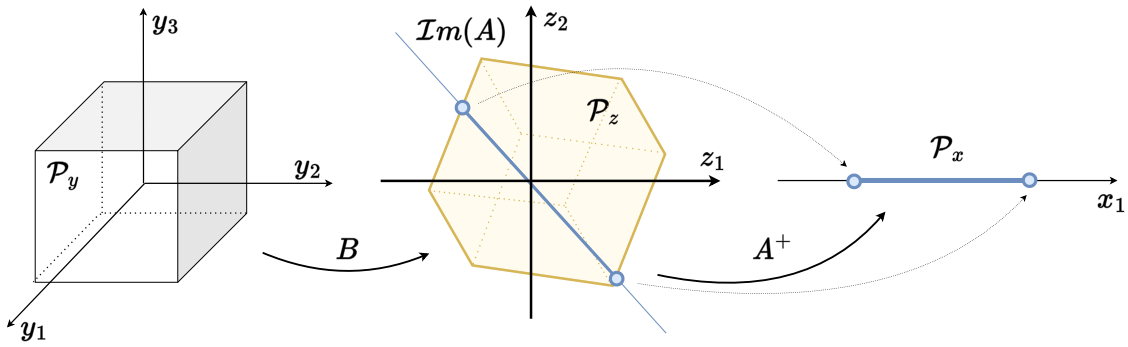

FIGURE 3.7: An example ($n = 3$, $k = 2$, $m = 1$) of a construction of the intersection-projection polytope $\mathcal{P}_x$ from the input set $\mathcal{P}_y$. Geometrically, first the input set $\mathcal{P}_y$ is projected to the intermediate space $\mathbb{R}^k$ to obtain the projection polytope $\mathcal{P}_z$ (yellow). Then this polytope is intersected with the image $\mathcal{I}m\,(A)$ of matrix A (blue middle). Finally the intersection $\mathcal{P}_z \cap \mathcal{I}m\,(A)$ is projected to the output space with the pseudo-inverse A^+ to obtain the final 1D polytope $\mathcal{P}_x$ (blue right).

3.2.3.1 Intersection-projection formulation

The unified generic polytope formulation given by the equation (3.3) can be seen as a special case of the polytope formulation that contains both intersection and projection formulation

$$\mathcal{P}_x \in \{\boldsymbol{x} \in \mathbb{R}^m \mid \underbrace{A\boldsymbol{x} = \boldsymbol{z} + \boldsymbol{b}_z}_{\text{intersection}}, \underbrace{\boldsymbol{z} = B\boldsymbol{y}}_{\text{projection}}, \ \boldsymbol{y} \in \mathcal{P}_y\} \tag{3.19}$$

where $\boldsymbol{x} \in \mathbb{R}^m$ is the output vector, $\boldsymbol{y} \in \mathbb{R}^n$ is an input vector and $\boldsymbol{z} \in \mathbb{R}^k$ is an intermediate space vector, where $n \geq k \geq m$. Matrix $B \in \mathbb{R}^{k \times n}$ is a projector matrix from n dimensional input space to the k dimensional intermediate space, matrix $A \in \mathbb{R}^{k \times m}$ is a projector matrix from m dimensional output space to the k dimensional intermediate space and $\boldsymbol{b}_z \in \mathbb{R}^k$ is the bias vector, defined in the intermediate space.

In this work, this polytope formulation is named *intersection-projection* as it is a special case of a intersection formulation (3.11) with polytope shaped input set which has the projection formulation. The final polytope $\mathcal{P}_x$ can be expressed as

$$\mathcal{P}_x \in \{\boldsymbol{x} \in \mathbb{R}^m \mid A\boldsymbol{x} = \boldsymbol{z} + \boldsymbol{b}_z, \ \boldsymbol{z} \in \mathcal{P}_z\} \tag{3.20}$$

where its input set polytope $\mathcal{P}_z$ is defined using the projection formulation

$$\mathcal{P}_z \in \{\boldsymbol{z} \in \mathbb{R}^k \mid z = B\boldsymbol{y}, \ \boldsymbol{y} \in \mathcal{P}_y\} \tag{3.21}$$

This formulation can also be transformed to an equivalent projection formulation, as described for the intersection formulation is Section 3.2.2, resulting in a polytope

$$\mathcal{P}_x = \{\boldsymbol{x} \in \mathbb{R}^m | \ \boldsymbol{x} = A^+\boldsymbol{z} - A^+\boldsymbol{b}_z, \ \boldsymbol{z} + \boldsymbol{b}_z \in \mathcal{I}m\,(A) \cap \mathcal{P}_z\} \tag{3.22}$$

However, this polytope formulation is much more computationally complex than both intersection and projection formulation, as it requires first computing the projection polytope $\mathcal{P}_z$ and then finding the intersection $\mathcal{I}m\,(A) \cap \mathcal{P}_z$ in order to find the polytope $\mathcal{P}_x$. A graphical example of constructing the intersection-projection formulation polytope is shown on Figure 3.7.

Remark. *This formulation corresponds to the formulations of the wrench and stiffness capacity polytopes for human musculoskeletal models, described in Section 2.2.3 and Section 2.2.6.*

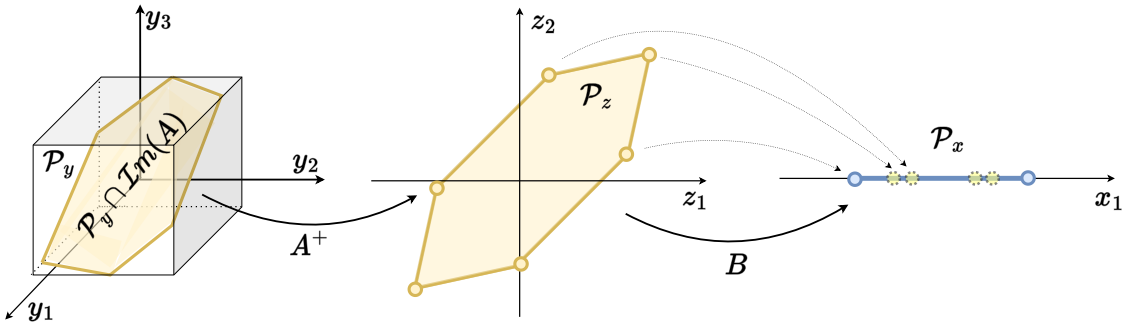

FIGURE 3.8: An example ($n = 3$, $k = 2$, $m = 1$) of a construction of the projection-intersection polytope $\mathcal{P}_x$ from the input set $\mathcal{P}_y$. Geometrically, first the input set $\mathcal{P}_y$ is intersected with the image $\mathcal{I}m(A)$ of the matrix A (yellow left). The intersection $\mathcal{P}_y \cap \mathcal{I}m(A)$ is then projected to the intermediate space using the pseudo-inverse A^+ to obtain the polytope $\mathcal{P}_z$ (yellow middle). Finally, an affine transformation B is applied to the polytope $\mathcal{P}_z$ to transforming it to the final polytope 1D ($m = 1$) $\mathcal{P}_x$ (blue right).

3.2.3.2 Projection-intersection formulation

A different special case of polytope formulation combining both projection and intersection formulation can be expressed as

$$\mathcal{P}_x \in \{\boldsymbol{x} \in \mathbb{R}^m \mid \underbrace{\boldsymbol{x} = B\boldsymbol{z} + \boldsymbol{b}_x}_{\text{projection}}, \underbrace{A\boldsymbol{z} = \boldsymbol{y}}_{\text{intersection}}, \ \boldsymbol{y} \in \mathcal{P}_y\} \tag{3.23}$$

where $\boldsymbol{x} \in \mathbb{R}^m$ is the output vector, $\boldsymbol{y} \in \mathbb{R}^n$ is an input vector and $\boldsymbol{z} \in \mathbb{R}^k$ is an intermediate space vector, where $n \geq k \geq m$. Matrix $A \in \mathbb{R}^{n \times k}$ is a projector matrix from n dimensional input space to the k dimensional intermediate space, matrix $B \in \mathbb{R}^{m \times k}$ is a projector matrix from k dimensional intermediate space to the m dimensional output space and $\boldsymbol{b}_x \in \mathbb{R}^m$ is the bias vector, defined in the output space.

In this work, this polytope formulation is named *projection-intersection* as it is a special case of a projection formulation (3.6) with polytope shaped input set which has the intersection formulation. The final polytope $\mathcal{P}_x$ can be expressed as

$$\mathcal{P}_x \in \{\boldsymbol{x} \in \mathbb{R}^m \mid \boldsymbol{x} = B\boldsymbol{z} + \boldsymbol{b}_x, \ \boldsymbol{z} \in \mathcal{P}_z\} \tag{3.24}$$

where its input set polytope $\mathcal{P}_z$ is defined using the intersection formulation

$$\mathcal{P}_z \in \{\boldsymbol{z} \in \mathbb{R}^n \mid A\boldsymbol{z} = \boldsymbol{y}, \ \boldsymbol{y} \in \mathcal{P}_y\} \tag{3.25}$$

This formulation can also be transformed to an equivalent projection formulation, using the same procedure described in Section 3.2.2, resulting in a polytope

$$\mathcal{P}_x = \{\boldsymbol{x} \in \mathbb{R}^m | \ \boldsymbol{x} = BA^+\boldsymbol{y} + \boldsymbol{b}_x, \ \boldsymbol{y} \in \mathcal{I}m(A) \cap \mathcal{P}_y\} \tag{3.26}$$

This polytope formulation is again much more computationally complex than both intersection and projection formulation, as it requires first computing the intersection $\boldsymbol{y} \in \mathcal{I}m(A) \cap \mathcal{P}_y$, followed by the projection $BA^+\boldsymbol{y}$. A graphical example of constructing the projection-intersection formulation polytope is shown on Figure 3.8.

Remark. *This generic formulation corresponds to the formulation of the velocity capacity polytope for human musculoskeletal models, described in the Section 2.2.5.*

3.2.4 Synthesis of polytope formulations

Section 3.2 proposes a structured view on different families of polytope formulations, with the aim to group the polytope formulations that can be used with the same polytope transformation algorithms. The discussed families are based on the generic polytope formulation (3.3),

$$\mathcal{P}_x = \{\boldsymbol{x} \in \mathbb{R}^m \mid A\boldsymbol{x} = B\boldsymbol{y} + \boldsymbol{b},\ \boldsymbol{y} \in \mathcal{P}_y\} \tag{3.27}$$

which, as described in Chapter 2 (synthesis Section 2.4), unifies all the common polytope representations for characterising physical abilities of robots and humans.

There are two main families of polytope formulations discussed in this section: projection and intersection polytope formulation.

The projection formulation (3.47) corresponds to the specific case of the formulation (3.27) where the matrix A is an identity matrix $I_{m\times m}$. The projection formulation is defined as the linear affine transformation of the input set $\mathcal{P}_y$ into the output space through the matrix B, forming the polytope $\mathcal{P}_x$.

The intersection formulation (3.11) corresponds to the specific case of the formulation (3.27) where matrix B is an identity matrix $I_{n\times n}$. As described in Section 3.2.2, the polytopes $\mathcal{P}_x$ expressed in this formulation can be seen geometrically as a linear projection of the intersection of the input set $\mathcal{P}_y$ and the image space $\mathcal{I}m\,(A)$ of the matrix A, to the output space.

Furthermore, two special cases of these formulations are discussed in Section 3.2.3 intersection-projection formulation, corresponding to the intersection polytope where the input set $\mathcal{P}_y$ has the projection formulation, and projection-intersection formulation, corresponding to the projection polytope where the input set $\mathcal{P}_y$ has the intersection formulation.

Table 3.1 shows, in a condensed manner, how the introduced polytope forms correspond to the generic formulation (3.27).

Formulation	Eqn.	$\boldsymbol{x}$	A	B	$\boldsymbol{y}$	Input set $\mathcal{P}_y$	$\boldsymbol{b}$
Projection	3.6	$\boldsymbol{x} \in \mathbb{R}^m$	$I_{m\times m}$	B	$\boldsymbol{y} \in \mathbb{R}^n$	$\mathcal{P}_y$	$\boldsymbol{b}_x \in \mathbb{R}^m$
Intersection	3.11	$\boldsymbol{x} \in \mathbb{R}^m$	A	$I_{n\times n}$	$\boldsymbol{y} \in \mathbb{R}^n$	$\mathcal{P}_y$	$\boldsymbol{b}_y \in \mathbb{R}^n$
Combined cases							
Intersection-projection	3.19	$\boldsymbol{x} \in \mathbb{R}^m$	A	B	$\boldsymbol{y} \in \mathbb{R}^n$	$\mathcal{P}_y$	$\boldsymbol{b}_z \in \mathbb{R}^k$
Projection-intersection	3.23	$\boldsymbol{x} \in \mathbb{R}^m$	$I_{m\times m}$	B	$\boldsymbol{z} \in \mathbb{R}^k$	$\mathcal{P}_z$	$\boldsymbol{b}_x \in \mathbb{R}^m$

TABLE 3.1: Ths table brings the correspondence in the introduced polytope formulations and the generic formulation (3.27).

The following sections leverage the proposed view on different polytope formulations and provide an overview of standard algorithms for transforming these families of polytope formulations to standard polytope representations ($\mathcal{H}$ and $\mathcal{V}$).

3.3 An overview of polytope transformation strategies

Transforming polytopes to their standard representations ($\mathcal{V}$ and $\mathcal{H}$) is a well studied problem in literature. Over the years, many efficient algorithms [122, 129, 130] have been proposed for vertex enumeration, finding the $\mathcal{V}$-representation, and facet enumeration, finding the $\mathcal{H}$-representation.

However, the algorithms are often developed for a specific polytope formulation, where their efficiency might vary considerably with the size of the problem (dimension of input and output spaces) [131] and the complexity of the evaluated polytope (number of faces and vertices) [132]. Therefore, the choice of the appropriate polytope transformation algorithm depends on to the polytope formulation, the representation required by the application, as well as the required time-efficiency.

This section brings a non-exhaustive overview of standard algorithms for transforming different polytope formulations (intersection and projection) with different input set shapes (interval $\mathcal{I}_y$ and polytope $\mathcal{P}_y$) into different standard representations ($\mathcal{V}$ and $\mathcal{H}$). Additionally, Section 3.3.4 discusses different polytope approximation strategies in the context of improving the efficiency of the vertex and half-plane evaluation. Finally, Section 3.3.5 brings a condensed view of the proposed overview in a form of Table 3.2.

3.3.1 Standard polytope representation conversion algorithms

Converting polytope representations from half-plane ($\mathcal{H}$) to vertex ($\mathcal{V}$) and *vice-versa* is a well studied problem in literature. Many different algorithms have been developed capable of performing the transformations in both directions, with various degrees of efficiency.

There are two main families of approaches developed in the literature: Incremental Algorithms (INA) and Graph Traversal Algorithms (GTA) [133, Chapter 8.] [134].

Incremental Algorithms (INA) construct the $\mathcal{V}$-representation of polytopes by iteratively intersecting the half-planes defined by the $\mathcal{H}$-representation, while keeping only the intersection points that correspond to the vertices of the polytope. These methods construct the $\mathcal{H}$-representation in an iterative manner as well, by constructing the half-plane equations from the vertices and keeping only the ones corresponding to the faces of the polytope. Examples of incremental algorithms are the Double-Description Method (DDM) [129, 135] or the Beneath and beyond method [136].

Graph Traversal Algorithms (GTA) are based on representing the polytope in a form of a graph of its vertices connected by its edges. This graph is then traversed in different fashions in order to obtain all the vertices ($\mathcal{V}$-representation) and faces ($\mathcal{H}$-representation) of the polytope. Examples of such methods are the Pivoting Method (PIM) by Bremner *et al.* [122], the Gift-wrapping algorithm by Seidel [137] of the Reverse search algorithm by Avis and Fukuda [130]

A comprehensive review of different conversion algorithm was proposed by Avis *et al.* [134], comparing the efficiency of different families of algorithms on different standard polytope benchmarks, as well as their available implementations. In general, the GTA methods are more efficient for higher dimensional problems, while INA methods are more efficient for lower dimensional problems, for dimensions up to 12 [133, Chapter 8.3].

These methods are standard building blocks for using polytopes in practical applications as well as for computing different operations over polytopes, such as intersections, Convex-Hulls and Minkowski sums (described in Appendix A). However, the polytope formulations families described in the previous section are not expressed neither as $\mathcal{H}$ nor $\mathcal{V}$-representation. Therefore, these formulations require either additional steps in order to be used with standard INA or GTA algorithms, or in some cases, dedicated algorithms that are specific to their formulations.

3.3.2 Strategies for the intersection formulation

This section brings an overview of approaches used for transformation of polytopes with the intersection formulation, described in Section 3.2.2, into their respective $\mathcal{V}$ and $\mathcal{H}$-representation.

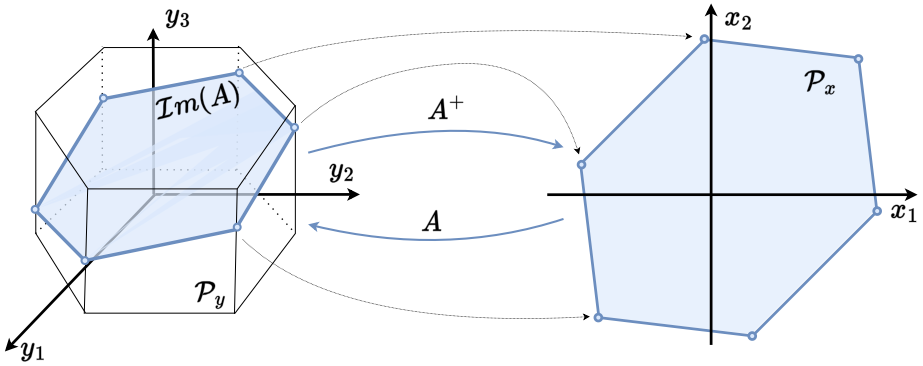

FIGURE 3.9: An example of constructing an intersection formulation polytope with an $m = 2$ dimensional output space and an $n = 3$ dimensional input space. The input set $\mathcal{P}_y$ has a polytope form, and its intersection with the image of the matrix A is shown in blue (left). The final polytope $\mathcal{P}_x$ is shown in blue as well (right).

The polytopes with the intersection formulation can be expressed as

$$\mathcal{P}_x = \{\boldsymbol{x} \in \mathbb{R}^m | \; A\boldsymbol{x} = \boldsymbol{y} + \boldsymbol{b}_y, \; \boldsymbol{y} \in \mathcal{P}_y\} \tag{3.28}$$

where $\boldsymbol{y} \in \mathbb{R}^n$, $\boldsymbol{b}_y \in \mathbb{R}^n$ are the n dimensional input vector and input bias, $\boldsymbol{x} \in \mathbb{R}^m$ is the m dimensional output vector, $A \in \mathbb{R}^{n \times n}$ is a transformation matrix from output to the input space and $\mathcal{P}_y$ is the input set.

The geometrical representation of the polytope $\mathcal{P}_x$ defined by (3.28) is shown on Figure 3.9, for the example of $n = 3$ dimensional input set $\mathcal{P}_y$ and $m = 2$ dimensional polytope $\mathcal{P}_x$. The resulting polytope $\mathcal{P}_x$ is an affine projection of the intersection of the input polytope $\mathcal{P}_y$ and the image $\mathcal{I}m\,(A)$ of the matrix A into the lower dimensional output space. As shown on Figure 3.9, the vertices and faces of the polytope $\mathcal{P}_x$ do not correspond to the projection of the vertices and faces of the input polytope $\mathcal{P}_y$. Rather, as discussed in Section 3.2.2, the polytope $\mathcal{P}_x$ is generated by projecting the intersection $\mathcal{I}m\,(A) \cap \mathcal{P}_y$, of the polytope $\mathcal{P}_y$ and the image $\mathcal{I}m\,(A)$ of the matrix A, to the output space. Therefore, there is a unique mapping between the vertices and faces of the intersection $\mathcal{I}m\,(A) \cap \mathcal{P}_y$, and the vertices and the faces of the polytope $\mathcal{P}_x$, given through the pseudo-inverse inverse [128] of the matrix A.

Algorithms for finding the $\mathcal{V}$ and $\mathcal{H}$-representation vary in complexity, depending on the structure of the input set $\mathcal{P}_y$.

3.3.2.1 Finding the $\mathcal{H}$-representation

If the input set $\mathcal{P}_y$ is expressed in its $\mathcal{H}$-representation

$$\mathcal{P}_y = \{\boldsymbol{y} \in \mathbb{R}^n \mid H_y\boldsymbol{y} \leq \boldsymbol{d}_y\} \tag{3.29}$$

then finding the $\mathcal{H}$-representation of the polytope $\mathcal{P}_x$ is straightforward, by replacing the vector $\boldsymbol{y}$ with $A\boldsymbol{x} - \boldsymbol{b}_y$ in the equation above

$$\mathcal{P}_x = \{\boldsymbol{x} \in \mathbb{R}^m \mid H_yA\boldsymbol{x} \leq \boldsymbol{d}_y - H_y\boldsymbol{b}_y\} \tag{3.30}$$

Even though this $\mathcal{H}$-representation (3.30) of the polytope $\mathcal{P}_x$ follows directly from its definition and can be easily expressed, it might not be minimal. This means that even though the equation

(3.30) is correct and fully describes the polytope $\mathcal{P}_x$, there might be some redundant half-planes. Many algorithms have been developed over the years for removing the redundant half-planes equations [138], and their computational complexity is generally equivalent to solving a series of Linear Program (LP) problems [139],[133, Chapter 7.2]. Therefore, depending on the application and the computational complexity of the polytope description necessary, such techniques can be used to reduce the equation (3.30) to the minimal set of linear constraints.

However, if the polytope $\mathcal{P}_y$ is not expressed in $\mathcal{H}$-representation, in most cases the most efficient strategy is to first transform it to its $\mathcal{H}$-representation and the apply the manipulation described above. For example if the polytope $\mathcal{P}_y$ is expressed by its $\mathcal{V}$-representation,

$$\mathcal{P}_y = \text{Conv}\left(\boldsymbol{y}_{v1},\ \boldsymbol{y}_{v2},\ \ldots\right) \tag{3.31}$$

where $\boldsymbol{y}_{vi} \in \mathbb{R}^n$ are the vertices of $\mathcal{P}_y$. These vertices can be transformed to the $\mathcal{H}$-representation using the techniques described in Section 3.3.1, which can then be used with the above described approach (3.30) to obtain the $\mathcal{H}$-representation of the polytope $\mathcal{P}_x$.

Therefore, when it comes to finding the $\mathcal{H}$-representation of the polytope $\mathcal{P}_x$, the main complexity comes form determining the $\mathcal{H}$-representation of the input set $\mathcal{P}_y$. The input set $\mathcal{P}_y$, in general case, can have any formulation, not necessarily corresponding neither to the $\mathcal{V}$ nor to the $\mathcal{H}$-representation.

Therefore, in the remainder of this section three special cases of the input set formulations are discussed: intersection and projection formulation of the input set $\mathcal{P}_y$, as well as the interval form $\mathcal{I}_y$.

Special case: intersection formulation

If the input set $\mathcal{P}_y$ has the intersection formulation itself

$$\mathcal{P}_y = \{\boldsymbol{y} \in \mathbb{R}^n \mid C\boldsymbol{y} = \boldsymbol{z} + \boldsymbol{b}_z,\ \boldsymbol{z} \in \mathcal{P}_z\} \tag{3.32}$$

where $\boldsymbol{z} \in \mathcal{R}^k$ is a new its k dimensional input vector, bounded within the set $\mathcal{P}_z$, $\boldsymbol{b}_z \in \mathbb{R}^k$ is the bias vector and the matrix $C \in \mathbb{R}^{k \times n}$ is a projector form the n dimensional space to the k dimensional space, where $k \geq n$.

Then the two intersection formulations can be combined into the new polytope $\mathcal{P}_x$, by replacing $\boldsymbol{y}$ with $A\boldsymbol{x} - \boldsymbol{b}_y$

$$\mathcal{P}_x = \{\boldsymbol{x} \in \mathbb{R}^m \mid \underbrace{CA}_{A'}\boldsymbol{x} = \boldsymbol{z} + \underbrace{\boldsymbol{b}_z + C\boldsymbol{b}_y}_{\boldsymbol{b}'_z},\ \boldsymbol{z} \in \mathcal{P}_z\} \tag{3.33}$$

This new combined polytope has the intersection formulation as well, where the polytope $\mathcal{P}_z$ becomes its input set. Therefore, all the approaches described Section 3.3.2.1 are valid for this polytope respectively.

Special case: projection formulation

The case where the input set $\mathcal{P}_y$ has the projection formulation corresponds to the combined special case called intersection-projection formulation, described in Section 3.2.3.1.

The most straightforward approach to finding the $\mathcal{H}$-representation of the polytope $\mathcal{P}_x$ consists in first finding the $\mathcal{H}$-representation of the input set $\mathcal{P}_y$. Then the above described method (3.29-3.30) can be used to find the final $\mathcal{H}$-representation of the polytope $\mathcal{P}_x$.

The common strategies for finding the $\mathcal{H}$-representation of polytopes with the projection formulation are described in Section 3.3.3.

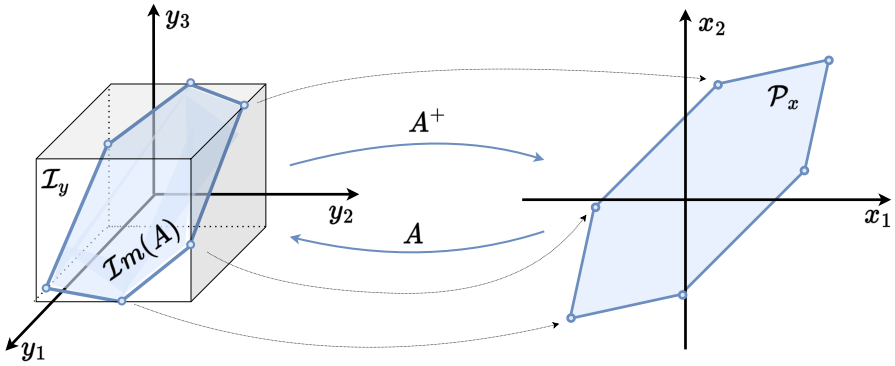

FIGURE 3.10: An example of the constructing an intersection formulation polytope with $m = 2$ dimensional output space and $n = 3$ dimensional input space. The intersection $\mathcal{I}m\,(A) \cap \mathcal{I}_y$ of the hyperrectangle shaped input space $\mathcal{I}_y$ with the image of the matrix A is shown in blue (left). This intersection can be projected to the output space using the inverse of the matrix A to obtain the polytope $\mathcal{P}_x$ (blue right).

Special case: Interval form $\mathcal{I}_y$

The intersection polytope formulation (3.28) with the interval limits

$$\mathcal{I}_y = \{\boldsymbol{y} \in \mathbb{R}^n \mid \boldsymbol{y} \in [\boldsymbol{y}_{min}, \boldsymbol{y}_{max}]\} \tag{3.34}$$

is a spacial case of the intersection polytope formulation where the input space limits are defined as independent min-max ranges

$$\mathcal{P}_x = \{\boldsymbol{x} \in \mathbb{R}^m \mid A\boldsymbol{x} = \boldsymbol{y} + \boldsymbol{b}_y,\ \boldsymbol{y}_{min} \leq \boldsymbol{y} \leq \boldsymbol{y}_{max}\} \tag{3.35}$$

Figure 3.10 graphically represents the construction the polytope $\mathcal{P}_x$ from the input set $\mathcal{I}_y$.

When the input set has the interval form $\mathcal{I}_y$ the $\mathcal{H}$-representation of the polytope $\mathcal{P}_x$ can be found directly by substituting $\boldsymbol{y}$ with $A\boldsymbol{x} - \boldsymbol{b}_y$ in the interval equation $\boldsymbol{y} \in [\boldsymbol{y}_{min}, \boldsymbol{y}_{max}]$ and rewriting it in the matrix from

$$\underbrace{\begin{bmatrix} A \\ -A \end{bmatrix}}_{H_x} \boldsymbol{x} \leq \underbrace{\begin{bmatrix} \boldsymbol{y}_{max} + \boldsymbol{b}_y \\ -\boldsymbol{y}_{min} + \boldsymbol{b}_y \end{bmatrix}}_{\boldsymbol{d}_x} \tag{3.36}$$

where matrix $H_x \in \mathbb{R}^{2n \times m}$ and the vector $\boldsymbol{d}_x \in \mathbb{R}^{2n}$ can be used to express the $\mathcal{H}$-representation of the polytope $\mathcal{P}_x$

$$\mathcal{P}_x = \{\boldsymbol{x} \in \mathbb{R}^m | \ H_x\boldsymbol{x} \leq \boldsymbol{d}_x\} \tag{3.37}$$

However, as discussed at the beginning of Section 3.3.2.1, additional steps might be necessary to remove the redundant half-plane equations within the matrix H_x and the vector $\boldsymbol{d}_x$.

3.3.2.2 Finding the $\mathcal{V}$-representation

Finding the $\mathcal{V}$-representation of polytopes with the intersection formulation, is a much more complex operation with respect to finding the $\mathcal{H}$-representation.

In general, finding the $\mathcal{V}$-representation of the polytope $\mathcal{P}_x$, is performed by first finding its $\mathcal{H}$-representation, as described in the previous section (Section 3.3.2.1). Then, depending on the size of the problem, different standard representation conversion algorithms, described in Section 3.3.1, can be used to obtain its $\mathcal{V}$-representation.

However, as the special case of the intersection formulation, where the input set has interval shape $\mathcal{I}_y$, is particularly present in the robotics literature, several efficient algorithms have been proposed for finding its $\mathcal{V}$-representation directly.

Special case: Interval form $\mathcal{I}_y$

When the input set has interval shape $\mathcal{I}_y$, the compact form of this polytope $\mathcal{P}_x$ with the intersection formulation can be expressed as

$$\mathcal{P}_x = \{\boldsymbol{x} \in \mathbb{R}^m \mid A\boldsymbol{x} = \boldsymbol{y} + \boldsymbol{b}_y,\ \boldsymbol{y} \in [\boldsymbol{y}_{min},\ \boldsymbol{y}_{max}]\} \tag{3.38}$$

Several efficient algorithms were developed for finding all the vertices of polytopes with this formulation. These algorithms, exploit the geometry of the problem and provide better efficiency than the standard conversion INA or GTA based methods.

An efficient algorithm for vertex enumeration of the intersection type polytope is proposed by Gouttefarde *et al.* [140]. The algorithm assumes that the output space dimension is always $m = 2$, in which case, the polytope $\mathcal{P}_x$ becomes a 2D polygon. The algorithm then exploits the 2D geometry of the problem and efficiently navigates the boundaries of the polygon in search for extremities. However, this algorithm does not scale well to higher dimensional problems.

A different algorithm, finding the $\mathcal{V}$-representation of the intersection polytope $\mathcal{P}_x$ with interval limits $\mathcal{I}_y$, is proposed by Chiacchio *et al.* [35]. The algorithm leverages the hyperrectangle geometry of the interval input set $\mathcal{I}_y$ and performs an efficient exhaustive search through its faces. This algorithm has been improved by Sasaki [106], significantly reducing the computational complexity by exploiting the geometry of the intersection of the hyperrectangle (interval) $\mathcal{I}_y$ and the image of matrix $\mathcal{I}m\,(A)$. Moreover, this improved algorithm is based on the equivalent projection formulation of the intersection polytope, described in the Section 3.2.2.

$$\mathcal{P}_x = \{\boldsymbol{x} \in \mathbb{R}^m \mid \boldsymbol{x} = A^+\boldsymbol{y} + A^+\boldsymbol{b}_y,\ \boldsymbol{y} \in \mathcal{I}m\,(A) \cap [\boldsymbol{y}_{min}, \boldsymbol{y}_{max}]\} \tag{3.39}$$

Both of these algorithms, Chiacchio *et al.* [35] and Sasaki [106], are based on the efficient exhaustive search of the hyperrectangle $\mathcal{I}_y$ faces. However, the number of faces of the hyperrectangle grows exponentially with the input space dimension

$$N_{n,k} = 2^{n-k}\binom{n}{k}$$

where n is the dimension of the input space, $N_{n,k}$ is the number of the k dimensional hyperrectangle faces. Therefore, as the dimension of the input space n grows, the computational efficiency of these algorithms decreases exponentially.

3.3.3 Strategies for the projection formulation

This section brings a short overview of methods used for the transformation of polytopes with projection formulation, described in Section 3.2.1, into their respective $\mathcal{V}$ and $\mathcal{H}$-representation.

The projection polytope (3.6) with polytope input set $\mathcal{P}_y$ can be expressed as

$$\mathcal{P}_x = \{\boldsymbol{x} \in \mathbb{R}^m | \ \boldsymbol{x} = B\boldsymbol{y} + \boldsymbol{b}_x,\ \boldsymbol{y} \in \mathcal{P}_y\} \tag{3.40}$$

where $\boldsymbol{y} \in \mathbb{R}^n$ is an n dimensional input vector, $\boldsymbol{x}, \boldsymbol{b}_x \in \mathbb{R}^m$ are the m dimensional output vector and bias, while $B \in \mathbb{R}^{m \times n}$ is a transformation matrix from input to the output space.

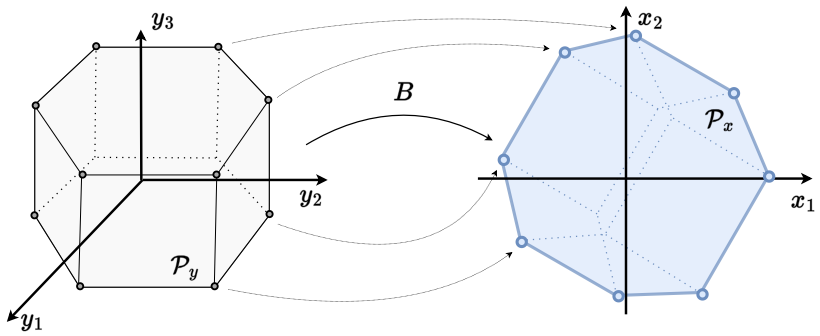

FIGURE 3.11: An example of the constructing a projection formulation polytope with $m = 2$ dimensional output space and $n = 3$ dimensional input space, where the input space $\mathcal{P}_y$ has a polytope shape. In this formulation, the whole input space $\mathcal{P}_y$ (grey) is projected using the matrix B to the output space to obtain the polytope $\mathcal{P}_x$ (blue).

The geometrical representation of the polytope $\mathcal{P}_x$ defined by (3.40) is shown on Figure 3.11, for the example of $n = 3$ dimensional polytope $\mathcal{P}_y$ and $m = 2$ dimensional polytope $\mathcal{P}_x$. The resulting polytope $\mathcal{P}_x$ is an affine projection of the input polytope $\mathcal{P}_y$ into the lower dimensional output space. Furthermore, the vertices and the faces of the polytope $\mathcal{P}_x$ correspond to the subset of the projected vertices and faces of the input polytope $\mathcal{P}_y$.

Given the structure of the input set $\mathcal{P}_y$ algorithms for finding the $\mathcal{V}$ and $\mathcal{H}$-representation vary in complexity.

3.3.3.1 Finding the $\mathcal{V}$-representation

If the input set polytope $\mathcal{P}_y$ is expressed using its vertex or $\mathcal{V}$-representation

$$\mathcal{P}_y = \text{Conv}\left(\boldsymbol{y}_{v1},\ \boldsymbol{y}_{v2},\ \ldots\right) \tag{3.41}$$

where $\boldsymbol{y}_{vi} \in \mathbb{R}^n$ are the vertices of the polytope $\mathcal{P}_y$, the vertices of the projection formulation polytope (3.40) can then be found by projecting the vertices of $\mathcal{P}_y$ to the output space using the matrix $B \in \mathbb{R}^{m \times n}$ and calculating their Convex-Hull $\text{Conv}\left(\cdot\right)$

$$\mathcal{P}_x = \text{Conv}\left(B\boldsymbol{y}_{v1},\ B\boldsymbol{y}_{v2},\ \ldots\right) \tag{3.42}$$

The Convex-Hull algorithm finds the $\mathcal{V}$-representation of the projection polytope directly, as the vertices of the polytope $\mathcal{P}_x$ are a subset of the projected vertices $\boldsymbol{y}_{vi}$ of the input polytope $\mathcal{P}_y$, as shown in the example on Figure 3.11.

If the input polytope $\mathcal{P}_y$ is expressed using its half-plane or $\mathcal{H}$-representation,

$$\mathcal{P}_y = \{\boldsymbol{y} \in \mathbb{R}^n \mid H_y\boldsymbol{y} \leq \boldsymbol{d}_y\} \tag{3.43}$$

there are two main approaches decoupling the problem.

The first and straight-forward approach consists in finding the $\mathcal{V}$-representation of the input set $\mathcal{P}_y$ first, using the standard representation conversion methods described in Section 3.3.1. Then the above described procedure (3.41-3.42) can be used to find the $\mathcal{V}$-representation of $\mathcal{P}_x$. However, in many cases, when the input space dimension n is high or if the geometry of the polytope $\mathcal{P}_y$ is complex (large number of faces and vertices), finding its $\mathcal{V}$-representation might be computationally demanding.

The second approach, more efficient in many cases, consists in finding the $\mathcal{H}$-representation of

the polytope $\mathcal{P}_x$ directly from the input set's $\mathcal{H}$-representation. One of the most well known algorithms for projecting the $\mathcal{H}$-representation of polytopes is arguably the Fourier-Motzkin Elimination (FME) described by Dantzig and Eaves [141]. This algorithm uses an iterative inequality elimination method to isolate the set of half-plane equations bounding the final polytope $\mathcal{P}_x$. However, this approach has an exponential complexity with the dimension of the input space n [142], and additionally it does not guarantee a minimal representation [143]. A more efficient algorithm is introduced by Jones *et al.* [144] called Equality-Set Projection (ESP). As opposed to FME, this algorithm is *output sensitive*, having its execution time proportional to the complexity of the final polytope $\mathcal{P}_x$ (it number of faces and vertices), making it particularly well suited to the high dimensional input spaces $\mathbb{R}^n$. A more in depth comparison of these methods, as well their comparison to several other polytope projection algorithms, is brought in the work by Glassle *et al.* [145]. Once the $\mathcal{H}$-representation of the polytope $\mathcal{P}_x$ is obtained using one of these methods, the standard representation conversion methods can be used to obtain the $\mathcal{V}$-representation of the polytope $\mathcal{P}_x$ defined in the m dimensional output space.

The same two approaches can be used in the general case, where the input set $\mathcal{P}_y$ does not correspond neither to the $\mathcal{V}$ nor to the $\mathcal{H}$-representation. The choice of the more suitable one depends on the efficiency of transforming the input set $\mathcal{P}_y$ into its $\mathcal{V}$ or $\mathcal{H}$-representation.

In remaining part of this section three special cases of the input set formulation are discussed: intersection and projection formulation of the input set $\mathcal{P}_y$, as well as the interval form $\mathcal{I}_y$.

Special case: intersection formulation

If the input set $\mathcal{P}_y$ has the intersection formulation, then the formulation of the polytope $\mathcal{P}_x$ corresponds to the projection-intersection formulation, the combined special case described in Section 3.2.3.2.

The $\mathcal{H}$ and $\mathcal{V}$-representations of the input set $\mathcal{P}_y$, with intersection formulation, can be obtained using the strategies discussed in Section 3.3.2. As discussed in Section 3.3.2, in general, the $\mathcal{H}$-representation of intersection polytopes can be found more efficiently than the $\mathcal{V}$-representation, making the second approach described in Section 3.3.3.1 better suited.

Special case: projection formulation

If the input set $\mathcal{P}_y$ has the projection formulation itself

$$\mathcal{P}_y = \{\boldsymbol{y} \in \mathbb{R}^n \mid \boldsymbol{y} = D\boldsymbol{z} + \boldsymbol{b}_y,\ \boldsymbol{z} \in \mathcal{P}_z\} \tag{3.44}$$

where $\boldsymbol{z} \in \mathcal{R}^k$ is its k dimensional input vector, bounded within the set $\mathcal{P}_z$, $\boldsymbol{b}_y \in \mathbb{R}^n$ is the bias vector, and the matrix $D \in \mathbb{R}^{n \times k}$ is a projection matrix form the k dimensional space to the n dimensional space.

Then the two projection formulations can be combined into the new polytope $\mathcal{P}_x$, by replacing the $\boldsymbol{y}$ with $D\boldsymbol{z} + \boldsymbol{b}_y$ in initial formulation of $\mathcal{P}_x$ (3.40).

$$\mathcal{P}_x = \{\boldsymbol{x} \in \mathbb{R}^m \mid \boldsymbol{x} = \underbrace{BD}_{B'}\boldsymbol{z} + \underbrace{B\boldsymbol{b}_z + \boldsymbol{b}_x}_{\boldsymbol{b}'_x},\ \boldsymbol{z} \in \mathcal{P}_z\} \tag{3.45}$$

This new formulation corresponds to the projection formulation as well. Therefore, the same approaches described in Section 3.3.3.1 can be used to find its $\mathcal{V}$-representation respectively.

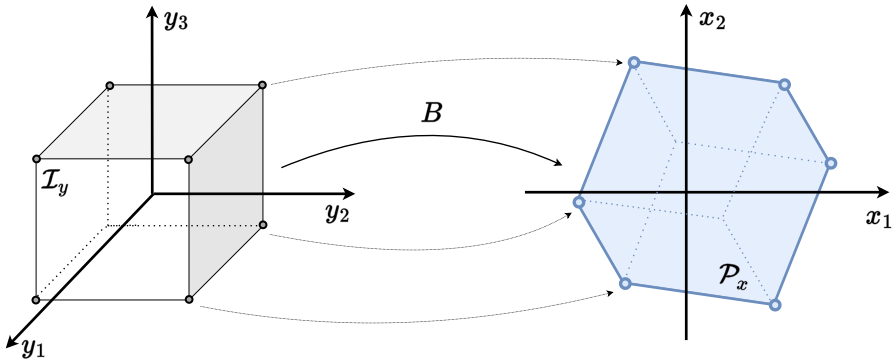

FIGURE 3.12: An example of the constructing a projection formulation polytope with $m = 2$ dimensional output space and $n = 3$ dimensional input space, where the input space $\mathcal{I}_y$ has a hyperrectangle (interval) shape. In the projection formulation, the whole input space $\mathcal{I}_y$ (grey) is projected using the matrix B to the output space to obtain the polytope $\mathcal{P}_x$ (blue).

Special case: Interval form $\mathcal{I}_y$

The projection polytope formulation $\mathcal{P}_x$ with the interval limits $\mathcal{I}_y$

$$\mathcal{I}_y = \{\boldsymbol{y} \in \mathbb{R}^n \mid \boldsymbol{y} \in [\boldsymbol{y}_{min}, \boldsymbol{y}_{max}]\} \tag{3.46}$$

is a spacial case of the projection polytope formulation which can be compactly expressed as

$$\mathcal{P}_x = \{\boldsymbol{x} \in \mathbb{R}^m | \ \boldsymbol{x} = B\boldsymbol{y} + \boldsymbol{b}_x, \ \boldsymbol{y}_{min} \leq \boldsymbol{y} \leq \boldsymbol{y}_{max}\} \tag{3.47}$$

Figure 3.12 graphically represents the construction the polytope $\mathcal{P}_x$ from the input set $\mathcal{I}_y$.

The most straight-forward way of finding the vertices $\boldsymbol{x}_{vi} \in \mathbb{R}^m$ of the projection polytope $\mathcal{P}_x$ is by first enumerating the 2^n vertices $\boldsymbol{y}_{vi} \in \mathbb{R}^n$ of the hyperrectangle (interval) $\mathcal{I}_y$, by creating a list of all the combinations of the minimal $\boldsymbol{y}_{min}$ and maximal $\boldsymbol{y}_{max}$ values of $\boldsymbol{y}$

$$\boldsymbol{y}_{v1} = \begin{bmatrix} y_{1,min} \\ y_{2,min} \\ \dots \\ y_{n,min} \end{bmatrix}, \quad \boldsymbol{y}_{v2} = \begin{bmatrix} y_{1,max} \\ y_{2,min} \\ \dots \\ y_{n,min} \end{bmatrix}, \quad \dots, \quad \boldsymbol{y}_{v2^n} = \begin{bmatrix} y_{1,max} \\ y_{2,max} \\ \dots \\ y_{n,max} \end{bmatrix} \tag{3.48}$$

Then these vertices can be projected to the lower dimensional output space $\mathbb{R}^m$ using the projection matrix $B \in \mathbb{R}^{m \times n}$, where the polytope $\mathcal{P}_x$ can be found by calculating the Convex-Hull $\text{Conv}(\cdot)$ of the projected points.

$$\mathcal{P}_x = \text{Conv}\left(B\boldsymbol{y}_{v1}, B\boldsymbol{y}_{v2}, \ \dots, \ B\boldsymbol{y}_{v2^n}\right) \tag{3.49}$$

The complexity of this approach depends on two factors. The dimension of the of the input space n and the dimension of the output space m. The number of vertices of the hyperrectangle (interval) grows exponentially (2^n) with the dimension of the input space n, and constructing a matrix of $2^n \times n$ entries can become impractical. On the other hand, as this approach requires a Convex-Hull algorithms, which are executed in the m-dimensional output space, the dimension m might make the Convex-Hull algorithm impractical as their complexity grows significantly with the dimension of the space m [74].

Therefore for higher dimensional output spaces (typically $m \geq 4$) and input spaces (causing the memory issues due to $2^n \times n$ matrix construction) a more efficient approach might be to first

calculate the $\mathcal{H}$-representation of this polytope, using the methods described in the following section, and then use standard representation conversion methods (Section 3.3.1), to find its $\mathcal{V}$-representation.

3.3.3.2 Finding the $\mathcal{H}$-representation

If the input polytope $\mathcal{P}_y$ is expressed using its half-plane or $\mathcal{H}$-representation,

$$\mathcal{P}_y = \{\boldsymbol{y} \in \mathbb{R}^n \mid H_y \boldsymbol{y} \leq \boldsymbol{d}_y\} \tag{3.50}$$

the straight-forward approach consists in decoupling the problem and find the $\mathcal{V}$-representation first, using the standard representation conversion methods described in Section 3.3.1, then follow the above described procedure (Section 3.3.3.1) to find the $\mathcal{V}$-representation of $\mathcal{P}_x$.

As described in the previous section, in many cases, when the input space dimension n is high or if the geometry of the polytope $\mathcal{P}_y$ is complex (large number of faces and vertices), finding its $\mathcal{V}$-representation might be computationally demanding. In those cases, if the $\mathcal{H}$-representation of the polytope $\mathcal{P}_y$ is available, the $\mathcal{H}$-representation of the polytope $\mathcal{P}_x$ can be found directly using the algorithms such as FME and ESP.

These same two approaches can be used in the general case, where the input set $\mathcal{P}_y$ does not correspond neither to the $\mathcal{V}$ nor to the $\mathcal{H}$-representation. The choice of the more suitable one depends on the efficiency of transforming the input set $\mathcal{P}_y$ into its $\mathcal{V}$ or $\mathcal{H}$-representation.

Special case: Interval form $\mathcal{I}_y$

The projection polytope formulation $\mathcal{P}_x$ with the interval limits $\mathcal{I}_y$ can be compactly expressed

$$\mathcal{P}_x = \{\boldsymbol{x} \in \mathbb{R}^m | \ \boldsymbol{x} = B\boldsymbol{y} + \boldsymbol{b}_x, \ \boldsymbol{y}_{min} \leq \boldsymbol{y} \leq \boldsymbol{y}_{max}\} \tag{3.51}$$

Apart from the two approaches described at the beginning of this section (Section 3.3.3.2), there are several more efficient algorithms introduced in the literature, specific for this formulation and typically exploiting the polytope's zonotope structure.

One such efficient algorithm, exploiting the geometry of the formulation (3.51) of the projection polytope $\mathcal{P}_x$ with interval limits $\mathcal{I}_y$, is introduced by Bouchard *et al.* [146] and improved by Gouttefarde and Krut [147]. It is often referred to as Hyper-Plane Shifting Method (HPSM). This algorithm finds the minimal $\mathcal{H}$-representation of the polytope (3.51) by efficiently performing the exhaustive search of all the possible half-plane combinations corresponding to the $m-1$ dimensional polytope $\mathcal{P}_x$ faces. Even-though much more efficient than the Fourier-Motzkin Elimination (FME), this algorithm still has considerable (binomial) complexity, as it relies on the exhaustive search in the n dimensional input space where the number N_h of hyper-planes to be tested equals to

$$N_h = \binom{n}{m-1}$$

3.3.4 Polytope approximation strategies

Exact vertex and facet enumeration methods, such as the standard representation conversion methods described in Section 3.3.1 or more specific methods for the projection and the intersection formulation described in Section 3.3.2 and Section 3.3.3, rely on different versions of exhaustive search in the n-dimensional input space, which is often higher dimensional than the output space $n > m$. Their execution time is typically exponential, or polynomial in the

best case, with respect to the dimension of the input space n, the number of vertices and the faces of the input set $\mathcal{P}_y$ and the final polytope $\mathcal{P}_x$ [132]. Therefore, in cases where the input space is much higher dimensional than the output space $n \gg m$ or when the polytopes $\mathcal{P}_x$ and $\mathcal{P}_y$ have complex geometries (large number of faces and vertices), such approaches may not be practically viable for interactive, in-the-loop, applications.

To overcome this issue, various approximate approaches, such as the Ray Shooting Method (RSM) [148] and the Convex-Hull Method (CHM) [149] have been developed, reducing the complexity and improving the execution time.

The RSM algorithms are relatively simple to setup as they rely on different forms of sampling (shooting rays) of the polytope $\mathcal{P}_x$, in the low dimensional output space. Their execution time is proportional to the number of sampling directions (rays shot), therefore, by choosing an appropriate sampling strategy, the RSM algorithms can provide an efficient approximation of the polytope $\mathcal{P}_x$. However, these algorithms do not provide any bound on their estimation error and rely highly on hand tuned initial parameters. Figure 3.13 shows a visual example of a RSM approximation of a polygon $\mathcal{P}_x$ using uniform sampling in the output space with 8 rays.

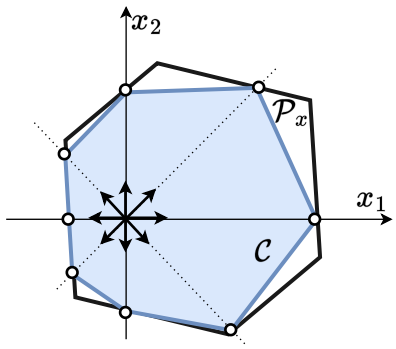

FIGURE 3.13: An example of the approximation $\mathcal{C}$ of the 2D ($m = 2$) polytope $\mathcal{P}_x$ using a RSM algorithm. The input space is sampled with 8 rays.

The CHM algorithms, first introduced by Lassez [149] and Huynh *et al.* [150], propose an efficient iterative approximation of the polytope $\mathcal{P}_x$, while at the same time avoiding the complexity of the exhaustive search. The CHM algorithms simplify the final polytope $\mathcal{P}_x$ geometry by removing the need to find all the faces and vertices of the polytope. Such algorithms are usually defined in the lower-dimensional output space $\boldsymbol{x} \in \mathbb{R}^m$ making them *output sensitive*, having the execution time proportional to the number of vertices and the faces of the output space polytope $\mathcal{P}_x$. In addition to searching in the lower m dimensional output space, they enable finding an efficient inner (or outer) approximation of the polytope $\mathcal{P}_x$, while satisfying a user defined level of accuracy. Finally, they find both the vertices and the faces of the polytope $\mathcal{P}_x$ at the same time, providing both $\mathcal{V}$ and the $\mathcal{H}$-representation [145].

The execution of a typical CHM algorithm consists in two phases. In the first phase an initial approximation of the polytope $\mathcal{P}_x$ is constructed finding a subset of $m+1$ vertices forming an initial m-dimensional Convex-Hull $\mathcal{C}$ approximation of the polytope $\mathcal{P}_x$. In the second phase, the Convex-Hull approximation $\mathcal{C}$ is refined iteratively until the desired accuracy is reached. The CHM algorithms use a sequence of Linear Programs (LPs) to find new vertices of the polytope in each iteration, followed by the Convex-Hull algorithm to group them to the faces. An example of the CHM algorithm iterations for $m = 2$ output space is shown on Figure 3.14.

The CHM algorithms have several limitations though. The resolution of the algorithms relies on the iterative application of the Convex-Hull algorithm which complexity grows exponentially for output space dimensions $m > 3$ [74]. As a result, the applications of these methods have so far been limited to the low-dimensional output spaces $m \leq 3$. Additionally, as different polytope formulations require different LP formulations, the implementations of these methods are often somewhat specific to their respective applications.

There are several examples in the literature where the approximation based methods were used for improving the efficiency of different computationally expensive polytope evaluation

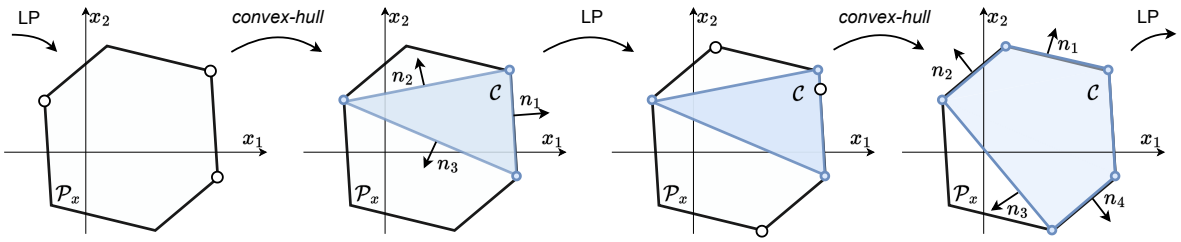

FIGURE 3.14: This figure shows the procedure of successive approximation of the polytope $\mathcal{P}_x$ using the CHM algorithm. The face normal vectors $\boldsymbol{n}_i$ of the Convex-Hull $\mathcal{C}$ are used with LP to find new vertices which are then used to update the Convex-Hull $\mathcal{C}$ and furthermore improve the approximation of the polytope.

problems. Bretl and Lall [151] have proposed a CHM based algorithm for approximating 2D ($m = 2$) polytopes in the context of legged robot locomotion. Del Prete *et al.* [152] recently proposed an efficiency improvement of this algorithm, and Audren and Kheddar [153] extended it to the 3D ($m = 3$) use-cases. This algorithm calculates the inner and outer approximation of the polytope $\mathcal{P}_x$ at the same time, while its accuracy condition can be set a as desired ratio between the inner and outer approximation volumes. Ponce and Faverjon [154] have introduced a derivative of the CHM algorithm to calculate the 2D, 3D and 4D ($m = 2, 3, 4$) polytopes in the context of grasping objects with multiple fingers. While Xu *et al.* [155] used a CHM algorithm in the context in the context of information theory. Both of these algorithms find all vertices and faces of the output polytope $\mathcal{P}_x$, without exploiting CHM's approximation capacity. Furthermore, Carmichael and Liu [61, 108] used an RSM algorithm for approximating 2D and 3D ($m = 2, 3$) polytopes in the context of human force capacity estimation, based on a musculoskeletal model. In order to enable real-time execution, their RSM implementation relies on the uniform sampling in the output space.

In summary, when the complexity of the polytope $\mathcal{P}_x$ transformation does not permit using standard exact methods in the practical applications, especially when the dimension n of the input space is high while the dimension of the output space m is reasonably low $m \leq 3$, polytope approximation methods, such as RSM and CHM algorithms, provide a more efficient solutions. The RSM algorithms provide a fast sampling based approximation, however without any guarantees on the approximation error or accuracy. On the other hand, CHM algorithms use an efficient iterative approach to the polytope approximation, while at the same time being capable of guaranteeing the maximal approximation error.

3.3.5 Synthesis of transformation strategies

Section 3.3 brings an overview of the standard polytope evaluation strategies applicable to finding the $\mathcal{H}$ and $\mathcal{V}$-representation of polytopes with the projection and the intersection formulation. For each one of the formulations, several special cases of the input space $\mathcal{P}_y$ are considered and the appropriate methods are described. The overview proposes both standard exact polytope evaluation methods (Sections 3.3.1 to 3.3.3) and polytope approximation methods (Section 3.3.4). Additionally, a brief discussion on their computational complexity is given as well.

The aim of this overview is to serve as a guide for finding an appropriate polytope evaluation strategy given the polytope $\mathcal{P}_x$ formulation, the shape of the input set $\mathcal{P}_y$ and provide an insight into their time-efficiency. As a visual tool, this section proposes a condensed view of the overview in the form of Table 3.2. The table groups the discussed state of the art methods

Formulation	Generic (Unified) $A\boldsymbol{x} = B\boldsymbol{y} + \boldsymbol{b}$, $\boldsymbol{y} \in \mathcal{P}_y$ Equation (3.3)		Projection (Proj.) $\boldsymbol{x} = B\boldsymbol{y} + \boldsymbol{b}_x$, $\boldsymbol{y} \in \mathcal{P}_y$ Equation (3.6)								Intersection (Int.) $A\boldsymbol{x} = \boldsymbol{y} + \boldsymbol{b}_y$, $\boldsymbol{y} \in \mathcal{P}_y$ Equation (3.11)							
Finding representation	$\mathcal{H}$	$\mathcal{V}$	$\mathcal{H}$			$\mathcal{V}$					$\mathcal{V}$			$\mathcal{H}$				
Input set $\mathcal{P}_y$ special cases	-		$\mathcal{I}_y$	$\mathcal{H}_y$	other	$\mathcal{I}_y$	Proj.	$\mathcal{V}_y$	Int.	other	$\mathcal{I}_y$	other	$\mathcal{H}_y$	Int.	$\mathcal{I}_y$	$\mathcal{H}_y$	Proj.	other
State of the art resolution strategies																		
Overview Section	-		Section 3.3.3.2			Section 3.3.3.1					Section 3.3.2.2			Section 3.3.2.1				
Standard exact Methods	N/A		HPSM [147]	FME [141], ESP [144]	N/A	Section 3.3.1 ex. DDM [129]			N/A		Chiacchio [35], Sasaki [106]	N/A	Section 3.3.1 ex. DDM [129]				N/A	
Approximation Methods	RSM [61]		Section 3.3.4															
Proposed algorithms																		
VEPOLI^2 Section 3.4											VEPOLI^2							
ICHM Section 3.5	ICHM																	

TABLE 3.2: This table brings a condensed view of the overview of the standard polytope evaluation methods applicable to different polytope formulations. The table brings methods for projection and intersection formulations and their generic unified form. As described in the overview different standard methods are often defined for finding $\mathcal{H}$ or $\mathcal{V}$-representation for one of the polytope formulations. Furthermore, the methods are often specific to one spacial case of the input set $\mathcal{P}_y$: they might require its Interval $\mathcal{I}_y$, Projection (Proj.) or Intersection (Int.) form, or in some cases its half-plane and vertex representation $\mathcal{H}_y$ or $\mathcal{V}_y$. The table shows that there are several cases where the standard methods are not directly applicable (N/A), but might require a sequence of several methods in order to be evaluated. The table also shows the applicability (red cells) of the algorithms VEPOLI^2 and ICHM introduced in Section 3.4 and Section 3.5 respectively.

with respect to their direct applicability to different evaluation cases. If the polytope evaluation case does not have any method directly applicable, it is denoted with N/A (Not Applicable).

The table demonstrates that there are several polytope evaluation cases where the standard exact methods are not directly applicable, or potentially require several methods used sequentially. On the other hand, the approximation methods are applicable to the transformation of both the intersection and the projection formulations. Moreover, in the case of the generic formulation (3.3), to the best of our knowledge, only the Ray Shooting Method (RSM) method by Carmichael and Liu [61] has been applied.

The following two sections introduce two new polytope evaluation algorithms called VEPOLI^2 (Section 3.4) and ICHM (Section 3.5). Both algorithms are added to Table 3.2, showing their applicability to different polytope evaluation problems.

3.4 VEPOLI^2: New vertex finding algorithm for intersection formulation

One typical example of the polytope with the intersection formulation (3.11) and interval input set (3.4) is the feasible wrench polytope, discussed in Section 2.1.5,

$$\mathcal{P}_f(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) = \left\{ \boldsymbol{f} \in \mathbb{R}^m \mid \boldsymbol{\tau} \in [\boldsymbol{\tau}_{min}, \boldsymbol{\tau}_{max}] \,, \quad J^T(\boldsymbol{q})\boldsymbol{f} = \boldsymbol{\tau} - \boldsymbol{\tau}_b(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) \right\} \tag{3.52}$$

where the input space corresponds to the space of applicable joint torques $\boldsymbol{\tau} \in \mathbb{R}^n$, the output space is the set feasible Cartesian Space (CS) wrenches (forces) $\boldsymbol{f} \in \mathbb{R}^m$ and the mapping between the two is given through the Jacobian transpose matrix $J(\boldsymbol{q})^T \in \mathbb{R}^{n \times m}$. Additionally, the bias vector $\boldsymbol{\tau}_b$ groups the influences of robot's motion and gravity.

Being able to characterise the robot's wrench generation capacity exactly, as in the case of the polytope (3.52), has a potential to enable more adapted robot control strategies leveraging this fine information about robot's true capacity. As the polytope $\mathcal{P}_f$ is state dependant, its shape and the size vary significantly with respect to the robot's state $\{\boldsymbol{q}, \dot{\boldsymbol{q}}\}$ and potentially desired acceleration $\ddot{\boldsymbol{q}}$ to be achieved. Therefore, when it comes to using the polytope $\mathcal{P}_f$ in real-time robot control applications, where the robot exchanges wrenches with different objects and tools in the environment, the wrench polytope $\mathcal{P}_f$ has to be evaluated in real-time too.

Typical robot control strategies require the cycle times in the range of a few milliseconds in order to ensure satisfactory results. Therefore, to integrate the wrench polytopes $\mathcal{P}_f$ in the robot control applications, the polytope $\mathcal{P}_f$ needs to be transformed to a suitable representation ($\mathcal{H}$ or $\mathcal{V}$) in comparable times as well.

In a generic form a robot's wrench capacity polytope $\mathcal{P}_x$ can be expressed as

$$\mathcal{P}_x = \{\boldsymbol{x} \in \mathbb{R}^m \mid \boldsymbol{y} \in [\boldsymbol{y}_{min}, \boldsymbol{y}_{max}], \; A\boldsymbol{x} = \boldsymbol{y}\} \tag{3.53}$$

where the matrix $A \in \mathbb{R}^{n \times m}$ corresponds to the Jacobian transpose $J(\boldsymbol{q})^T$ and $\boldsymbol{x} \in \mathbb{R}^m$ corresponds to the CS wrench $\boldsymbol{f}$, while $\boldsymbol{y} \in \mathbb{R}^n$, without the loss of generality, corresponds to the achievable joint torques $\boldsymbol{\tau}$ reduced by the fixed bias $\boldsymbol{\tau} - \boldsymbol{\tau}_b$. Additionally, the input set can be expressed in the interval form

$$\mathcal{I}_y = \{\boldsymbol{y} \in \mathbb{R}^n \mid \boldsymbol{y} \in [\boldsymbol{y}_{min}, \boldsymbol{y}_{max}] \; \} \tag{3.54}$$

As discussed in Section 3.3.2.1, the $\mathcal{H}$-representation of the intersection formulation based polytopes comes directly from their definition and is easy to obtain. Therefore, in this section

the accent is put on transforming the polytope $\mathcal{P}_f$ to its $\mathcal{V}$-representation. As discussed in Section 3.3.2.2, there are several algorithms introduced in the literature that enable efficient vertex enumeration of intersection polytopes $\mathcal{P}_x$ with interval input set $\mathcal{I}_y$. Two of such examples are the algorithms introduced by Chiacchio *et al.* [35] and Sasaki [106].

In this section, a new efficient algorithm for finding the $\mathcal{V}$-representation of this polytope formulation is presented, building upon the findings from the algorithm proposed by Sasaki [106]. The proposed algorithm reduces the complexity of the exhaustive search for the vertices of the polytope $\mathcal{P}_x$ by exploiting the geometry of the problem. The algorithm is compared, Section 3.4.3, against the the state of the art algorithms from Chiacchio *et al.* [35] and Sasaki [106] and shown to have lower complexity and shorter execution time. Furthermore, the results show that the proposed algorithm is capable of finding the $\mathcal{V}$-representation of this family of polytopes within a few milliseconds, for problem sizes equivalent to those of evaluating the wrench capacity (3.52) of standard robotic manipulators, opening doors for potential applications in real-time applications.

The algorithm is named VEPOLI^2, standing for **V**ertex **E**numeration algorithm for **POL**ytopes with **I**ntersection formulation and **I**nterval limits.

3.4.1 Problem definition

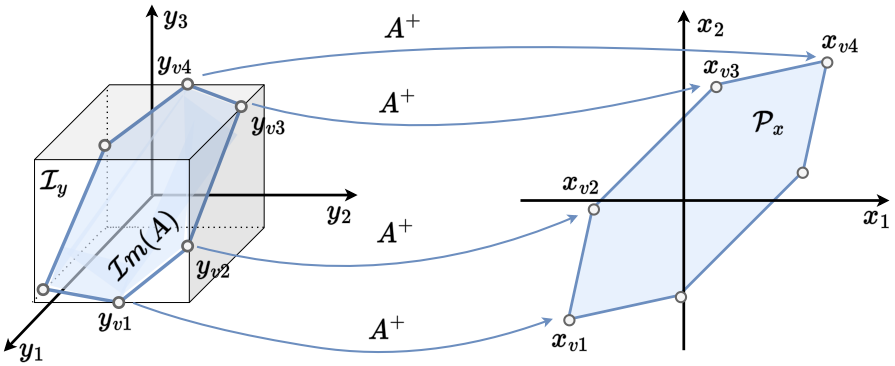

FIGURE 3.15: Example intersection polytope $\mathcal{P}_x$ with $n = 3$ dimensional interval input set $\mathcal{I}_y$ and $m = 2$ dimensional output space. The polytope $\mathcal{P}_x$ vertices $\boldsymbol{x}_{vi}$ correspond to the vertices of the intersection $\mathcal{I}_y \cap \mathcal{I}m\,(A)$ denoted as $\boldsymbol{y}_{vi}$, through the pseudo-inverse A^T of the matrix A.

As shown on Figure 3.15, the space of all the input vectors $\boldsymbol{y} \in \mathbb{R}^n$, bounded within the interval shaped input set $\mathcal{I}_y$, geometrically forms an n-dimensional hyperrectangle (orthotope) with n pairs of parallel sides defined by the output limits $\boldsymbol{y} \in [\boldsymbol{y}_{min},\ \boldsymbol{y}_{max}]$. The image of the A matrix $\mathcal{I}m\,(A)$ is a r-dimensional subspace of the hyperrectangle, where r is the rank of A. In the remainder of this section, the matrix A is, without loss of generality, assumed to be full column rank, i.e. $m{=}r$.

As discussed in Section 3.2.2, the output space polytope $\mathcal{P}_x$ is the direct affine projection of the intersection between the image $\mathcal{I}m\,(A)$ and the input set $\mathcal{I}_y$ using the pseudo-inverse A^+ of the matrix A. Therefore this algorithm aims to efficiently find the vertices $\boldsymbol{x}_{vi}$ of the polytope $\mathcal{P}_x$, by efficiently finding the vertices $\boldsymbol{y}_{vi}$ of the intersection $\mathcal{I} \cap \mathcal{I}m\,(A)$, and project them to the output space using the relationship

$$\boldsymbol{x}_{vi} = A^+ \boldsymbol{y}_{vi}, \qquad \text{where} \quad \boldsymbol{y}_{vi} \in \mathcal{I}_y \cap \mathcal{I}m\,(A) \tag{3.55}$$

Therefore the operation of enumerating all the vertices $\boldsymbol{x}_{vi}$ of the polytope $\mathcal{P}_x$ can be seen as first finding the $\mathcal{V}$-representation (the set of vertices $\boldsymbol{y}_{vi}$) of the intersection $\mathcal{I} \cap \mathcal{I}m\,(A)$,

followed by its projection to the output space through the pseudo-inverse A^+, to obtain the $\mathcal{V}$-representation of the polytope $\mathcal{P}_x$

$$\mathcal{P}_x = \text{Conv}\left(A^+\boldsymbol{y}_{v1},\ A^+\boldsymbol{y}_{v1},\ \dots\right) \tag{3.56}$$

In their work, Sasaki [106] have shown that, when the input space dimension is n and the image $\mathcal{I}m\,(A)$ dimension is m, the extreme values (vertices) of the intersection $\boldsymbol{y}_{vi}$ belong to the $n-m$ dimensional faces of the hyperrectangle $\mathcal{I}_y$. Figure 3.16 demonstrates this relationship on an example with the input space dimension $n=3$ and two different output space dimensions $m=1$ and $m=2$. For $m=1$ the vertices $\boldsymbol{y}_{vi}$ are on $m-n=2$ dimensional faces of the hyperrectangle corresponding to its sides, and when $m=2$ the vertices $\boldsymbol{y}_{vi}$ are on $m-n=1$ dimensional faces of the hyperrectangle, its edges.

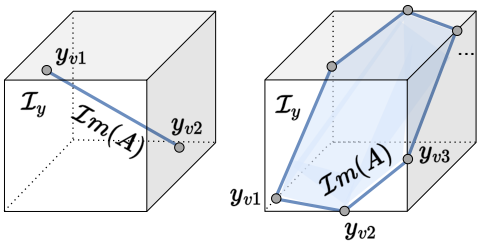

FIGURE 3.16: Example of intersection vertices $\boldsymbol{y}_{vi}$ placement on $n-m$ dimensional faces of hyperrectangle $\mathcal{I}_y$. Left, $m=1$, the vertices $\boldsymbol{y}_{vi}$ are on $n-m=2$ dimensional faces (sides), while on the right $m=2$ the vertices are on $n-m=1$ dimensional faces (edges)

In order to find the $\mathcal{V}$-representation of the intersection $\mathcal{I}\cap\mathcal{I}m\,(A)$, the state-of-the-art algorithms, proposed by Chiacchio *et al.* [35] and Sasaki [106], propose an exhaustive search over all the n-m dimensional hyperrectangle faces to find the extreme values of $\boldsymbol{y}_{vi}$. In this work a new representation of output vector $\boldsymbol{y}$ is proposed enabling an efficient navigation of the $n-m$ dimensional hyperrectangle faces. Additionally, an efficient approach to discarding the hyperrectangle faces that cannot contain vertices is integrated in the algorithm, further reducing the complexity of the exhaustive search.

3.4.2 Proposed VEPOLI^2 algorithm

Consider an input vector $\boldsymbol{y}$ bounded within the input set $\mathcal{I}_y$. It can be defined as

$$\boldsymbol{y} = \boldsymbol{y}_{min} + \alpha_1\boldsymbol{y}_1 + \alpha_2\boldsymbol{y}_2 + \ldots + \alpha_n\boldsymbol{y}_n \tag{3.57}$$

where $\alpha_i \in [0,1]$ are scalars weights and vectors $\boldsymbol{y}_i$ are orthogonal base vectors in Joint Space (JS)aligned with i-th axis of the hyperrectangle $\mathcal{I}_y$, defined as $\boldsymbol{y}_i = \begin{bmatrix}0 & \dots & y_{i,max} - y_{i,min} & \dots & 0\end{bmatrix}^T$. As shown on the example on Figure 3.17, the input vector $\boldsymbol{y}$ representation (3.57) can be exploited to reach any point within the hyperrectangle $\mathcal{I}_y$, by choosing appropriate scalars α_i in range $[0,1]$. Finding the appropriate values of the scalars α_i, for any given $\boldsymbol{y}$, can be formulated in a form of linear system

$$\underbrace{\begin{bmatrix}\boldsymbol{y}_1 & \cdots & \boldsymbol{y}_n\end{bmatrix}}_{Y_{n\times n}} \underbrace{\begin{bmatrix}\alpha_1 \\ \cdots \\ \alpha_n\end{bmatrix}}_{\boldsymbol{\alpha}_{n\times 1}} = \boldsymbol{y}_{min} - \boldsymbol{y}, \qquad \boldsymbol{\alpha} = Y^{-1}(\boldsymbol{y}_{min} - \boldsymbol{y}) \tag{3.58}$$

where the matrix Y, containing the orthogonal base vectors $\boldsymbol{y}_i$, is square and always invertible. Furthermore, as the intersection vertices $\boldsymbol{y}_{vi}$ belong to the n-m dimensional faces of the input hyperrectangle $\mathcal{I}_y$, the representation (3.57) can be further simplified. Any input vector $\boldsymbol{y}$, belonging to an $n-m$ dimensional face of the hyperrectangle $\mathcal{I}_y$, can be reached by choosing the appropriate values of $n-m$ scalars $\alpha_i \in [0,1]$, while the other m scalars are fixed to either 0 or 1 ($\alpha_i \in \{0,1\}$).

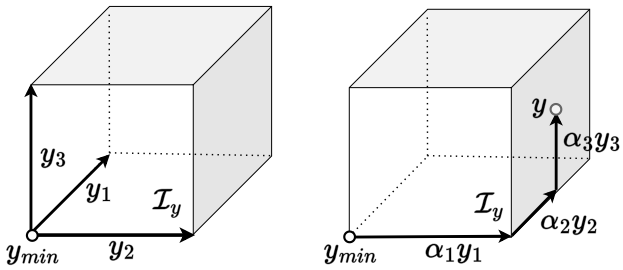

FIGURE 3.17: Example of the vector representation (3.57) on the $n=3$ dimensional hyperrectangle $\mathcal{I}_y$. Base vectors $\boldsymbol{y}_i$ are shown on the left, and on the right their linear combination using scalars α_i is shown reaching the point $\boldsymbol{y} \in \mathcal{I}_y$.

Furthermore, $n-m$ scalars $\alpha_i \in [0,1]$ can be interpreted as coordinates of the $n-m$ dimensional face, while the remaining m fixed scalars $\alpha_i \in \{0,1\}$, to either 0 or 1, define its origin $\boldsymbol{y}_o$

$$\boldsymbol{y}_o = \boldsymbol{y}_{min} + \alpha_1 \boldsymbol{y}_1 + ... + \alpha_m \boldsymbol{y}_m, \qquad \alpha_i \in \{0,1\} \tag{3.59}$$

Then any $\boldsymbol{y}$, belonging to a face of the hyperrectangle, can be expressed using the face's origin $\boldsymbol{y}_o$ and its set of $n-m$ scalars α_i

$$\boldsymbol{y} = \boldsymbol{y}_o \; + \; \alpha_{m+1}\boldsymbol{y}_{m+1} + \; \cdots \; + \alpha_n \boldsymbol{y}_n, \qquad \alpha_i \in [0,1] \tag{3.60}$$

The vector $\boldsymbol{\alpha} \in \mathbb{R}^n$ can be conveniently divided into two components, $\boldsymbol{\alpha}_o \in \mathbb{R}^m$ containing m scalars α_i defining the origin of the face and $\boldsymbol{\alpha}_x \in \mathbb{R}^{n-m}$ containing $n-m$ scalar coordinates α_i of the face. Additionally, the same can be done with the matrix Y, divided in $Y_o \in \mathbb{R}^{n \times m}$ corresponding to m base vectors used to define the origin $\boldsymbol{y}_o$, and $Y_x \in \mathbb{R}^{n \times (n-m)}$ corresponding to the $n-m$ base vectors $\boldsymbol{y}_i$ spanning the face. Then the origin $\boldsymbol{y}_o$ and the final vector $\boldsymbol{y}$ can be expressed as

$$\boldsymbol{y}_o = \boldsymbol{y}_{min} + Y_o \boldsymbol{\alpha}_o, \qquad \boldsymbol{y} = \boldsymbol{y}_o + Y_x \boldsymbol{\alpha}_x, \qquad \boldsymbol{\alpha}_o \in \{\mathbf{0},\mathbf{1}\}, \quad \boldsymbol{\alpha}_x \in [\mathbf{0},\mathbf{1}] \tag{3.61}$$

Therefore, in order to find all the vertices $\boldsymbol{y}_{vi}$ of the intersection $\mathcal{I}_y \cap \mathcal{I}m\,(A)$, the proposed VEPOLI^2 algorithm tests if there exists a point intersection (a vertex) between the $\mathcal{I}m\,(A)$ and every single one of the $n-m$ dimensional faces of the hyperrectangle $\mathcal{I}_y$.

Since all the $\boldsymbol{y}$ belonging to the image $\mathcal{I}m\,(A)$ can be expressed as $\boldsymbol{y} = A\boldsymbol{x}$, this relationship can be further exploited to relate the vertices $\boldsymbol{y}_{vi}$ of the intersection $\mathcal{I}m\,(A) \cap \mathcal{I}_y$ and the vertices $\boldsymbol{x}_{vi}$ of the final polytope $\mathcal{P}_x$. For each face to be tested, the equations (3.53) and (3.55) can be combined

$$\boldsymbol{y}_{vi} = A\boldsymbol{x}_{vi} = \boldsymbol{y}_{oi} + \alpha_{m+1}\boldsymbol{y}_{m+1} + \; ... \; + \alpha_n \boldsymbol{y}_n \tag{3.62}$$

where, $\boldsymbol{y}_{oi}$ is the origin of the $n-m$ dimensional face, spanned by the scalar $n-m$ scalars $\boldsymbol{\alpha}_{xi}$. For each face i, by fixing the appropriate m scalars $\boldsymbol{\alpha}_{oi}$ to 0 or 1, its origin $\boldsymbol{y}_{oi}$ can be calculated using (3.61). In order to determine if this face contains the vertex of the intersection $\mathcal{I}_y \cap \mathcal{I}m\,(A)$, a linear system derived from equation (3.62) can be solved. This system finds the remaining $n-m$ scalars $\boldsymbol{\alpha}_{xi}$ as well as the corresponding vertex $\boldsymbol{x}_{vi}$ at the same time

$$\underbrace{\begin{bmatrix} A & -\boldsymbol{y}_{m+1} \; \ldots \; -\boldsymbol{y}_n \end{bmatrix}}_{Z_{n\times n}} \begin{bmatrix} \boldsymbol{x}_{vi} \\ \alpha_{m+1} \\ \vdots \\ \alpha_n \end{bmatrix} = \begin{bmatrix} A & Y_{xi} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_{vi} \\ \boldsymbol{\alpha}_{xi} \end{bmatrix} = \boldsymbol{y}_{oi}, \qquad \begin{bmatrix} \boldsymbol{x}_{vi} \\ \boldsymbol{\alpha}_{xi} \end{bmatrix} = Z^{-1}\boldsymbol{y}_{oi}, \tag{3.63}$$

Matrix $Z \in \mathbb{R}^{n\times n}$ is a square matrix where each one of the vectors $\boldsymbol{y}_i$ is independent. If the matrix Z is singular, geometrically, the image $\mathcal{I}m\,(A)$ is parallel to the face being tested, therefore no intersections exist. If Z is invertible and the $n-m$ scalars $\boldsymbol{\alpha}_{xi}$ are obtained, a simple check if $\boldsymbol{\alpha}_{xi} \in [\mathbf{0},\mathbf{1}]$ can be used to determine if $\boldsymbol{x}_{vi}$ is a vertex of the polytope $\mathcal{P}_x$. Geometrically, the condition $\boldsymbol{\alpha}_{xi} \in [\mathbf{0},\mathbf{1}]$ ensures that the intersection point between the $\mathcal{I}m\,(A)$ and the $n-m$ dimensional face, is also inside $\mathcal{I}_y$. One of the nice features of this approach is that it does not require an explicit inversion of A, as in equation (3.55), to obtain the set of vertices composing $\mathcal{P}_x$.

Following three sections bring the approaches further improving the efficiency of the proposed algorithm. First, the dimension of the matrix Z is reduced using the Singular Value Decomposition (SVD), then the parallelism of the faces of the hyperrectangle is exported to reduce the number of the computations of the system (3.63). Finally, an additional condition is introduced enabling to discard faces and avoid the unnecessary matrix inversions.

3.4.2.1 Matrix size reduction using the SVD

In order to reduce the dimension of the linear system (3.63), the SVD [156] of A^T is used

$$A^T = U \underbrace{\begin{bmatrix} S & O_{m\times(n-m)} \end{bmatrix}}_{\Sigma} \underbrace{\begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix}}_{V^T} \tag{3.64}$$

where $S = diag(\sigma_1 \dots \sigma_m)$ is a diagonal matrix containing the m singular values of A^T. $V_1^T \in \mathbb{R}^{m\times n}$ is the projector from the input space onto the image of $\mathcal{I}m\,(A)$ and $V_2^T \in \mathbb{R}^{n-m\times n}$ is a basis of $\mathcal{K}er\left(A^T\right)$. V_1^T and V_2^T are orthogonal vector subspaces yielding $V_2^T V_1 = 0$ and thus

$$V_2^T A\boldsymbol{x} = \mathbf{0} \tag{3.65}$$

Then, multiplying the equation (3.63) with V_2^T

$$V_2^T \begin{bmatrix} A & Y_{xi} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_{vi} \\ \boldsymbol{\alpha}_{xi} \end{bmatrix} = \underbrace{V_2^T A\boldsymbol{x}_{vi}}_{\mathbf{0}} + V_2^T Y_{xi}\boldsymbol{\alpha}_{xi} = V_2^T \boldsymbol{y}_{oi} \tag{3.66}$$

allows to reduce the system (3.63) to

$$\underbrace{V_2^T \begin{bmatrix} -\boldsymbol{y}_{m+1} & \dots & -\boldsymbol{y}_n \end{bmatrix}}_{T_{(n-m)\times(n-m)}} \boldsymbol{\alpha}_x = V_2^T Y_{xi}\boldsymbol{\alpha}_{xi} = V_2^T \boldsymbol{y}_{oi} \tag{3.67}$$

If T is invertible, $\boldsymbol{\alpha}_{xi}$ can be computed by

$$\boldsymbol{\alpha}_{xi} = T^{-1} V_2^T \boldsymbol{y}_{oi} \tag{3.68}$$

When the computed $\boldsymbol{\alpha}_{xi}$ complies with the check $\boldsymbol{\alpha}_{xi} \in [\mathbf{0},\mathbf{1}]$, the corresponding polytope vertex can be calculated as

$$\boldsymbol{x}_{vi} = A^+\left(\boldsymbol{y}_{oi} + Y_{xi}\boldsymbol{\alpha}_{xi}\right) \tag{3.69}$$

This new approach requires the calculation of the SVD and the pseudo-inverse A^+. Nevertheless, since the pseudo-inverse can be efficiently calculated from the SVD as $A^+ = U\Sigma^{T+}V^T$ and since both the SVD and pseudo-inverse are calculated once per algorithm run, the computation efficiency is greatly improved due to the matrix dimension reduction from n to $(n-m)$ when inverting T instead of Z.

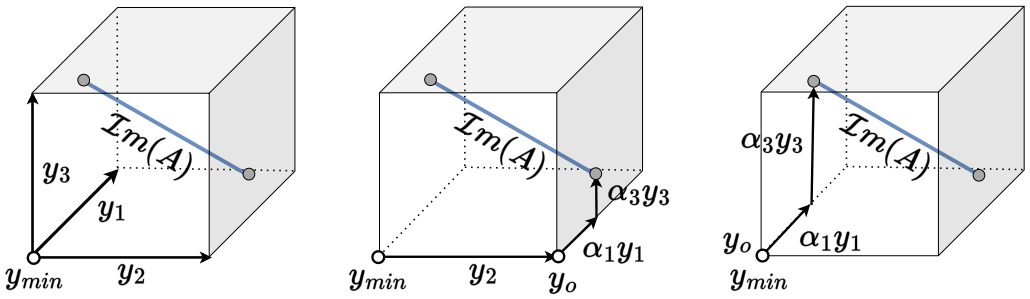

FIGURE 3.18: Example of input space interpretation of the VEPOLI^2 algorithm for the system n=3 and m=1. The two vertices of the intersection $\{\boldsymbol{y}_{v1}, \boldsymbol{y}_{v2}\}$ are reached by solving the linear system (3.63) or (3.67). The vertices belong to the parallel ($n-m=2$ dimensional faces) sides of the hyperrectangle $\mathcal{I}_y$.

3.4.2.2 Exploiting the parallelism of the hyperrectangle faces

The number of $n-m$ dimensional faces of an n dimensional hyperrectangle $\mathcal{I}_y$ can be calculated using the expression

$$N_{n,(n-m)} = 2^{n-(n-m)} \binom{n}{n-m} = 2^m \binom{n}{m}$$

More precisely, an n dimensional hyperrectangle has $\binom{n}{m}$ sets of 2^m parallel faces with the dimension n–m. As illustrated on Figure 3.19, a 3D ($n=3$) hyperrectangle has $\binom{3}{1}=3$ sets of $2^2=4$ parallel ($n-m=1$) edges, as well as $\binom{3}{2}=3$ sets of $2^1=2$ parallel ($n-m=2$) sides.

All the parallel n–m dimensional faces of the hyperrectangle $\mathcal{I}_y$ can be spanned using the same set of $n-m$ scalars $\boldsymbol{\alpha}_{xi}$, while the remaining m scalars $\boldsymbol{\alpha}_{oi}$ can be used to find their 2^m origins $\boldsymbol{y}_{oi}$.

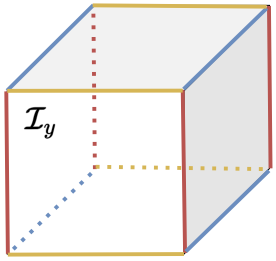

FIGURE 3.19: All the parallel edges of the 3D hyperrectangle shown in differing colours. It has 3 (red, blue, yellow) sets of 4 parallel edges, 12 edges in total.

Figure 3.18 illustrates the case where $\mathcal{I}m(A)$ is a line ($m=1$) and the input space is $n=3$ dimensional. In this case the intersection vertices $\{\boldsymbol{y}_{v1}, \boldsymbol{y}_{v2}\}$ lie in n-m=2 dimensional hyperrectangle faces. In that example, both vertices $\boldsymbol{y}_{vi}$ are placed on two parallel faces, spanned by the two scalars α_1 and α_3, while the scalar α_2, set to 0 or 1, is used to find their origins $\boldsymbol{y}_{oi}$.

In the linear system (3.67), matrix T has the same content for each parallel face of the hyperrectangle. Therefore, the inverse T^{-1} can be recalculated only once per set of parallel faces. Then, for each of the 2^m parallel faces within the set, the vertex $\boldsymbol{x}_{vi}$ and the corresponding $\boldsymbol{\alpha}_{xi}$, can be obtained by multiplying the 2^n origins $\boldsymbol{y}_{oi}$ with T^{-1}, as described by (3.68-3.69).

Overall, in order to test all possible combinations of the n–m dimensional faces which may contain vertices of the intersection (and in term the final polytope $\mathcal{P}_x$), the total of $\binom{n}{m}=\frac{n!}{m!(n-m)!}$ inversions of the matrix T and $2^m\binom{n}{m}$ checks $\boldsymbol{\alpha}_{xi} \in [\mathbf{0}, \mathbf{1}]$ have to be performed.

In the case depicted in Figure 3.18 (n=3, m=1), T is inverted only $\binom{3}{1}$=3 times and conditions are evaluated 6 times, which corresponds exactly to the number of sides of the hyperrectangle. In the same example, if the output space would be 2D (n=3, m=2) the number of T inversions is 3 and the number of condition evaluation is 12, which corresponds to the number of edges of the hyperrectangle.

3.4.2.3 Matrix inverse condition

The most time consuming aspect of the proposed VEPOLI^2 algorithm is the relatively high number of matrix T inversions. Therefore this section proposes a necessary condition to discard the sets of faces that cannot contain a vertex and avoid unnecessary matrix T inversions.

Due to the constrained nature of the scalars $\boldsymbol{\alpha}_{xi} \in [\mathbf{0}, \mathbf{1}]$, it is possible to efficiently calculate the bounds ($\boldsymbol{t}_{ub}$ and $\boldsymbol{t}_{lb}$) on the $T\boldsymbol{\alpha}_{xi}$

$$T\boldsymbol{\alpha}_{xi} \in [\boldsymbol{t}_{lb}, \boldsymbol{t}_{ub}] \tag{3.70}$$

As the column vectors $\boldsymbol{t}_i$ within the matrix T are, in the general case not orthogonal, the min-max interval shaped set can not exactly characterise the feasible set of $T\boldsymbol{\alpha}_{xi}$. However, an interval that over-approximates it, can be found very efficiently by row-wise summing only positive and only negative elements t_{ij} of T

$$t_{i,lb} = \sum_j \max(t_{ij}, 0), \qquad t_{i,lb} = \sum_j \min(t_{ij}, 0) \tag{3.71}$$

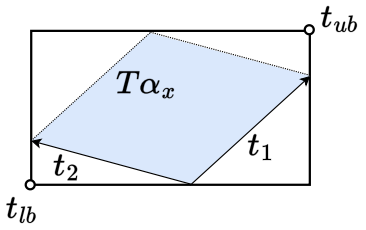

FIGURE 3.20: The bounding-box calculated using the equation (3.71). The size of the example system is $n - m = 2$, where $\boldsymbol{t}_1, \boldsymbol{t}_2$ are column vectors of the matrix T.

As shown on Figure 3.20, this creates a bounding box around the space defined by $T\boldsymbol{\alpha}_{xi}$.

In order for the system (3.67) to have a solution, the null-space projections of the vectors $\boldsymbol{y}_{oi}$ have to respect the same bounds as well

$$V_2^T \boldsymbol{y}_{oi} \in [\boldsymbol{t}_{lb}, \boldsymbol{t}_{ub}] \tag{3.72}$$

Therefore, using this bounding box, the necessary check can be devised for the inversion of the matrix T. If all the 2^m combinations of $V_2^T \boldsymbol{y}_{oi} \notin [\boldsymbol{t}_{lb}, \boldsymbol{t}_{ub}]$, the system (3.67) cannot have a solution which satisfies $\boldsymbol{\alpha}_{xi} \in [\mathbf{0}, \mathbf{1}]$, and there is no need to invert T to figure it out.

The condition $V_2^T \boldsymbol{y}_{oi} \in [\boldsymbol{t}_{lb}, \boldsymbol{t}_{ub}]$ is a necessary condition, however as the interval $[\boldsymbol{t}_{lb}, \boldsymbol{t}_{ub}]$ presents an over-approximation of the achievable set $T\boldsymbol{\alpha}_x, \boldsymbol{\alpha}_x \in [\mathbf{0}, \mathbf{1}]$, it is not a sufficient condidion.

The bounding box (3.71), as well as the 2^m face origins $\boldsymbol{y}_{oi}$ can be calculated efficiently, allowing for a computationally inexpensive reduction in the number of matrix inversions. The upper bound on the number of matrix inversions N_{inv} can be found by studying the worst-case scenario, where all the origins $\boldsymbol{y}_{oi}$ pass the condition (3.72). In this case the number of inversion corresponds to the binomial $\binom{n}{m}$. In the general case, the number of matrix inversions N_i is bounded and equals to

$$N_{inv} \leq \binom{n}{m} = \frac{n!}{m!(n-m)!}$$

The number of checks N_c of the condition $\boldsymbol{\alpha}_{xi} \in [\mathbf{0}, \mathbf{1}]$, per matrix inversion, is bounded as well, as the origin vectors $\boldsymbol{y}_{oi}$ that do not comply with the necessary condition (3.72) can be discarded from the these checks as well

$$N_c \leq 2^m$$

3.4.3 Performance and complexity comparison

To demonstrate the efficiency of the VEPOLI^2 algorithm, it is compared against the polytope vertex search algorithm introduced by Chiacchio *et al.* [35]. Furthermore the comparison is extended to the algorithm proposed by Sasaki [106] which is, to our knowledge, the only

algorithm exploiting the geometric structure of the intersection formulation polytope with interval limits (3.53).

System size	**Chiacchio**[35]	**Sasaki** [106]	**VEPOLI**^{2}
(n, m)	matrix inversions: $\frac{2n!}{m!(2n-m)!}$ matrix size: $2n \times 2n$ time[ms]: mean $\pm$ std (max)	$\frac{n!}{m!(n-m)!}$ $m \times m$	$\leq \frac{n!}{m!(n-m)!}$ $n\text{-}m \times n\text{-}m$
$(4, 2)$	24 8×8 2.7±0.1 (3.6)	6 2×2 0.93±0.1 (1.1)	4.2±1.4 (6) 2×2 0.64±0.1 (1.5)
$(4, 3)$	32 8×8 3.7±0.3 (6.0)	4 3×3 0.79±0.08 (1.4)	2.9±1.1 (4) 1×1 0.54±0.1 (1.4)
$(6, 3)$	220 12×12 14.2±0.9 (20)	20 3×3 2.1±0.2 (3.4)	2.8±2.5 (10) 3×3 1.5±0.2 (2.7)
$(6, 6)$	64 12×12 34.7±1 (44.3)	1 6×6 0.32±0.07 (0.56)	1±0 (1) 6×6 0.24±0.04 (0.46)
$(7, 3)$	364 14×14 25±0.5 (27)	35 3×3 3.5±0.1 (5.4)	7.3±4.2 (20) 4×4 2.6±0.2 (4.3)
$(7, 6)$	448 14×14 150±34 (360)	7 6×6 1.6±0.5 (4)	6.4±1 (7) 1×1 1.2±0.3 (3.0)

TABLE 3.3: Complexity and execution time (in milliseconds) comparison for three different vertex search algorithms. The comparison provided for the number of matrix inversions, matrix size and time of execution, averaged over 1000 random systems.

The three algorithms have been tested on finding the $\mathcal{V}$-representation of the three different systems, inspired by the wrench ($\boldsymbol{f} \in \mathbb{R}^m$) polytope $\mathcal{P}_f$ enumeration for three different robots: a 4R planar robot (n=4), the Universal Robots UR5 6DoF robot (n=6) and the Franka Emika Panda 7DOF robot (n=7). The planar 4R robot is used to calculate its planar force ($m = 2$) and its planar wrench ($m = 3$) polytope, while UR5 and Panda are used to calculate Cartesian Space (CS) force ($m = 3$) and wrench ($m = 6$) polytopes. The results are averaged over 1000 randomly generated matrices A and input sets $\mathcal{I}_y$. All algorithms have been implemented in the programming language Matlab and tested on a laptop equipped with a 1.90GHz Intel i7-8650U processor. The source code of the experiment is publicly available on GitLab[1].

Table 3.3 shows the results of this complexity evaluation. The VEPOLI^2 algorithm substantially reduces the number of matrix inversions and thus reduces the processing time considerably: 4-10× faster execution than Chiacchio's algorithm and on average 30% faster than Sasaki's. Results show that, even for the generic cases that correspond to the 6DOF ($n = 6$) and 7DOF ($n = 7$) industrial robots, the VEPOLI^2 algorithm is capable of evaluating both force ($m = 3$) and wrench ($m = 6$) polytope vertices under $3ms$. Such a low processing time opens numerous opportunities for the online use of polytope based capacity, evaluation especially in the area of robot control.

Figure 3.21 shows the visualisation of the CS force ($m = 3$) polytope for the UR5 and Panda robots. Both polytopes are calculated using the VEPOLI^2 algorithm.

[1]Gitlab: https://gitlab.inria.fr/auctus-team/people/antunskuric/papers/polytope_vertex_search

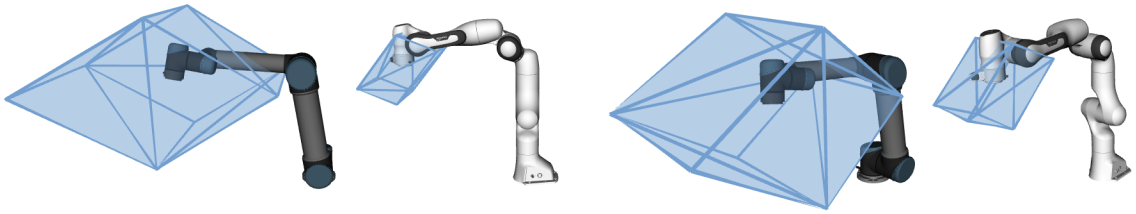

FIGURE 3.21: Two different views of Panda ($n = 7$) and UR5 ($n = 6$) robot's force polytope ($m = 3$) calculated using the VEPOLI^2 algorithm. Both robots and polytopes are shown in the same scale (1m : 1000N). The figure shows that the UR5 has much larger polytope than Panda robot. This is due to its larger force capacity, as UR5 is rated for 5kg loads whereas Panda is rated for 3kg loads.

3.4.4 Discussion on limitations

This algorithm, being based on the algorithm proposed by Sasaki [106], is still based on the exhaustive search. Even though significantly reducing the complexity of the Sasaki's method, it is not particularly well suited for high dimensions n of the input space, especially if the difference between the input and the output space dimension $n-m$ is large.

The algorithm has two computational bottlenecks: the number of matrix T inversions N_{inv} and the number of checks $\boldsymbol{\alpha}_{xi} \in [\mathbf{0}, \mathbf{1}]$ per matrix inversion N_c.

If the system has high dimensional input space and low dimensional output space ($n \gg m$) then the number of matrix inversions $N_{inv} = \binom{n}{m}$ will be large, while the number of checks $N_c = 2^m$ will be low. On the other hand, if the system has both high dimensional input space and high dimensional output space ($n \approx m$ and $n, m \gg 0$), then the number of matrix inversions $N_{inv} = \binom{n}{m}$ will be relatively low and will not be the limiting factor, but the number of checks $N_c = 2^m$ which will be very high. Therefore, although the algorithm is created in a generic form and can find the $\mathcal{V}$-representation of any polytope belonging to this family, it is particularly efficient when the dimensions of the input and the output space are relatively low.

Furthermore, as the VEPOLI^2 algorithm exploits the null-space $\mathcal{K}er\left(A^T\right)$ of the matrix A to gain on efficiency, it is not suitable for the systems where the matrix A is square ($m = n$), as in that case the null-space does not exist $\mathcal{K}er\left(A^T\right) = \{\varnothing\}$. However, as discussed in the Section 3.2.2, this specific case of interval projection is much easier to work with as it corresponds to the projection formulation, which $\mathcal{V}$-representation can be determined in a straight-forward manner, as described in Section 3.3.3.1, by projecting the vertices of the hyperrectangle to the output space.

3.4.5 Algorithm implementation

The pseudo-code of the proposed algorithm is given in Algorithm 1, while an efficient open-source Python implementation is publicly available within the package `pycapacity`[2], which is described more in detail in Chapter 7.

Section 4.2 brings the application of the VEPOLI^2 algorithm for the real-time robot control in the human-robot collaboration scenario, where the algorithm is exploited in order to calculate the robot's wrench capacity polytope online.

`pycapacity`

[2] https://auctus-team.github.io/pycapacity/

Algorithm 1 The VEPOLI^2 algorithm pseudo-code

Require: A, $\boldsymbol{y}_{min}$, $\boldsymbol{y}_{max}$ (Eq. 3.54)
  $U, \Sigma, V^T \leftarrow svd(A^T)$
  $A^+ = U\Sigma^{T+}V^T$
  V_1, $V_2 \leftarrow V$
  calculate n base vectors $\boldsymbol{y}_1 \dots \boldsymbol{y}_n$ (Eq. 3.57)
  init $\mathcal{V}$-rep: X_v, $Y_v \leftarrow []$
  for all $\binom{n}{m}$ combinations of m fixed α_i **do**
    construct matrices Y_o and Y_x
    $T = V_2^T Y_x$
    if matrix T invertible **then**
      find bounds $\boldsymbol{t}_{lb}, \boldsymbol{t}_{ub}$ (Eq. 3.71)
      init list of origins $\boldsymbol{y}_o$ satisfying the condition (Eq. 3.72): $L_o \leftarrow []$
      for all 2^m vectors $\boldsymbol{\alpha}_o$ **do**
        $\boldsymbol{y}_o = \boldsymbol{y}_{min} + Y_o \boldsymbol{\alpha}_o$
        if $V_2^T \boldsymbol{y}_o \in [\boldsymbol{t}_{lb}, \boldsymbol{t}_{ub}]$ **then**
          update candidates: $L_o \leftarrow [L_o,\ \boldsymbol{y}_o]$
      if L_o not empty **then**
        calculate the inverse T^{-1}
        for all $\boldsymbol{y}_o$ in L_o **do**
          $\boldsymbol{\alpha}_x = T^{-1} V_2^T \boldsymbol{y}_o$
          if $\boldsymbol{\alpha}_x \in [\mathbf{0}, \mathbf{1}]$ **then**
            $\boldsymbol{y}_v = \boldsymbol{y}_o + Y_x \boldsymbol{\alpha}_x$
            $\boldsymbol{x}_v = A^+ \boldsymbol{y}_v$
            update $\mathcal{V}$-rep: $X_v \leftarrow [X_v,\ \boldsymbol{x}_v]$, $Y_v \leftarrow [Y_v,\ \boldsymbol{y}_v]$
  return $\mathcal{V}$-rep: X_v, Y_v

3.5 ICHM: New algorithm for polytope approximation

This section introduces a new polytope approximation algorithm called **I**mplicit (or **I**terative) **C**onvex-**H**ull **M**ethod (ICHM). This algorithm is based Convex-Hull Method (CHM), developed by Lassez [149], and tailored specifically for the generic (implicit) polytope formulation

$$\mathcal{P}_x = \left\{ \boldsymbol{x} \in \mathbb{R}^m \,|\, A\boldsymbol{x} = B\boldsymbol{y} + \boldsymbol{b}, \quad \boldsymbol{y} \in \mathcal{P}_y \right\} \tag{3.73}$$

where the input set $\mathcal{P}_y$ can be expressed with its $\mathcal{H}$-representation

$$\mathcal{P}_y = \left\{ \boldsymbol{y} \in \mathbb{R}^n \,|\, H_y \boldsymbol{y} \leq \boldsymbol{d}_{\boldsymbol{y}} \right\} \tag{3.74}$$

This polytope formulation, as discussed in Section 2.4, unifies the formulations of all the common polytope characterisations of human's and robot's physical abilities. The correspondence of different physical ability polytopes and the formulation (3.73) can be found in Table 2.1.

The generic polytope formulation (3.73) has an implicit form $A\boldsymbol{x} = B\boldsymbol{y}$ which cannot be used directly with standard polytope transformation strategies described in Section 3.3. More specifically, its formulation corresponds to the intersection-projection, introduced in Section 3.2.3.1, a special case of the intersection formulation

$$\mathcal{P}_x \in \{ \boldsymbol{x} \in \mathbb{R}^m \mid A\boldsymbol{x} = \boldsymbol{z} + \boldsymbol{b},\ \boldsymbol{z} \in \mathcal{P}_z \} \tag{3.75}$$

where the input set polytope $\mathcal{P}_z \in \mathbb{R}^k$ is defined using the projection formulation

$$\mathcal{P}_z \in \{\boldsymbol{z} \in \mathbb{R}^k \mid z = B\boldsymbol{y},\ \boldsymbol{y} \in \mathcal{P}_y\} \tag{3.76}$$

Finding the $\mathcal{V}$ and $\mathcal{H}$-representation of the polytope $\mathcal{P}_x$, using the standard methods, requires separating the problem and first transforming the input set $\mathcal{P}_z$ in a suitable representation (usually $\mathcal{H}$), followed by the second step of transforming the final polytope $\mathcal{P}_x$.

However, when the input space is high dimensional $n \gg 1$, both polytope $\mathcal{P}_z$ and $\mathcal{P}_x$ have complex geometries, consisting in large number of vertices and faces. The high dimensional input space (and low dimensional output spaces $n \gg m$) are very common when it comes to characterising the physical abilities of humans, based on their musculoskeletal modes. The musculoskeletal models often have large number of muscles $n \gg 1$, often more than 50, while the output space usually corresponds to the 3D Cartesian Space (CS) ($m = 3$). In such cases, the exact polytope transformation methods have intractable computation times, as they relay on the exhaustive search in the high dimensional input space.

To overcome the complexity of the exhaustive search, various approximate approaches, such as the Ray Shooting Method (RSM) [148] and the Convex-Hull Method (CHM) [149] described in Section 3.3.4, have been developed, reducing the computation complexity and improving the execution time. Although RSM algorithms are relatively simple to setup they do not provide any bound on their estimation error and highly rely on hand tuned initial parameters. CHM [149] algorithms on the other hand, perform a very efficient iterative approximation, while at the same time guaranteeing the bound of the user defined approximation error. In each iteration CHM refines the approximation, by using Linear Programs (LPs) to find new vertices of the polytope and Convex-Hull to group them to faces. Furthermore, CHM finds the $\mathcal{V}$ and the $\mathcal{H}$-representation of the polytope at the same time. However in its standard formulation, the CHM is not suitable for the family of problems given with equation (3.73).

Therefore, this section proposes a new algorithm extending the CHM algorithm to the generic polytope formulation (3.73). Section 3.5.1 introduces the LP formulation adapting the CHM approach to the implicit problem formulation (3.73), while Section 3.5.2 brings the ICHM algorithm overview, as well as the pseudo-code and the geometrical representation. Finally, Section 3.5.3 brings the performance analysis of the ICHM and comparison to the state-of-the-art methods on the example of finding the $\mathcal{V}$-representation of the human wrench capacity polytope based on the musculoskeletal models, which is particularly challenging both due its implicit formulation and high dimensional input space $n \gg m$.

3.5.1 Linear programming formulation

From a geometrical point of view, solving a Linear Program (LP) problem boils down to finding a vertex of a polytope [157].

$$\begin{aligned} \boldsymbol{x}_v = \arg\max_{\boldsymbol{x}} \quad & \boldsymbol{c}^T\boldsymbol{x} \\ \text{s.t.} \quad & \boldsymbol{x} \in \mathcal{P}_x \end{aligned} \tag{3.77}$$

However, in order for the polytope $\mathcal{P}_x$ to be suitable for LP optimisation in the m-dimensional output space, its implicit formulation $A\boldsymbol{x} = B\boldsymbol{y}$ needs to be expressed in explicit form.

In the general case, the input space is higher dimensional $n \geq m$ and the matrix A is not square. As discussed in Section 3.2.2, to find an explicit form of equation (3.73), the

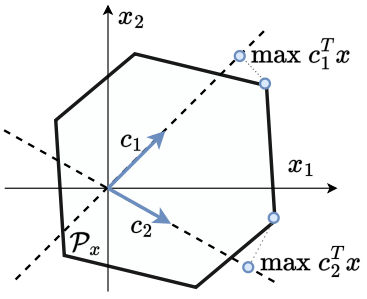

FIGURE 3.22: The figure demonstrates the LP based vertex search by choosing two different vectors $\boldsymbol{c}$.

pseudo-inverse A^+ can be used as a solution only if $B\boldsymbol{y}+\boldsymbol{b}$ belongs to the image $\mathcal{I}m\,(A)$ of A [156]. Therefore, equation

$$A\boldsymbol{x} = B\boldsymbol{y}+\boldsymbol{b} \tag{3.78}$$

can be transformed to

$$\boldsymbol{x} = A^+(B\boldsymbol{y}+\boldsymbol{b}), \qquad \text{where} \quad B\boldsymbol{y}+\boldsymbol{b} \in \mathcal{I}m(A) \tag{3.79}$$

Using the Singular Value Decomposition (SVD) [156] of the matrix A^T produces $A^T = U\Sigma V^T$. The rotation matrix $V \in \mathbb{R}^{n\times n}$ is separated into $V_1 \in \mathbb{R}^{n\times m}$, a projector to the image $\mathcal{I}m\,(A)$ of A, and $V_2 \in \mathbb{R}^{n\times(n-m)}$, a projector to its null-space $\mathcal{K}er\left(A^T\right)$. As all the vectors belonging to the $\mathcal{I}m\,(A)$ have zero projection to the null-space $\mathcal{K}er\left(A^T\right)$, an equality constraint can be devised for $B\boldsymbol{y}+\boldsymbol{b}$ to belong to the image $\mathcal{I}m\,(A)$ of A.

$$V_2^T B\boldsymbol{y} + V_2^T\boldsymbol{b} = \boldsymbol{0} \tag{3.80}$$

Finally, combining equations (3.80), (3.79), (3.74) and (3.73), one can create a Linear Program (LP) capable of reaching all vertices of the polytope $\mathcal{P}_x$

$$\begin{aligned} \max_{\boldsymbol{y}} \quad & \boldsymbol{c}^T A^+ B\boldsymbol{y} + A^+\boldsymbol{b} \\ \text{s.t.} \quad & V_2^T B\boldsymbol{y} = -V_2^T\boldsymbol{b} \\ & H_y\boldsymbol{y} \le \boldsymbol{d}_y \end{aligned} \tag{3.81}$$

by appropriately choosing the projection $\boldsymbol{c}$, as demonstrated on Figure 3.22.

3.5.2 Proposed ICHM algorithm overview

Given the polytope definition (3.73) and the Linear Program (LP) problem defined in (3.81), the proposed ICHM algorihtm provides a structured way to choose vectors $\boldsymbol{c}$ to find all the vertices ($\mathcal{V}$-representation) and facets ($\mathcal{H}$-representation) of the polytope $\mathcal{P}_x$. In order to do so, the method leverages the iterative Convex-Hull $\mathcal{C}$ calculation, where each newly found vertex extends $\mathcal{C}$. The normal vectors of the faces of the newly obtained $\mathcal{C}$ are used to decide for the new vectors $\boldsymbol{c}$ to use in the LP (3.81). This iterative process is performed until some specified level of accuracy is reached. Figure 3.23 visually demonstrates several iterations of the algorithm for the 2-dimensional polytope $\mathcal{P}_x$ example.

3.5.2.1 Initial Convex-Hull

More precisely, the first step of the algorithm constructs an initial set of vertices X_v and an initial Convex-Hull $\mathcal{C}$. In a m-dimensional space, the minimal number of points to create a volume is $m+1$. There are various ways proposed in the literature to obtain the initial set of points in order to start the refinement process, such as starting by a random set of directions $\boldsymbol{c}$ proposed by Lassez [149] in their original paper. Another approach, proposed by Del Prete *et al.* [152] proposes to uniformly sample the input space.

The approach proposed in this work exploits the base vectors $\boldsymbol{u}_i \in U$ obtained using the Singular Value Decomposition (SVD) of the matrix $(A^+B) = U\Sigma V^T$. This approach has two benefits. One one hand, the base vectors $\boldsymbol{u}_i$ are orthogonal vectors in the output space corresponding to the highest variability (amplification) directions, given by the singular values σ_i of the matrix A^+B. Therefore, the chances of not obtaining $m-1$ different vertices when projecting the polytope $\mathcal{P}_x$ in these directions is much lower than in random directions. And on the other hand, for any matrix A and B, the matrix U is relatively efficient to calculate.

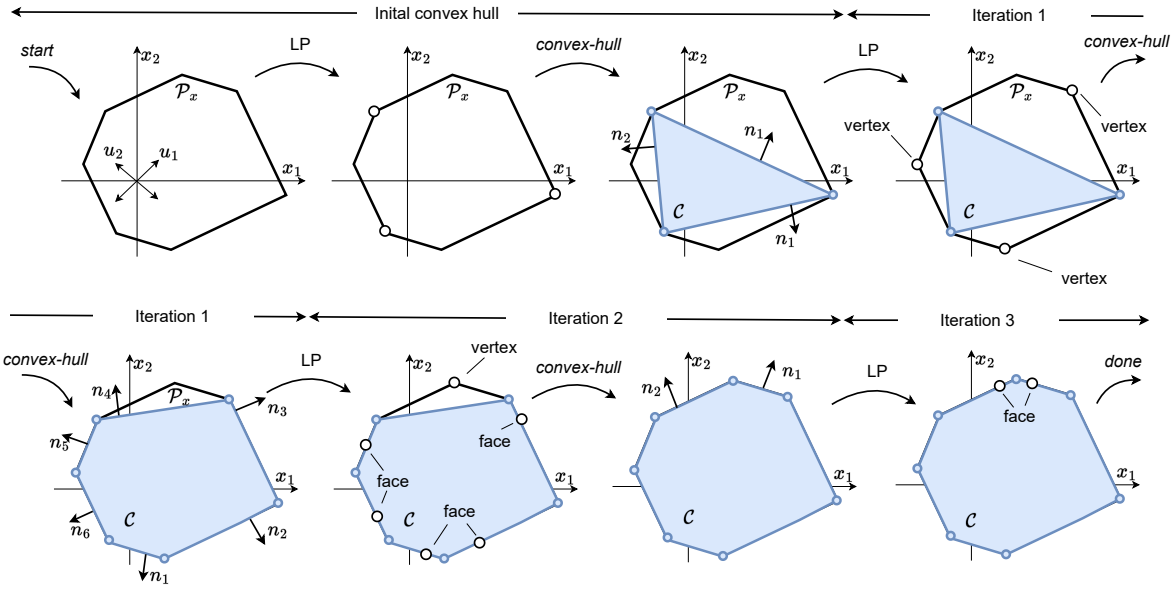

FIGURE 3.23: This figure shows the procedure of the successive approximation of the polytope $\mathcal{P}_x$ using the proposed ICHM algorithm. The approximation starts with the initial Convex-Hull $\mathcal{C}$ construction, using the vectors $\boldsymbol{u}_i \in U$. Then the approximation is refined by using the face normal vectors $\boldsymbol{n}_i$ of the Convex-Hull $\mathcal{C}$ with the LP (3.81) to find new vertices which are then used to update the Convex-Hull $\mathcal{C}$. The check (3.84) is used to determine if the LP results belong to the faces or they are new vertices of the polytope. The algorithm stops when no new vertices are found.

Therefore, the proposed approach to find the initial set of vertices consists in solving two LPs (3.81) for each one of the m base vectors $\boldsymbol{u}_i$, one that minimises ($\boldsymbol{c} = -\boldsymbol{u}_i$) and one that maximises ($\boldsymbol{c} = \boldsymbol{u}_i$) the projection of the polytope $\mathcal{P}_x$. In the ideal case, this approach obtains $2m$ vertices of the polytope $\mathcal{P}_x$. Once the initial set of vertices $\boldsymbol{x}_v$ is found, the initial Convex-Hull $\mathcal{C}$ can be calculated. Its centroid position is given by $\boldsymbol{x}_c = \frac{1}{N}\sum \boldsymbol{x}_{v,i}$.

3.5.2.2 Incremental refinement

The next step of the algorithm is to iterate over all the faces $\mathcal{F}_i$ of the Convex-Hull $\mathcal{C}$. For each new face $\mathcal{F}_i$, the normal $\boldsymbol{n}_i$ is found and used as a candidate $\boldsymbol{c}_i$ in the direction pointing out of polytope $\mathcal{P}_x$. The normal vector direction is verified by projecting the vector going from the centroid $\boldsymbol{x}_c$ to any vertex $\boldsymbol{x}_j$ of face $\mathcal{F}_i$, onto the normal $\boldsymbol{n}_i$ and verifying if the scalar product is positive or negative.

$$\boldsymbol{c}_i = \begin{cases} \boldsymbol{n}_i, & \text{if } \boldsymbol{n}_i^T(\boldsymbol{x}_j - \boldsymbol{x}_c) \geq 0, \\ -\boldsymbol{n}_i, & \text{otherwise.} \end{cases} \tag{3.82}$$

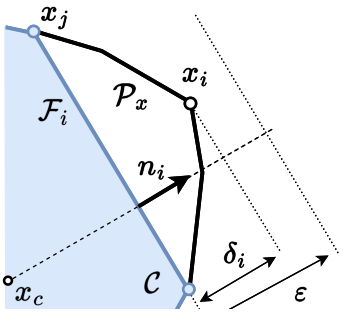

FIGURE 3.24: The figure shows an example of the polytope face condition from equations (3.84) and (3.83). The face $\mathcal{F}_i$ of the Convex-Hull $\mathcal{C}$ is considered to be the face of $\mathcal{P}_x$, because $\delta_i < \varepsilon$.

Solving equation (3.81) with $\boldsymbol{c}_i$, the obtained $\boldsymbol{x}_i$ can be either a vertex of the polytope $\mathcal{P}_x$ or be coplanar with face $\mathcal{F}_i$. To determine if $\boldsymbol{x}_i$ is coplanar with the face, a simple check can be devised, which verifies if the orthogonal (normal) distance from $\boldsymbol{x}_i$ to any vertex $\boldsymbol{x}_j$ of face $\mathcal{F}_i$

$$\delta_i = \boldsymbol{n}_i^T(\boldsymbol{x}_j - \boldsymbol{x}_i) \tag{3.83}$$

is within a certain user defined accuracy ε

$$\boldsymbol{x}_i = \begin{cases} \text{vertex}, & \text{if } |\delta_i| \geq \varepsilon, \\ \text{on the face}, & \text{otherwise.} \end{cases} \tag{3.84}$$

Figure 3.24 provides a graphical interpretation of this check and of the values δ_i and ε.

If $\boldsymbol{x}_i$ belongs to face $\mathcal{F}_i$ of the Convex-Hull $\mathcal{C}$ then $\mathcal{F}_i$ is considered to be a face of the polytope $\mathcal{P}_x$. In that case the algorithm updates the $\mathcal{H}$-representation of $\mathcal{P}_x$

$$H \leftarrow \begin{bmatrix} H \\ \boldsymbol{n}_i^T \end{bmatrix}, \quad \boldsymbol{d} \leftarrow \begin{bmatrix} \boldsymbol{d} \\ \boldsymbol{n}_i^T \boldsymbol{x}_j \end{bmatrix} \tag{3.85}$$

where $\boldsymbol{n}_i$ is the face normal vector and $\boldsymbol{n}_i^T \boldsymbol{x}_j$ is the orthogonal distance from the origin to the face $\mathcal{F}_i$. On the other hand, if $\boldsymbol{x}_i$ is a vertex of the polytope it is appended to the $\mathcal{V}$-representation list $X_v \leftarrow [X_v,\ \boldsymbol{x}_i]$.

3.5.2.3 Stopping condition

Once all the faces of the Convex-Hull $\mathcal{C}$ are evaluated, the Convex-Hull is updated using the new vertex list X_v.

One of the algorithm stopping conditions proposed by Bretl and Lall [151], is to construct the inner and outer approximation of the polytope $\mathcal{P}_x$ at the same time and stop approximating when the ratio $r = \mathcal{V}ol\left(\mathcal{P}_{inner}\right) / \mathcal{V}ol\left(\mathcal{P}_{outer}\right)$ between their volumes reaches a certain threshold $r \leq 1$.

However, volume ratio condition is sometimes hard to interpret, as its representation of the approximation error does not have the same units as the physical quantity being represented by the polytope $\mathcal{P}_x$. Therefore, the ICHM algorithm exploits a different stopping condition setting the threshold on the maximal distance δ_i. The algorithm stops iterating if the vertex list X_v has no new elements or, in other words, if the maximal distance δ_i for all the faces $\mathcal{F}_i$ is lower than the user defined accuracy ε

$$\max\{|\delta_i|\} \leq \varepsilon \tag{3.86}$$

In this stopping condition, the maximal distance δ_i corresponds to the maximal error committed by the approximation, expressed with the same units as the units of the physical quantity being represented by the polytope. Provided that the output space has the same units in all the axis. Preserving the same units improves the interpretation of the approximation error and enables the user to intuitively decide on the acceptable approximation error ε for the application in question.

Finally, if the user sets the approximation error $\varepsilon = 0$, the ICHM algorithm finds the exact solution, all the vertices and the faces of the polytope $\mathcal{P}_x$.

3.5.3 Performance analysis

To evaluate the efficiency of the ICHM algorithm, the challenging polytope formulation, corresponding to the generic formulation (3.73), with a high-dimensional input space and a low-dimensional output space ($n \gg m$) is tested. Additionally, the algorithm performance is compared against two state-of-the-art approaches: an approximation approach based on the Ray Shooting Method (RSM) algorithm introduced by Carmichael and Liu [69] and an exact

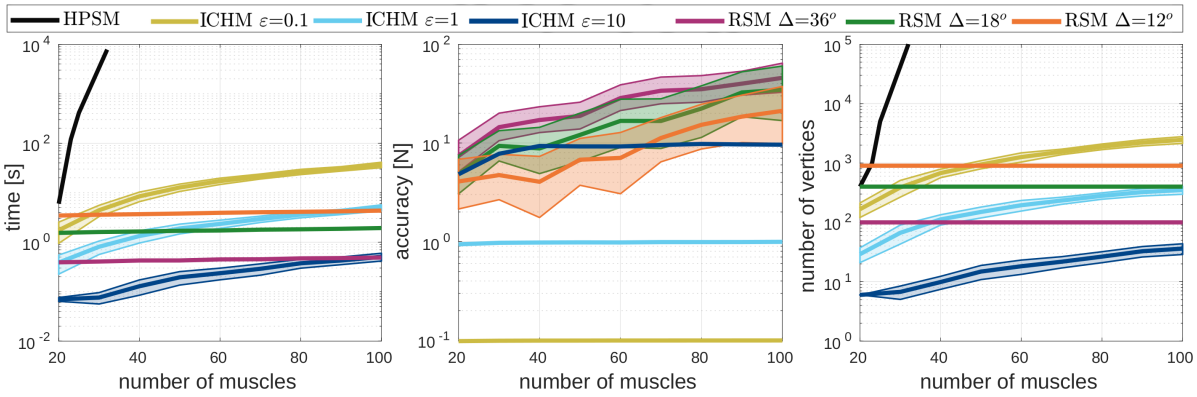

FIGURE 3.25: Figure presenting the performance analysis results in the logarithmic scale for three different algorithms: HPSM, RSM and proposed ICHM with respect to different number of muscles. The left figure shows the evolution of the execution time. The middle figure shows the evolution of the underestimation error, the maximal distance in between the polytope $\mathcal{P}_x$ and the acquired polytope, calculated as $\max\{|\delta_i|\}$. The right figure shows the evolution of the number of vertices found. All the plots show the averaged results, with variances, over 100 runs, where each run corresponds to one randomly generated model.

evaluation approach based on the combination of two efficient algorithms Hyper-Plane Shifting Method (HPSM) [147] and Pivoting Method (PIM) [122].

3.5.3.1 Human wrench capacity polytope

The polytope in question is the human's wrench capacity polytope, previously discussed in Section 2.2.3. The polytope $\mathcal{P}_f$ characterising the achievable wrenches $\boldsymbol{f}$, the human can generate given its musculoskeletal model, can be expressed as

$$\mathcal{P}_f(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}}) = \left\{\boldsymbol{f}\in\mathbb{R}^m \mid \boldsymbol{F}\in[\boldsymbol{F}_{min},\boldsymbol{F}_{max}],\ J^T(\boldsymbol{q})\boldsymbol{f} = N(\boldsymbol{q})\boldsymbol{F} - \boldsymbol{\tau}_b(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}})\right\} \tag{3.87}$$

where $\boldsymbol{f}\in\mathbb{R}^m$ is the Cartesian Space (CS) wrench, $\boldsymbol{F}\in\mathbb{R}^n$ is the vector of muscular forces, while $J(\boldsymbol{q})$ and $N(\boldsymbol{q})$ are configuration $\boldsymbol{q}$ dependant Jacobian and moment arm matrix. Additionally, the bias vector $\boldsymbol{\tau}_b$ corresponds to the joint torques corresponding to the effects of the gravity and the movement.

This polytope formulation is challenging due to the high dimensional input space (large number of muscles), making the geometry of the polytope $\mathcal{P}_f$ relatively complex, having large number of faces and vertices. In such cases, standard exact methods for polytope transformation have long execution times, preventing this polytope representation to be used in interactive (online) applications. To reduce the computation times of the wrench polytope $\mathcal{P}_f$ evaluation, Carmichael and Liu [69] have proposed an algorithm based on the RSM, described in Section 3.3.4. Although the algorithm enables real-time approximation of the polytope $\mathcal{P}_f$, the obtained approximation is coarse and without any bounds on the approximation accuracy.

3.5.3.2 Experiment and results

Therefore, to evaluate the performance of the proposed ICHM algorithm, a comparative experiment is performed, where the ICHM algorithm's performance is compared against the RSM method proposed by Carmichael and Liu [69] as well as against the two step exact approach based on the HPSM [147] and PIM [122]. The experiment consists in finding the $\mathcal{V}$-representation of the CS force $m=3$ polytope $\mathcal{P}_f$ for a randomised mock-up musculoskeletal model with $k=7$ degrees of freedom and a number of muscles ranging from $n=20$ to 100.

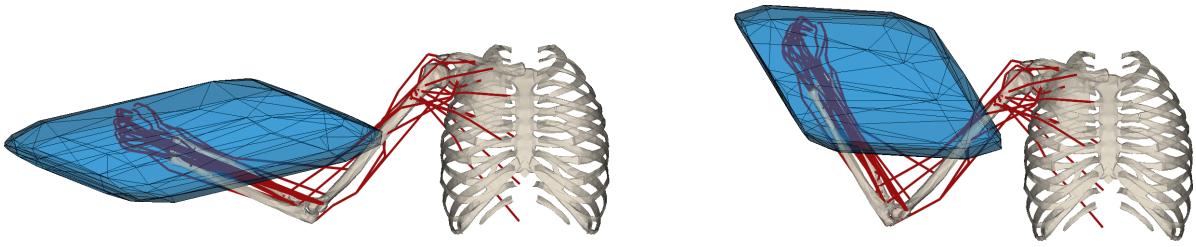

FIGURE 3.26: CS force polytope of a musculoskeletal model of human upper limb [158] with 7DOF and 50 muscles, evaluated for two different configurations using the proposed ICHM algorithm. The visualisation is performed using `bioviz` [159] and polytopes are scaled with a ratio 1m : 1000N

The RSM algorithm [108] requires uniform sampling of the ray directions in the 3D space. In this experiment the sampling is performed based on a two Euler angles parametrisation. For this method, three linearly increasing levels of granularity are tested for each Euler angle: $\Delta = 36^o$, 18^o and 12^o; making for $n_r = 10, 20$ and 30 rays per angle. Overall number of ray directions in 3D space ($N_r = n_r^2$) is then 100, 400 and 900. For the proposed ICHM algorithm, three exponentially increasing levels of accuracy are tested $\varepsilon = 0.1$, 1 and 10 N. All the algorithms are implemented in Matlab and run on a 1.90GHz Intel i7-8650U processor. The source code of the experiment is publicly available on GitLab[3].

Results of the experiments, averaged over 100 algorithm runs, are shown on Figure 3.25. The results confirm that the exact approach using the HPSM method has an execution time exponentially related to the number of muscles. Even for only 30 muscles it already takes more than 4.5 h to calculate, therefore this method is not tested on more than 30 muscles, where it finds over 10^4 vertices.

The RSM algorithm shows nearly constant time of execution for the full range of tested muscles and the constant number of vertices found, which is expected. The results show however, that for low muscle numbers $d < 25$, when using a fine granularity $\Delta = 12^o$ the RSM algorithm finds more vertices than the exact solution found by the HPSM. Furthermore, the middle plot of Figure 3.25 shows that the RSM estimation error increases considerably with the number of muscles d, followed by a very high variance. These results confirm that, as the polytope shape is not spherical, by uniformly covering the space of ray directions, RSM based algorithms will necessarily estimate certain areas of the polytope better than the others.

The graphs show that the proposed ICHM method's execution time depends near-linearly of the number of muscles considered d and the estimation error bound parameter ε. The accuracy graph, shown in the middle plot, shows that the proposed ICHM algorithm is capable of limiting the estimation error of the polytope evaluation under desired value ε regardless of the number of muscles. Furthermore, considering the vertex number found by the algorithms, it can be seen that the number of vertices has a nearly-linear relationship with the number of muscles d and the variable ε.

The demonstrated efficiency of the ICHM algorithm opens many doors for its applications in real-time systems, providing the user with an easy to understand trade-off between speed and accuracy. Figure 3.26 shows the visualisation of the CS force ($m = 3$) polytope of a human musculoskeletal model with 50 muscles and 7DOFs, calculated using the ICHM algorithm.

[3]Gitlab: https://gitlab.inria.fr/auctus-team/people/antunskuric/papers/human_wrench_capacity

3.5.4 Algorithm implementation

The pseudo-code of the ICHM algorithm is given in Algorithm 2, while an efficient open-source Python implementation is publicly available within the package `pycapacity`[4], which is described more in detail in Chapter 7.

Finally, Section 4.3 brings the application of the proposed algorithm for the real-time robot control in the human-robot collaboration scenario, where the algorithm is exploited in order to calculate the human's wrench capacity polytope online.

pycapacity

Algorithm 2 Proposed ICHM algorithm pseudo-code

Require: A,B, $\boldsymbol{y}_{min}$, $\boldsymbol{y}_{max}$, $\boldsymbol{b}$, ε
  $U, \Sigma, V^T \leftarrow svd(A^+)$
  init $\mathcal{H}$-rep: $H, \boldsymbol{d} \leftarrow [\,]$ and $\mathcal{V}$-rep: $\boldsymbol{x}_v, \boldsymbol{y}_v \leftarrow [\,]$
  for all $\boldsymbol{u}_i$ in U **do**
    $\boldsymbol{y}_i \leftarrow$ Linear Program (LP) (Eq. 3.81) with $\boldsymbol{c}{=}\boldsymbol{u}_i$ and $\boldsymbol{c}{=}-\boldsymbol{u}_i$
    $\boldsymbol{x}_v \leftarrow [\boldsymbol{x}_v,\ A^+B\boldsymbol{y}_i + A^+\boldsymbol{b}]$
  repeat
    calculate the Convex-Hull $\mathcal{C}$ of $\boldsymbol{x}_v$
    calculate the centroid $\boldsymbol{x}_c = \frac{1}{N}\sum_i \boldsymbol{x}_{v,i}$
    for all new face $\mathcal{F}_i$ in $\mathcal{C}$ **do**
      find normal $\boldsymbol{n}_i$ and one vertex $\boldsymbol{x}_j$ of $\mathcal{F}_i$
      $\boldsymbol{y}_i \leftarrow$ LP (Eq. 3.81) with $\boldsymbol{c}{=}\boldsymbol{n}_i$ (Eq. 3.82)
      $\boldsymbol{x}_i = A^+B\boldsymbol{y}_i + A^+\boldsymbol{b}$
      calculate δ_i (Eq.3.83)
      if $|\delta_i| \leq \varepsilon$ **then**
        new face: update $\mathcal{H}$-rep (Eq. 3.85)
      else
        new vertex: update $\mathcal{V}$-rep: $\boldsymbol{x}_v \leftarrow [\boldsymbol{x}_v,\ \boldsymbol{x}_i]$, $\boldsymbol{y}_v \leftarrow [\boldsymbol{y}_v,\ \boldsymbol{y}_i]$
  until $\max\{|\delta_i|\} \leq \varepsilon$
  return $\mathcal{V}$-rep: $\boldsymbol{x}_v$, $\boldsymbol{y}_v$ and $\mathcal{H}$-rep: H, $\boldsymbol{d}$

3.6 Conclusion

This chapter aims to provide a set of tools for efficient transforming of polytopes representing the physical abilities of humans and robots into more standardised forms suitable for practical applications. The focus is put on the two widely used polytope representations: vertex or $\mathcal{V}$ and half-plane or $\mathcal{H}$-representations. These representations offer various advantages, including compatibility with visualisation tools, in a for triangulated meshes, and optimisation-based applications like robot control and trajectory planning. Additionally, they enable efficient operations like Minkowski sums and intersections over multiple polytopes.

While the literature presents numerous algorithms for transforming polytope formulations into their $\mathcal{H}$ or $\mathcal{V}$-representation, these algorithms are often tailored to specific sets of polytope formulations. Chapter 2 proposes a generic view on different common polytope representation of humans' and robots' physical abilities and proposed a unifying polytope formulation describing all the proposed metrics, introduced in Section 2.4. However, this unified formulation is not directly suitable for the standard polytope transformation algorithms. Hence, Section 3.2

[4] https://auctus-team.github.io/pycapacity/

proposes a structured view on different polytope formulation families derived from the proposed generic unified formulation namely intersection and projection formulation, as well as some special cases.

Section 3.3 further provides an overview of applicable techniques for transforming these families into their $\mathcal{V}$ or $\mathcal{H}$-representation. The chapter briefly introduces the standard generic methods for mutual transformation between $\mathcal{V}$ and $\mathcal{H}$-representation, followed by the polytope formulation specific methods that are capable of better exploiting the geometry of the formulation families. Finally, several polytope approximation methods are discussed as a means of improving the efficiency of polytope transformation for high-dimensional problems. The aim of this overview is to provide a brief introduction into the applicable state-of-the-art methods when it comes to transforming different polytope formulations as well as to discuss their limitations and computational efficiency.

The final two sections of this chapter introduce two contributions of this thesis in the domain of polytope transformations. The Section 3.4 introduces an efficient vertex enumeration algorithm for the intersection polytope formulation with the interval input set. This algorithm is based on the work by Sasaki [106], exploiting the geometry of the hyperrectangle input set in order to improve its computational complexity of its exhaustive search and reduce the execution time. The algorithm is particularly well suited for low dimensional polytope evaluation, that are common when it comes to polytope characterisations of physical abilities of robotic manipulators. The performance of the proposed algorithm is compared to the state-of-the-art methods on finding the $\mathcal{V}$-representation of the force capacity polytope for redundant robotic manipulators, introduced in Section 2.1.5. The results show that the algorithm has significantly lower computational complexity and hence shorter execution time, in the order of magnitude of several milliseconds for standard robotic manipulators, implemented in Pyhton. Such short execution time opens doors for using these polytopes in real time applications, such as robot control. This algorithm has been published in the context of the scientific paper **A. Skuric** *et al.* [A1].

Section 3.5 brings a new polytope approximation algorithm based on the Convex-Hull Method (CHM) by Lassez [149]. The algorithm is developed for the generic polytope formulation directly and can therefore be used with all the common polytope formulation introduced in previous chapter. This algorithm performs an iterative approximation of the final polytope by using sequences of Linear Programs (LPs) to find new vertices of the polytope and Convex-Hull to group the to faces. The algorithm improves the approximation accuracy in each iteration and continues iterating when desired user defined approximation accuracy is reached. The execution time of the algorithm depends directly of the complexity of the polytope being transformed (number of faces and verteices), therefore by appropriately setting the desired accuracy, the algorithm is capable of simplifying the final geometry of the polytope and lowering the execution time. This is particularly useful when it comes to the high-dimensional problems, where the input space dimension is much higher than the output space $n \gg m$, where the standard exact methods have intractable execution times, as they are based on different forms of the exhaustive search. Such high-dimensional problems are very common when it comes to characterising physical abilities of humans based on their musculoskeletal models, as they often have large number of muscles n, while their capacities are characterised in the Cartesian Space (CS) $m = 3$. The performance of the proposed method is therefore tested on the challenging polytope formulation of the human's wrench capacity polytope, described in Section 2.2.3. The results of the proposed methods in comparison to the state-of-the-art methods show that it has significantly shorter computation time while at the same time guaranteeing the user defined approximation accuracy. Furthermore, the results show that the proposed algorithm, in the case of human's wrench capacity polytope, has execution time under half of a seconds

for musculoskeletal models up to 100 muscles, opening many doors for using this polytope formulation in the real time applications. The ICHM algorithm has been published in the context of the scientific paper **A. Skuric** *et al.* [A3] and later refined within **A. Skuric** *et al.* [A4].

Following chapter leverages the proposed algorithms and formulations to propose several use-cases of the polytope based physical ability characterisations in the context of the real-time robot control in different human robot physical collaboration scenarios. Chapter 4 proposes uses the real-time polytope evaluation for creating more adapted robot control strategies in the context of the physical human-robot interaction. Chapter 5 explores the possibilities of using polytopes for the real-time visual feedback to the operators. Moreover, Chapter 6 proposes a new CS trajectory planning strategy exploiting the polytope algebra. Finally, Chapter 7 presents the publicly available open-source software package implementing several algorithms introduced in this section and enabling an efficient, real-time compatible, evaluation of polytope based physical abilities for humans and robots.

Chapter 4

Enhancing human-robot physical collaboration with polytopes

Chapter 2 presentes the benefits of polytope representations of physical abilities for robots and humans, based on their musculoskeletal models. Polytopes, in addition to being their accurate representation, can actually be used as a unifying view on their different physical abilities and allow for characterising their common physical abilities, as one system, in the same polytope form. Chapter 3 then presents some of the most efficient algorithms, proposed in the literature, for transforming different polytope families into the standard forms suitable for different applications, such as visualisation or robot control. Furthermore, in Chapter 3, two new efficient algorithms for polytope transformation are proposed: VEPOLI^2 (introduced in Section 3.4) and ICHM (introduced in Section 3.5). These algorithms significantly reduce the computational complexity of the polytope transformation and enable their use in real-time applications.

The context of this chapter is put within a collaborative workstation [5, 160], where the human and the robot work in a close proximity and interact physically to execute different tasks. Collaborative workstations are promising tools for the future of the small-scale manufacturing [161, 162], such as the mini-satellite fabrication in the context of the LiChiE project. The aim of such collaboration is to improve the overall efficiency by combining the abilities of both humans (flexibility, adaptability, expertise, etc.) and robots (repeatability, precision, tirelessness, etc.), while at the same time improving the operator's safety, and overall well-being, when executing different tasks. Therefore, this chapter leverages the efficiency of the proposed algorithms and aims to demonstrate the potential of using polytope representation of human's and robot's physical abilities for creating real-time robot control strategies in the context of human-robot physical collaboration.

One of the main challenges of such collaborative workstations is creating more adaptive control strategies where the robot adapts its behaviour, not just with respect to the requirements of the task and its own abilities, but also to the current abilities of the human operator, as well as his safety. Creating these collaborative strategies requires having a set of tools for characterising the abilities of robots and humans, as well as quantifying different notions of human's well-being and safety.

This chapter, therefore, aims to demonstrate that polytope representations of different physical abilities of robots and humans, have a great potential to be used in this context. As showed in Section 2.3, polytopes allow expressing different physical abilities of robots and humans, as well as their collaboration as a single system, in the same polytope form. Such a unified view enables assessing if different tasks better suite the human's or the robot's abilities, or if they better suite their physical collaboration. This lays the foundation for more adapted objective and quantitative task allocation strategies. Furthermore, as both human's and robot's physical

abilities are state dependant and can vary significantly during the task execution, this work proposes to use the polytopes for capturing the changes in their abilities in real-time.

Having accurate information about robot's physical abilities in real-time, enables creating robot control strategies that adapt to the changes and better exploit robot's abilities. On the other hand, having accurate online information about human's physical abilities enables monitoring the operator's capacity to execute a certain task in real-time. This real-time information can be used to quantify the operators' lacking physical abilities to execute certain task, or in other words, quantify how much assistance they need from the robot. Such monitoring and assistance strategies allow ensuring that their abilities are not surpassed during the task execution, having a direct impact on the operator's safety and well-being.

Therefore, there are three main concepts that this chapter attempts to explore

Concept 1 (C1)**.** *Real-time physical ability information enables creating more adapted control strategies that better exploit robot's and human's individual abilities*

Concept 2 (C2)**.** *Real-time physical ability information enables creating collaborative control strategies that better exploit their common abilities when collaborating physically*

Concept 3 (C3)**.** *Real-time physical ability information enables creating more human-centred robot control strategies improving human's safety*

This chapter is structured as follows. Section 4.1 brings an example of the task of collaborative object carrying within a collaborative workstation, where two collaborative scenarios are proposed. The first scenario, introduced in Section 4.2, consists in two robots collaborating to carry a heavy object while the control strategies of each one of the robots take in consideration both their own and the other robot's physical abilities. The second scenario, proposed in Section 4.3, consists in the human operator and the robot collaboratively carrying a heavy object, where the robot's control strategy takes in consideration its own physical abilities and the physical abilities of the operator in the real-time. Section 4.4 brings the discussion on the limitations and perspectives of the proposed experiments and the collaborative control approaches.

4.1 Collaborative carrying of a heavy object

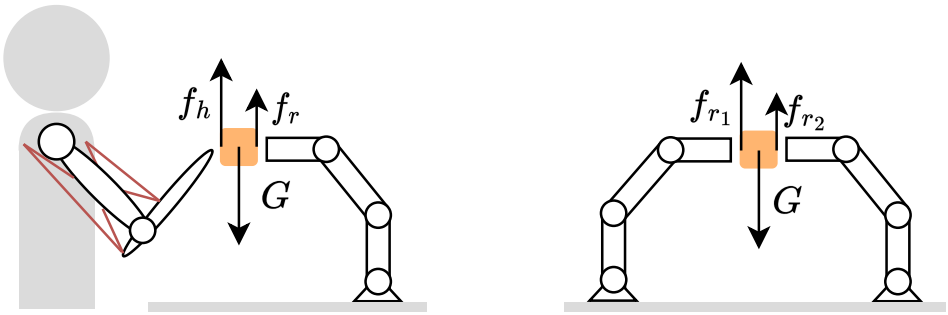

FIGURE 4.1: Illustrative example of collaborative object carrying in two different collaborative scenarios, human-robot collaboration on the left and dual robot arm collaboration on the right.

One traditional example of a collaborative task requiring physical interaction between multiple actors (humans and robots) is collaborative object carrying [163–165]. In this task N actors collaborate physically by applying forces $\boldsymbol{f}_{a_i}$ on the object with mass m, in order to compensate for its gravity G

$$G = m\boldsymbol{g} = \boldsymbol{f}_{a_1} + \cdots + \boldsymbol{f}_{a_N} \tag{4.1}$$

An illustration of the human-robot and dual robot arm collaborative carrying is given on Figure 4.1. As discussed in Section 2.3, in this particular collaboration scenario, each of the actors' capacity to generate forces $\boldsymbol{f}_{a_i}$ can be expressed in the polytope form

$$\boldsymbol{f}_{a_i} \in \mathcal{P}_{f,a_i} \tag{4.2}$$

while their joint capacity of generating the forces on the object can be found as the Minkowski sum of their polytopes

$$\mathcal{P}_f = \mathcal{P}_{f,a_1} \oplus \cdots \oplus \mathcal{P}_{f,a_N} \tag{4.3}$$

Therefore if gravity induced force acting the object G is within the polytope $\mathcal{P}_f$ the collaborative system is able to compensate for the objects weight

$$G \in \mathcal{P}_f \tag{4.4}$$

As both human's and robot's force capacity is state dependant[1] and changes during the execution of the task, the role of the robot control strategies is to find the suitable forces $\boldsymbol{f}_{a_i}$ that respect their individual capacities $\mathcal{P}_{f,a_i}$ and accomplish the task by compensating for the object's weight G.

The following two sections present two different robot control strategies for two different collaborative object carrying scenarios. Section 4.2 brings a physical collaboration scenario based on two robotic arms carrying an object with a mass of 12kg. Their physical abilities are calculated in real-time in the polytope form and integrated in the real-time robot control in order to exploit their full force capacity. Section 4.3, on the other hand, proposes the human-robot physical collaboration scenario for carrying of an object with the mass of 7kg. In this scenario, both of their physical abilities are evaluated online and used to create a human-centred robot control strategy, able to exploit their changing physical abilities while ensuring their safety.

4.2 Dual robotic arm collaborative object carrying

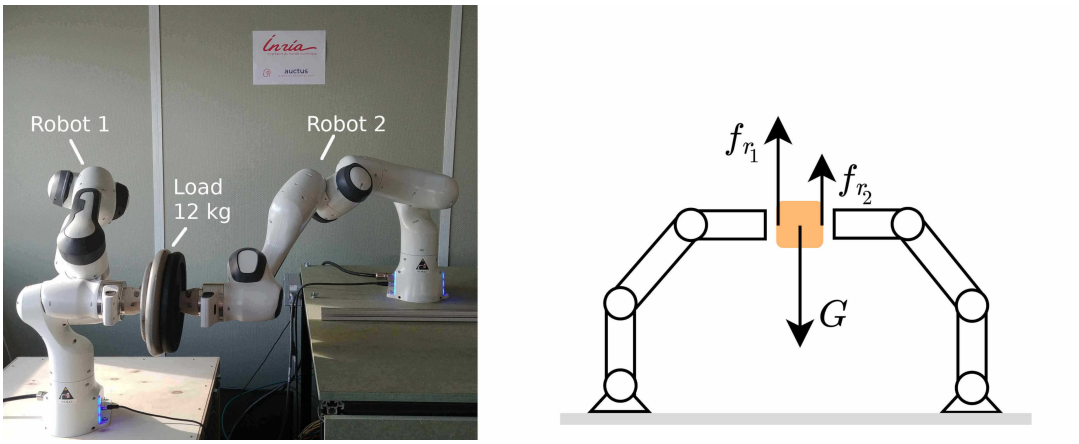

FIGURE 4.2: Figure showing the experimental setup for dual robot arm collaborative carrying. Two Franka Emika Panda robots jointly carry the object with a mass of $m = 12$kg, where each object compensates for a part of the total weight $\boldsymbol{f}_{r_1} + \boldsymbol{f}_{r_2} = m\boldsymbol{g}$. The object is rigidly fixed in the end-effectors of both robots.

This sections presents a physical collaboration scenario using two Franka Emika Panda robots involved in the collaborative carrying on an object of mass m. Each robot is contributing to

[1]The human's and the robot's state force capacity is dependant on their respective states. The exact relationships are described in Section 2.1.5 for robots and in Section 2.2.3 for humans.

the compensation of the object's gravity $G = m\boldsymbol{g}$ by producing forces $\boldsymbol{f}_{r_1}$ and $\boldsymbol{f}_{r_2}$, such that

$$\boldsymbol{f}_{r_1} + \boldsymbol{f}_{r_2} = G \tag{4.5}$$

Panda robots are rated to a maximal carrying capacity of 3kg, which corresponds to the absolute minimal carrying capacity of the robot, evaluated in one of its near-singular configurations. It is, by definition, an underestimation of the real robot's task space force capacity and relying on it, as it is commonly done, limits the scope of the possible tasks and applications.

The goal of this experiment is to demonstrate the fact that by taking into account the true force capacity of each robot, it is possible to go considerably beyond the robot's conservative rated capacity without comprising safety nor exceeding any of the actuation limits[2]. The weight of the object chosen for the experiments is $m = 12kg$, voluntarily far above the recommended joint carrying capacity $m \leq 6kg$.

The following Section 4.2.1 proposes an efficient approach for calculating the carrying capacity of both robots, leveraging the efficiency of the VEPOLI^2 algorithm (described in Section 3.4). Section 4.2.2 proposes a robot control strategy exploiting this real-time information in order to fully compensate for the continuous evolution of their capacity over time. Finally, Section 4.2.3 brings the experimental validation of the approach, as well as the discussion of the results.

4.2.1 Robot carrying capacity calculation

As the carrying task requires the robots to apply only the forces f_{r_1} and f_{r_2} that are opposite to the force of the object's gravity G, their physical ability to carry certain weight can be evaluated as the maximal applicable force in the vertical (z-axis) direction. Therefore, the robot's carrying capacity $\mathcal{F}_z$ can be expressed as the the special case of the robot's wrench polytope $\mathcal{P}_f$, described in Section 2.1.5, where the task space is one dimensional ($m = 1$)

$$\mathcal{F}_z(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \mathcal{P}_{f_z}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \{f \in \mathbb{R} \mid J_z(\boldsymbol{q})^T f = \boldsymbol{\tau} - \boldsymbol{\tau}_b(\boldsymbol{q}, \dot{\boldsymbol{q}}), \quad \boldsymbol{\tau} \in [\boldsymbol{\tau}_{min},\ \boldsymbol{\tau}_{max}]\} \tag{4.6}$$

where $\{\boldsymbol{q}, \dot{\boldsymbol{q}}\}$ is the robot's joint state, $f \in \mathbb{R}$ is the applicable scalar force in z-axis direction, $\boldsymbol{\tau} \in \mathbb{R}^n$ are the applied joint torques limited within the interval $\boldsymbol{\tau} \in [\boldsymbol{\tau}_{min},\ \boldsymbol{\tau}_{max}]$, $\boldsymbol{\tau}_b \in \mathbb{R}^n$ are the bias joint torques grouping the effects of gravity and robot's motions, while $J_z \in \mathbb{R}^{1\times n}$ is the configuration dependant Jacobian matrix with one line and n columns.

Geometrically, the $m = 1$ dimensional polytope, describing the range of scalar forces f applicable in the vertical direction, can be represented as the intersection of the complete wrench/force ($m = 6/m = 3$) polytope $\mathcal{P}_f$, described in Section 2.1.5, with the vertical axis. An illustrative view of this intersection is shown on Figure 4.3.

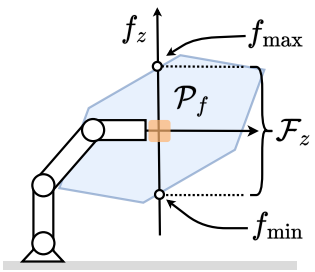

FIGURE 4.3: A geometric view of constructing the carrying capacity $\mathcal{F}_z$ as the intersection of the force polytope $\mathcal{P}_f$ and the z-axis. The carrying capacity is expressed at the center of the object.

For any given robot state $\{\boldsymbol{q}, \dot{\boldsymbol{q}}\}$ this one dimensional set $\mathcal{F}_z$ can be transformed into the form of a min-max interval

$$\mathcal{F}_z = \{f \in \mathbb{R} \mid f \in [f_{min},\ f_{max}]\} \tag{4.7}$$

Finding the limits $[f_{min},\ f_{max}]$ of the applicable scalar force f in the vertical direction can be viewed as finding a $\mathcal{V}$-representation of the 1D ($m = 1$) polytope $\mathcal{F}_z$. Then the

[2]It it worth noting that the mechanics of the robot's joints might not be designed to withstand the efforts above its rated payload in the long-run. However, this experiment aims to illustrate an optimised use of the robot's abilities, assuming perfectly rigid joints in the directions producing no motion.

vertex search algorithm VEPOLI^2, described in Section 3.4, can be used to efficiently find the limits (vertices) f_{min} and f_{max} of the applicable force in z-axis direction.

Furthermore, this procedure can be done for both robots r_1 and r_2, by characterising their carrying capacities $\mathcal{F}_{z,r_1}$ and $\mathcal{F}_{z,r_2}$ and finding their applicable vertical force ranges

$$f_{r_1} \in [f_{r_1,min},\ f_{r_1,max}], \quad f_{r_2} \in [f_{r_2,min},\ f_{r_2,max}] \tag{4.8}$$

Additionally, their combined carrying capacity $\mathcal{F}_{z,r_1+r_2}$ can be calculated as the Minkowski sum of their polytopes $\mathcal{F}_{z,r_1}$ and $\mathcal{F}_{z,r_2}$

$$\mathcal{F}_{z,r_1+r_2} = \mathcal{F}_{z,r_1} \oplus \mathcal{F}_{z,r_2}$$

or in this special case ($m = 1$), the sum of their intervals

$$f_{r_1+r_2} = f_{r_1} + f_{r_2} \in [f_{r_1,min} + f_{r_2,min},\ f_{r_1,max} + f_{r_2,max}].$$

In the context of this experiment, the real-time carrying capacity calculation is implemented using the Python open-source library `pycapacity`, developed in the context of this thesis and described in Chapter 7.

4.2.2 Collaborative robot control strategy

The collaborative carrying task requires the robot's to apply forces that compensate for the object's weight G, which can be expressed in a form

$$f_{r_1} + f_{r_2} - G = 0 \tag{4.9}$$

In the general case, there exists an infinity of ways the weight can be distributed between the two robots, satisfying equation (4.9) and, at the same time, respecting their carrying capacities $\mathcal{F}_{z,r_1}$ and $\mathcal{F}_{z,r_2}$.

In this work, a simple weight distribution strategy is proposed, where the relative load of each of the robots, with respect to their carrying capacity is evenly distributed.

$$\frac{f_{r_1}}{f_{r_1,max}} = \frac{f_{r_2}}{f_{r_2,max}} \tag{4.10}$$

In other words, the weight the robot carries is proportional to its carrying capacity. Combining equations (4.9) and (4.10), the weight carried by each one of the robots can be expressed as

$$f_{r_1} = \underbrace{\frac{f_{r_1,max}}{f_{r_1,max} + f_{r_2,max}}}_{\lambda_1} G, \qquad f_{r_2} = \underbrace{\frac{f_{r_2,max}}{f_{r_1,max} + f_{r_2,max}}}_{\lambda_2} G \tag{4.11}$$

where λ_1 and λ_2 represent the ratios of both robot's contribution to the overall carrying capacity, and their relationship can be expressed as $\lambda_1 + \lambda_2 = 1$.

In order to find the optimal forces f_{r_1} and f_{r_2} that, at the same time, compensate for the weight of the object (4.9), comply with the proposed weight distribution strategy (4.11) and respect their carrying capacities (4.8), a Quadratic Program (QP) based robot control strategy

is proposed

$$
\begin{aligned}
f_{r_1}, f_{r_2} = \underset{f_{r_1}, f_{r_2}}{\arg\min} \; & \overbrace{||G - f_{r_1} - f_{r_2}||^2}^{\text{weight compensation}} \; + \overbrace{\omega\lambda_2||f_{r_1}||^2 + \omega\lambda_1||f_{r_2}||^2}^{\text{weight distribution}} \\
s.t. \quad & f_{r_1} \in [f_{r_1,min}, f_{r_1,max}] \\
& f_{r_2} \in [f_{r_1,min}, f_{r_2,max}]
\end{aligned}
\tag{4.12}
$$

The proposed optimisation problem consists in two tasks. The first task is the weight compensation task, corresponding to equation (4.9), which is ensured by minimising the error $||G - f_{r_1} - f_{r_2}||^2$. Expressing equation (4.9) in this form, allows for the cases when the robot's carrying capacities are not sufficient for carrying the object $f_{r_1,max} + f_{r_2,max} < G$. Preventing such situations and ensuring the feasibility of the robots' tasks presents a substantial research challenge in the realm of robot task design.

In the context of collaborative carrying task, a potential avenue in addressing this issue comes from wrench feasibility analysis [18, 19]. These tools could enable quantifying the areas of the collaborative workspace, where a certain weight compensation capacity is guaranteed. Then, the collaborative carrying tasks could be designed within these ares in the workspace. It's worth noting that these approaches complement the proposed approach and could be used in tandem to further improve the robustness of the proposed method and bring it a step closer to practical real-world applications.

The second task ensures the even relative load distribution between the robots, as proposed by the weight distribution strategy given in equation (4.11). To achieve this behaviour, the norm of the first robot's force f_{r_1} is weighted by the second robot's contribution ratio λ_2, while the second robot's force f_{r_2} is weighted by the first robot's contribution ratio λ_1. This way, if the robot r_1 has higher carrying capacity $f_{r_1,max} > f_{r_2,max}$, the force f_{r_1} is weighted (penalised) less in the cost function $\lambda_1 > \lambda_2$, ultimately resulting in higher values found by the optimisation problem for robot 1. On the other hand, if the carrying capacities of both robots is equal, then

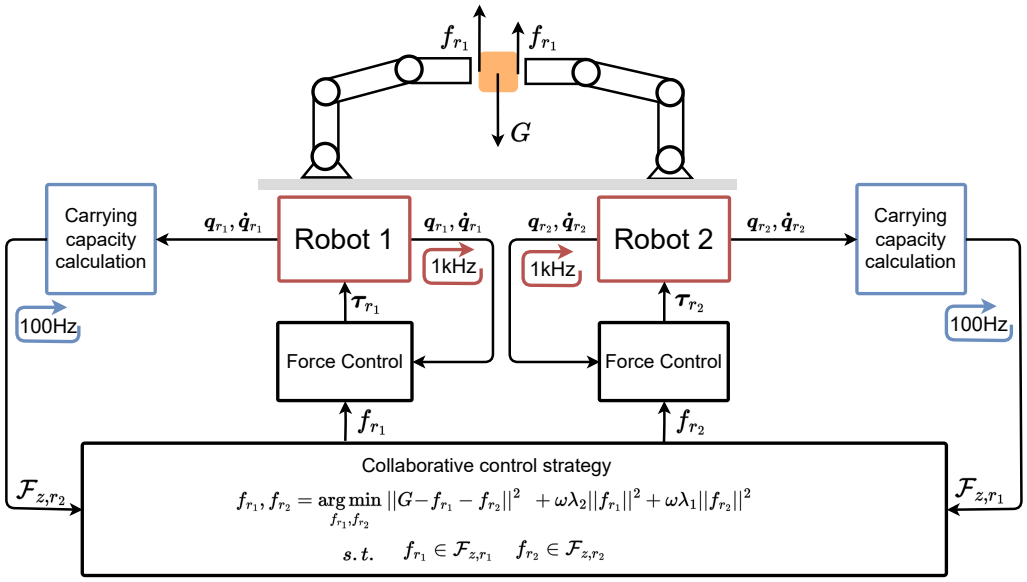

FIGURE 4.4: The block diagram showing the implemented robot control scheme. Each robot's low-level force control loop runs at 1kHz, while the carrying capacity calculation and the collaborative control strategy are evaluated at the frequency of 100Hz.

the weighing factors are equal too $\lambda_1 = \lambda_2$, resulting in the equal weight distribution.

The factor ω is used to set the priority of the weight distribution task much lower than the main task of the weight compensation of the object, in the experiments proposed in this sections the regularisation weight is set to $\omega = 1 \times 10^{-5}$.

Each robot control cycle starts by calculating the carrying capacities (4.8), followed by the resolution of the optimisation problem (4.12). Once the optimal vertical forces f_{r_i}, are obtained the joint torques $\boldsymbol{\tau}_{r_i}$, achieving those forces, are calculated for each of the robots and sent to their low-level joint controllers

$$\boldsymbol{\tau}_{r_i} = J^T_{z,r_i}(\boldsymbol{q}_{r_i}) f_{r_i} + \boldsymbol{\tau}_{b,r_i}(\boldsymbol{q}_{r_i}, \dot{\boldsymbol{q}}_{r_i}), \qquad \forall i \in \{1, 2\} \tag{4.13}$$

The calculation of the carrying capacity of both robots and the proposed collaborative control strategy are implemented in programming language Python and performed at the frequency of 100Hz. The robot's open-loop force control is implemented in programming language C++, using the library `pinocchio` [166], and run at the frequency of 1kHz. The complete software stack has been integrated using the Robot Operating System (ROS) [167] programming environment and run from one computer equipped with a 1.90GHz Intel i7-8650U processor. Figure 4.4 shows the schematic block diagram of the implemented collaborative robot control strategy.

4.2.3 Experimental validation

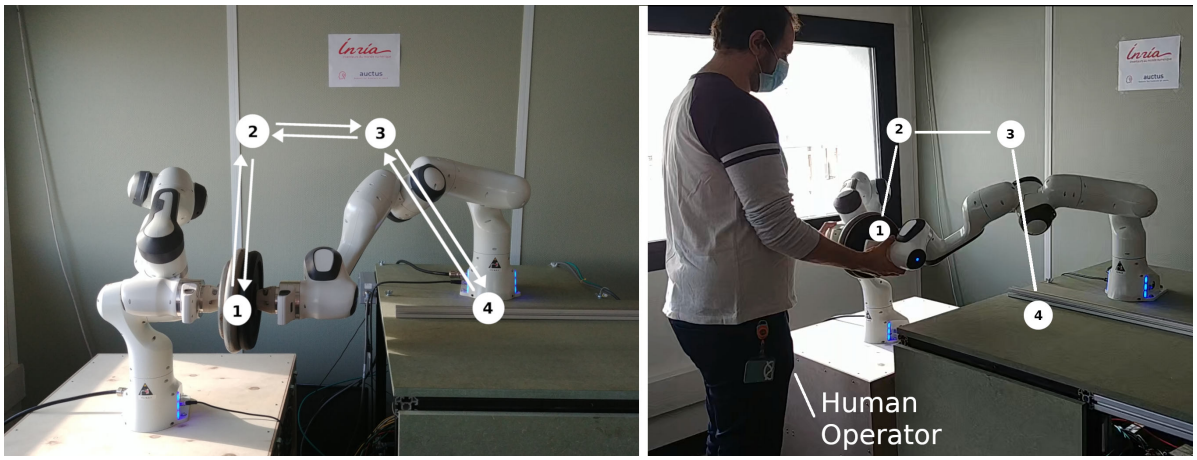

FIGURE 4.5: This figure shows the experimental setup and visualises the trajectory taken by the human operator. The robots compensate for the full object's weight while the operator moves the object through the workspace, doing a forward and backward pass through the 4 via-points.

In this experiment the two Panda robots jointly compensate for the weight of the object (12 kg), using the proposed collaborative control strategy. A human operator freely moves the object through the common workspace, implicitly changing both robots' configurations in real-time. As a consequence, the task space trajectory and the evolution of the robot configurations are not known in advance.

Video

The particular trajectory taken by the human operator in this experiment is indicated on Figure 4.5. It consists in taking the object through the set of four via-points, making a forward and backward pass. The proposed control strategy then leverages the real-time calculation of their carrying capacity and adapts

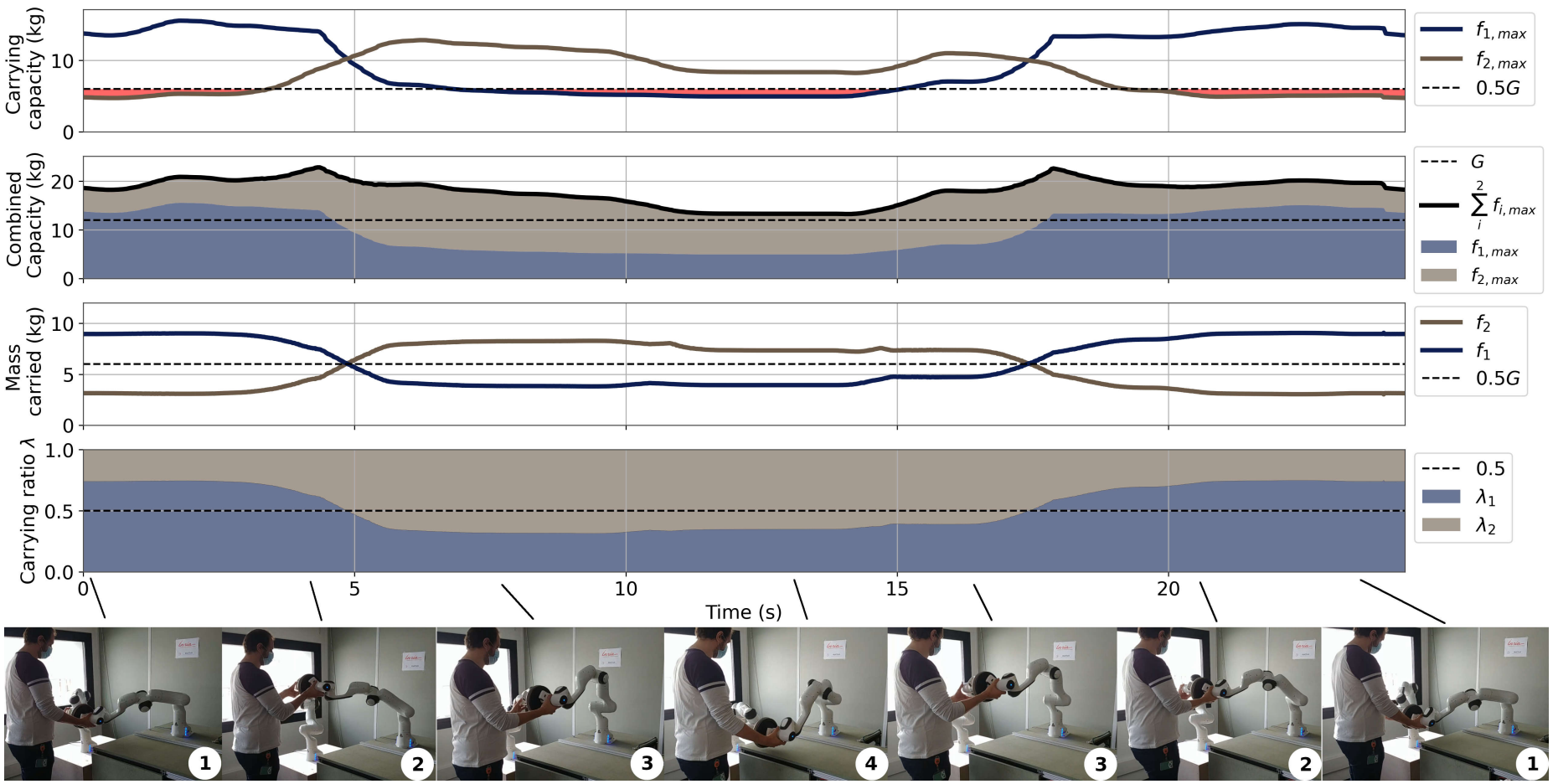

FIGURE 4.6: This figure shows the time evolution of different control and physical ability variables in the top graphs, as well as several images of the experiment taken when the operator attained different via-points with the object. The top graph shows the time evolution of the carrying capacity of the manipulators $f_{r_1,max}, f_{r_2,max}$, indicating, with red background, the time periods when the naive strategy $f_{r_1} = f_{r_2} = 0.5G$ (dashed line on all graphs) would not be feasible for at least one of the two robot. The second graph provides the evolution of their joint capacity $f_{max} = f_{r_1,max} + f_{r_2,max}$, indicating their individual contributions to the common capacity. The third graph indicates the time evolution of applied manipulator forces f_{r_1}, f_{r_2}, calculated using the proposed control strategy. Finally, the bottom graph shows the time evolution the weights λ_1, λ_2 used with the robot control. All the force values are expressed in kilograms for easier readability.

the weight distribution to ensure the compensation of the full object's weight during the whole experiment.

A straightforward control strategy, requiring each robot to compensate for half of object mass $f_{r_1} = f_{r_1} = 0.5G$ is taken as baseline to evaluate the performance of the new method. This experiment is illustrated in the accompanying video[3].

Figure 4.6 (bottom) shows several images taken during the experiment execution, in the moments when the operator reached different via-points. The graphs on the top show the time evolution of the evaluated maximal forces and force applied by each robot during the experiment. The naive strategy $f_{r_1} = f_{r_1} = 0.5G$ is shown in dashed lines for comparison.

The results show that the proposed control strategy is successful in ensuring compensation of the object weight during the full length of the experiment. In the starting via-point $t = 0$s and $t \geq 21$s, robot r_2 (robot on the right) is close to it's singular configuration and its load carrying capacity is close to 5kg. Controlling it to compensate for half of the weight of the object would result in a saturation of the actuators and potential damage to the hardware. The same is true for via-point 4 at $t = 12$s. In this via-point the robot r_1 (robot on the left) is not be able to compensate for half of the object weight, but thanks to the proposed adaptive control law, the task can successfully be achieved, supporting Concept 1 (C1) . Furthermore, any naive fixed strategy, assigning certain fixed ratio of the object's weight to the robots, would result in surpassing the robots' limits which in turn would fail to accomplishing the task. By exploiting the online calculation of the robots' physical abilities, the proposed strategy enables exploiting their changing abilities and accomplishing this task while at the same time guaranteeing that their capacities are not exceeded.

The results further show that even though in the large portion of the experiment one of the robots was not able to carry half of the object's weight, their joint carrying capacity exceeded the objects weight $f_{r_1,max} + f_{r_2,max} > G$ during the whole length of the experiment, conforming with Concept 2 (C2) .

However, this may not be the case over the entire common workspace of the two robots, but this example is a good illustration of the interest of accounting for the true capabilities of the system at the control level. In certain configurations the common carrying capacity of the robots might be lower than the objects weight $f_{r_1,max} + f_{r_2,max} < G$. The proposed control approach, in those cases would ensure that the robots' physical abilities are not exceeded, however it would not be able to guarantee the weight compensation of the object.

In summary, the experimental validation demonstrates the effectiveness of the proposed control strategy in real-time adaptation of weight distribution between multiple robots based on their varying physical abilities. The strategy ensures safe and efficient collaboration between robots, allowing them to collectively perform tasks that would be challenging or impossible for each robot to accomplish individually. Leveraging real-time information about their physical abilities, the proposed strategy offers a flexible solution for collaborative applications, eliminating the need for pre-calibration or in advance task-specific information.

4.3 Human-robot collaborative object carrying

In many industrial contexts, including the LiChiE project's mini-satellite fabrication process, tasks involving the manipulation of heavy objects are a frequent occurrence. Taking inspiration from these tasks, this section presents an illustrative scenario where a human and a robot

[3]Video: https://youtu.be/hApIv1oFuhk

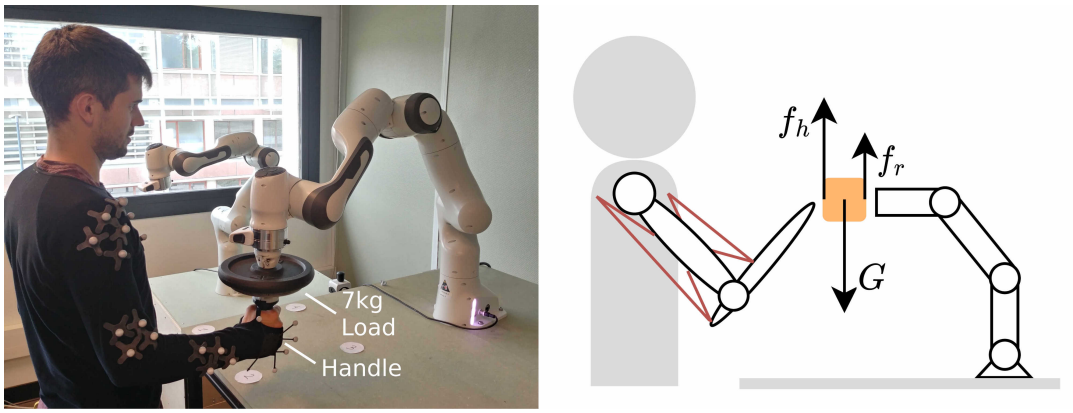

FIGURE 4.7: This figure show the experimental setup. The robot end-effector is fixed to the carried object and the human operator is holding the object by the handle. The object's mass is $m = 7$kg, while the operator and the robot collaborate to compensate the full weight of the object.

collaborate physically to carry a heavy object. In this scenario, shown on Figure 4.7, a human operator and a Franka Emika Panda robot jointly carry an object mass of $m = 7$ kg, each one compensating for a part of the total weight

$$\boldsymbol{f}_h + \boldsymbol{f}_r = m\boldsymbol{g} = G \tag{4.14}$$

In this experiment, the operator has the expert knowledge about the task and moves the object through the workspace to perform the task efficiently. The robot's role, in this case, is to assist the operator with the carrying of the heavy object and improve their safety and their overall well-being.

The usual choice for these scenarios is a simple gravity compensation strategy, making the robot carry the full weight of the object $\boldsymbol{f}_r = G$ and provide the operator with full movement transparency. Several such human-centred robot control strategies have recently been proposed in the literature, where the robot in addition to the full weight compensation improves different aspects of the human safety and ergonomics. One example of such approach is proposed by Ferraguti *et al.* [168], which REBA [45] ergonomic indicators to improve human posture while carrying an object.

However, as discussed in Section 4.2, assigning any fixed weight to the robotic manipulator requires *a priori* analysis of its changing capacity and determination of the *worst-case* weight limits, which in the case of the Franka Emika Panda robot is $m \leq 3$kg[4]. Therefore, carrying the 7kg load requires using a more capable robot which, in many cases, significantly increase the cost of such system and potentially reduces operator's safety. As more capable robots are usually bigger and heavier, they result in much higher exchanged forces in case of impact with the operator, raising many safety concerns [169]. Therefore, delegating the entire weight compensation to the robot is not always the most efficient or the safest solution for the operator.

On the other hand, several studies have shown that the human's physical feedback during task execution improves significantly their engagement in the task [170], improves the task execution efficiency and reduces the risk of error [171]. However, since the human's carrying capacity varies significantly throughout the workspace as well, before allocating any weight to the operator, their safety has to be ensured. As in the case of robots, the traditional approach consists in analysing the task in advance and determining, often in a very conservative manner,

[4]Given by the manufacturer in the form of the robot's rated payload.

the set of *worst-case* limits [172], often given in a form of different standards [49] and manuals [48].

In order to avoid making such coarse assumptions, several approaches have been proposed in the literature, able to quantify different measures of human physical abilities and adapt the human load to their changes online. Kim *et al.* [173] have proposed an approach estimating the joint torques in the operator's body produced by carrying the load of the object, where the robot's role is to adapt the operator's posture to minimise the magnitude of human's joint torques. The human body is approximated using a planar model actuated at the joint level. Furthremore, in their work, the human is the one carrying the full mass of the object, while the robot's only role of is to modify the operator's posture in order to improve his ergonomics.

A different approach is proposed by Carmichael and Liu [68], where the human and the robot collaborate physically to execute the carrying task. Their approach is employs on a more accurate human model, based on human musculoskeletal models, which is used to efficiently approximate the human's force capacity. The robot control strategy modulates the weight carried by the human with respect to the approximation calculated in real-time. The approach is developed in the context of human rehabilitation and used for the real-time control of assistive exoskeletons. The authors show that such approach is capable of adapting to the operator's pathology and reduce the rehabilitation time [174]. This approach demonstrates the potential of using human-centred robot control strategies which are able to adapt the load carried by the human precisely to his varying physical abilities. However due to the intractable computational complexity of the human force capacity calculation proposed by the authors, the scope of their work is limited to relatively simple planar musculoskeletal models. Furthermore, the robot control strategy proposed in their work is aimed to the rehabilitation and the use of assistive exoskeletons, which does not transpose well to more unstructured industrial scenarios.

Inspired by their work, this section aims to further demonstrate the potential of using the real-time knowledge about both robot's and human's physical abilities in the context of collaborative robot control in industrial scenarios. The proposed collaborative robot control strategy leverages the two new algorithms developed in the context of this work, capable of efficiently calculating the force capacity of robots (VEPOLI^2 described in Section 3.4) and humans, based on their musculoskeletal models (ICHM described in Section 3.5). Their changing capacities are calculated online and used to create collaborative robot control strategy, capable of both exploiting their full potential and, at the same time, ensure the safety of the operator.

Section 4.3.1 introduces an efficient strategy for calculating the operator's carrying capacity in real-time. Section 4.3.2 introduces the collaborative robot control strategy using the online information about the operator's carrying capacity, as well as the robot's capacity, described in Section 4.2.1. The experimental validation of this approach is described in Section 4.3.3 as well as the discussion on results.

4.3.1 Human carrying capacity calculation

As described in Section 2.2, the analysis of human's physical abilities in this work is based on their musculoskeleral models, as the most accurate models of human bodies currently available. More specifically, as shown on Figure 4.7, this experiment requires evaluating the operator's ability to carry the object in his right arm. Many different human arm musculoskeletal models are proposed and validated in the biomechanics literature [158, 175], as well as different software tools for their analysis [146, 176]. However, choosing an appropriate musculoskeletal, and adapting it to individual characteristics of the operator, is a challenging scientific problem.

In broad terms, using more detailed musculoskeletal models often leads to more accurate representation of human subjects [177], but comes with an increase in the computational complexity.

The computational efficiency of the proposed ICHM algorithm, described in Section 3.5, allows for considering relatively detailed models (even up to 50 muscles) while maintaining interactive execution times. Therefore, the analysis in this section assumes having an appropriate and detailed musculoskeletal model of the operator's arm.

When it comes to the collaborative carrying experiment, the operator is required to apply only the force f_h that contributes to compensating the object's gravity G. Therefore, the task is effectively one dimensional $m = 1$ and his physical ability to carry certain weight can be evaluated as the maximal applicable force in the vertical (z-axis) direction. Hence, the human's carrying capacity $\mathcal{F}_z$ can be expressed as the the special case of the human's wrench polytope $\mathcal{P}_f$, described in Section 2.2.3, where the task space is one dimensional ($m = 1$)

$$\mathcal{F}_z(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \mathcal{P}_{f_z}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \{f \in \mathbb{R} \mid J_z(\boldsymbol{q})^T f = N(\boldsymbol{q})\boldsymbol{F} - \boldsymbol{\tau}_b(\boldsymbol{q}, \dot{\boldsymbol{q}}), \quad \boldsymbol{F} \in [\boldsymbol{F}_{min},\ \boldsymbol{F}_{max}]\} \quad (4.15)$$

where $\{\boldsymbol{q}, \dot{\boldsymbol{q}}\}$ represent the human's joint state, $f \in \mathbb{R}$ is the applicable scalar force in z-axis direction, $\boldsymbol{F} \in \mathbb{R}^n$ are the applied muscular forces limited within the interval $\boldsymbol{F} \in [\boldsymbol{F}_{min},\ \boldsymbol{F}_{max}]$, $\boldsymbol{\tau}_b \in \mathbb{R}^n$ represents the bias joint torques grouping the effects of the gravity and human's motions, while $J_z \in \mathbb{R}^{1\times n}$ is the configuration dependant Jacobian matrix with one line and n columns and $N \in \mathbb{R}^{n\times d}$ is the state dependant moment arm matrix. Values n and d correspond to the number of joints and muscles considered in the musculoskeleral model.

Analogously to the robot's case, human's carrying capacity $\mathcal{F}_z$ is a $m = 1$ dimensional polytope, describing the range of scalar forces f applicable in the vertical direction. Geometrically, this polytope corresponds to the intersection of the complete wrench/force ($m = 6/m = 3$) polytope $\mathcal{P}_f$, described in Section 2.2.3, with the vertical axis. An illustrative view of this intersection is shown on Figure 4.8.

Furthermore, for any given human's state $\{\boldsymbol{q}, \dot{\boldsymbol{q}}\}$ this one dimensional set $\mathcal{F}_z$ can be transformed into the form of a min-max interval

$$\mathcal{F}_z = \{f \in \mathbb{R} \mid f \in [f_{min},\ f_{max}]\} \quad (4.16)$$

In order to find the limits $[f_{min},\ f_{max}]$ of the applicable scalar force f in the vertical direction, this problem can be formulated as finding the $\mathcal{V}$-representation of the 1D ($m = 1$) polytope $\mathcal{F}_z$. Then the real-time capable algorithm ICHM, proposed in Section 3.5, can then be used to find the limits (vertices) f_{min} and f_{max} of the applicable force in z-axis direction.

Following the similar procedure as for humans, the robot's carrying capacity can be determined in real-time as well, using the VEPOLI^2 algorithm, as proposed in Section 4.2.1. Then the human's carrying capacity $\mathcal{F}_{z,h}$ and robot's carrying capacity $\mathcal{F}_{z,r}$, characterising their feasible ranges of applicable vertical forces f_r and f_h can be expressed as in the form of ranges

$$f_h \in [f_{h,min},\ f_{h,max}], \qquad f_r \in [f_{r,min},\ f_{r,max}] \quad (4.17)$$

Additionally, their combined carrying capacity $\mathcal{F}_{z,h+r}$ can be calculated as the Minkowski sum of their carrying capacities $\mathcal{F}_{z,h}$ and $\mathcal{F}_{z,r}$

$$\mathcal{F}_{z,h+r} = \mathcal{F}_{z,h} \oplus \mathcal{F}_{z,r}$$

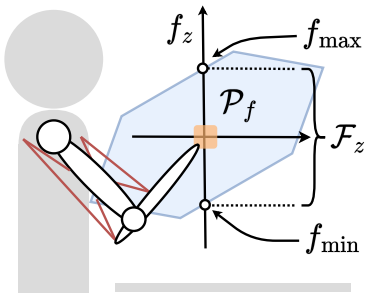

FIGURE 4.8: A geometric view of constructing the carrying capacity $\mathcal{F}_z$ as the intersection of the force polytope $\mathcal{P}_f$ and the z-axis. The carrying capacity is expressed at the center of the object.

or in this special ($m = 1$), the sum of the intervals

$$f_{h+r} = f_h + f_r \in [f_{h,min} + f_{r,min},\ f_{h,max} + f_{r,max}].$$

As in the previous experiment, the real-time carrying capacity calculation, using VEPOLI^2 and ICHM algorithms, is implemented using the Python open-source library `pycapacity`. This library is developed in the context of this thesis in Chapter 7.

4.3.2 Collaborative robot control strategy

The collaborative carrying task requires the human and the robot to apply forces that combined compensate for the object's weight G

$$f_h + f_r = G$$

As their carrying capacity changes in time, the main task of the robot's control algorithm consists in modulating the weight carried by the operator and the robot to ensure that their carrying capacity is not exceed while compensating for the object's weight.

In addition to the object's weight compensation, inspired by the Assist-As-Needed (AAN) paradigm proposed by Carmichael and Liu [68], a simple adaptive weight distribution strategy is proposed, with the goal to ensure that the human operator's relative load remains constant with respect to its real-time capacity.

$$\frac{f_h}{f_{h,max}} = const.$$

The fixed ratio chosen for the experiments is 30% of the human's carrying capacity, ensuring operator's safety and maintaining the constant level of engagement.

$$f_h = 0.3 f_{h,max}$$

It is worth acknowledging that determining the optimal ratio is a challenging scientific problem, one that may require individual determination for each human subject. In the context of this work, the value of 30% is selected arbitrary.

The main task of the object's weight compensation as well as the AAN strategy, maintaining the constant relative load of the operator, is implemented in a single robot control strategy based on Quadratic Program (QP)

$$\begin{aligned} f_r, f_h = \arg\min_{f_r, f_h} \ & \overbrace{||G - f_r - f_h||^2}^{\text{weight compensation}} + \overbrace{\omega||f_r||^2}^{\text{weight distribution}} \\ s.t. \quad & f_h \in [0.3 f_{h,min},\ 0.3 f_{h,max}] \\ & f_r \in [f_{r,min},\ f_{r,max}] \end{aligned} \tag{4.18}$$

The main component of the cost function of this QP is the minimisation of the error $||G - f_h - f_r||^2$, which ensures that the object's gravity is always compensated, as long as their carrying capacity defined in the constraints of the optimisation problem allows for it.

The AAN strategy is formulated as a regularisation task $\omega||f_r||^2$ in the cost function. This task minimises the magnitude of force f_r applied by the robot while having no penalisation for the human's force f_h. The resulting behaviour of the optimisation problem is to find the weight distribution between the human and the robot which results in the minimal force f_r applied

by the robot, while maximising the force of the operator f_h. In the other words, this weight distribution strategy produces the AAN behaviour where the human applies its maximal forces (30% of its carrying capacity) while the robot compensates for the rest. More precisely, the force the human applies is equal either to 30% of its carrying capacity or the full objects weight

$$f_h = \begin{cases} 0.3 f_{h,max}, & \text{if } G \geq 0.3 f_{h,max} \\ G, & \text{otherwise} \end{cases}$$

while the robot carries the remaining weight, as long as it is capable of doing so

$$f_r = \begin{cases} G - f_h, & \text{if } G - f_h \leq f_{r,max} \\ f_{r,max}, & \text{otherwise} \end{cases}$$

It is worth noting that, if the object's weight exceeds the common carrying capacity of the robot and the operator

$$f_{r,max} + 0.3 f_{h,max} < G$$

they are not able to compensate for the weight. In such scenarios, the proposed strategy will prioritise ensuring that their capacities are not exceeded, over the weight compensation.

The regularisation factor ω is used to set the priority of the AAN task lower than the main task of the weight compensation of the object. In the experiments proposed in this section the regularisation weight is set to $\omega = 1 \times 10^{-2}$.

In each control cycle human's $\mathcal{F}_{z,h}$ and robot's $\mathcal{F}_{z,r}$ carrying capacities are calculated. These values are then used for the resolution of the optimisation problem (4.18). Once the robot's optimal vertical force f_r is obtained, the joint torques $\boldsymbol{\tau}_r$ achieving this force are calculated

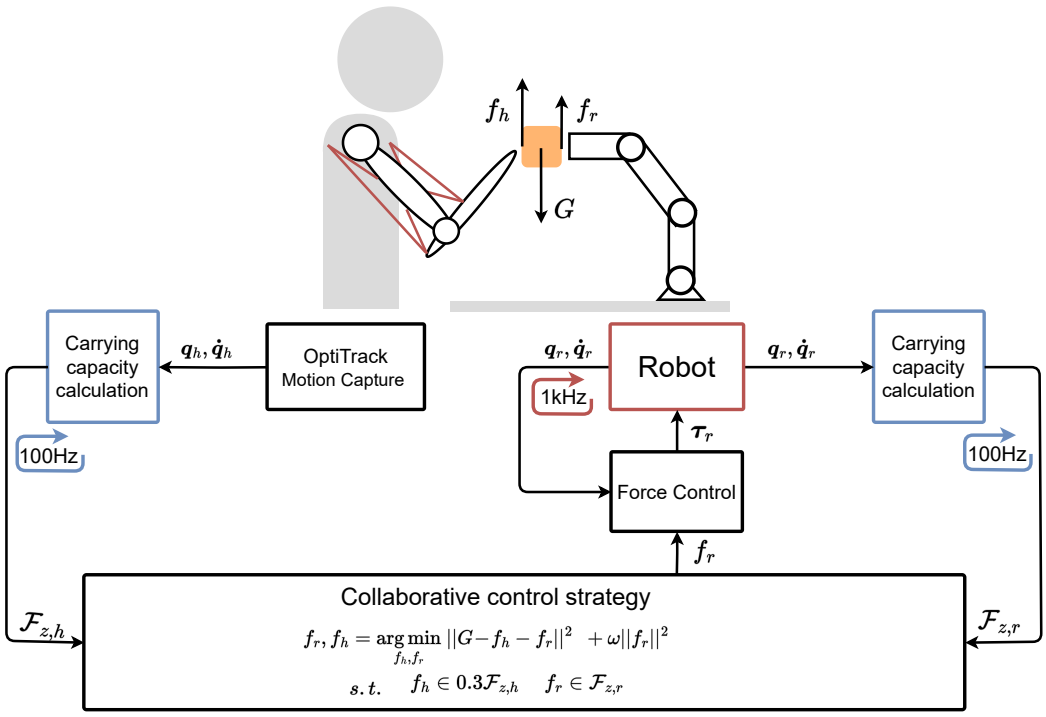

FIGURE 4.9: Block diagram showing the implemented robot control scheme. The robot's low-level force control loop runs at 1kHz, while the carrying capacity calculation of both robot and the operator, as well as the collaborative control strategy, are evaluated at the frequency of 100Hz. Human's posture is obtained using the motion capture system OptiTrack.

using the expression

$$\boldsymbol{\tau}_r = J_{z,r}^T(\boldsymbol{q}_r) f_r + \boldsymbol{\tau}_{b,r}(\boldsymbol{q}_r, \dot{\boldsymbol{q}}_r)$$

where $\{\boldsymbol{q}_r, \dot{\boldsymbol{q}}_r\} \in \mathbb{R}^{n_r}$ is the robot's n_r dimensional state, $J_r \in \mathbb{R}^{\times n_r}$ is the end-effector Jacobian with 1 line and n_r columns, $\boldsymbol{\tau}_{b,r} \in \mathbb{R}^{n_r}$ is robot's torque vector corresponding to the effects of the gravity and the robot's motion and $\boldsymbol{\tau}_r \in \mathbb{R}^{n_r}$ are the joint torques sent to the robot's low-level joint control.

The human's force f_h, calculated by the proposed control strategy, corresponds to the force the operator should apply in order for the object to stay static in the space. As the weight compensation is both operator's and robot's task, the operator will naturally tend to apply the force f_h. Therefore, the proposed control strategy indirectly closes the loop with the human operator through the forces applied on the object.

Both robot's and human's carrying capacity calculation algorithms, as well as the robot control strategy described in Section 4.3.2, are implemented in Python and run in real-time with the update frequency of 100Hz. The low-level robot control is implemented in C++, using the open-source library `pinocchio` [166], and run in real-time at the frequency of 1kHz. All the software components are integrated using the Robot Operating System (ROS) [167] and run on a computer equipped with a 1.90GHz Intel i7-8650U processor. Figure 4.9 shows the schematic block diagram of the implemented collaborative robot control strategy.

4.3.3 Experimental validation

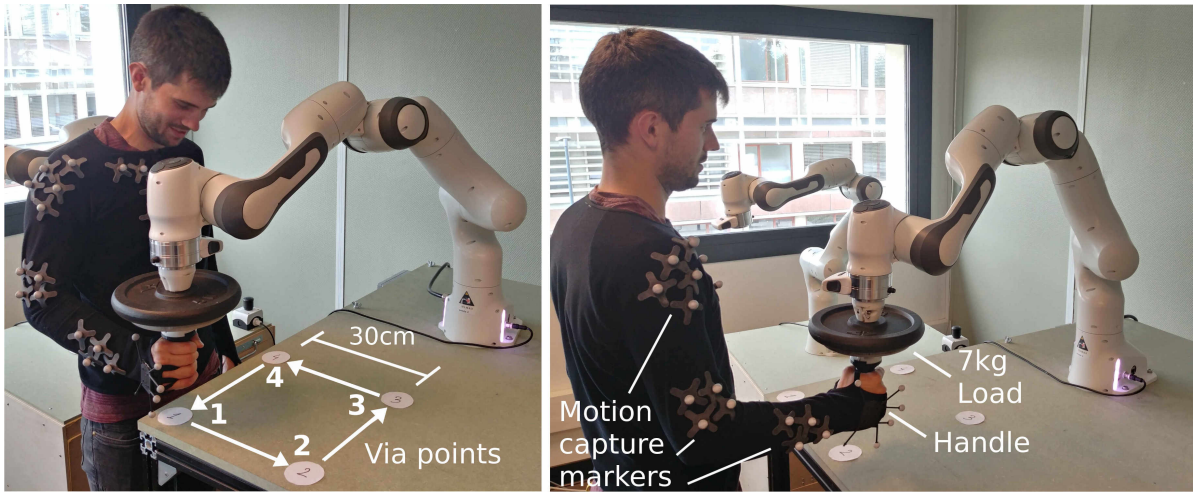

FIGURE 4.10: These figures show the experimental setup. The robot end-effector is fixed to the carried object and the human operator is holding the object by the handle. The motion capture system is used to acquire the pose of the human arm. The via-points are visually indicated to the human operator with numbered stickers on the table placed at the corners of a 30 cm square. The images shows the experiment 1 (left) and the experiment 2 (rigth).

In this experiment, the human operator performs a task by visually navigating the object through the common workspace, passing through the set of 4 via-points. The object is fixed in the end-effector of the robot as well as at the hand of the human operator using a handle. Figure 4.10 shows the experimental setup indicating the structure of the collaborative workstation, the placement of the via-points and the handle.

Video

The experiment is repeated two times, for two different operator positions, equally distant from the via-points, but rotated by 90° in space. These

operator positions are indicated on Figure 4.10 as well. Furthermore, in each of the two experiments, the operator performs 10 cycles through the via-points. The experiment is illustrated in the accompanying video[5].

The musculoskeletal model used in this experiment is the MOBL-ARMS model [158] of the human upper limb (right arm), developed by Holzbaur *et al.* [175]. A relatively detailed model consisting in 50 muscles ($d = 50$) and 7 degrees of freedom ($n = 7$). The visualisation of this model, using the biomechanics software Opensim [176], is shown on Figure 4.11. The operator's arm configuration is inferred in real-time using an OptiTrack [178] motion capture system. The musculoskeletal model is obtained using the efficient biomechanics library `biorbd` [159].

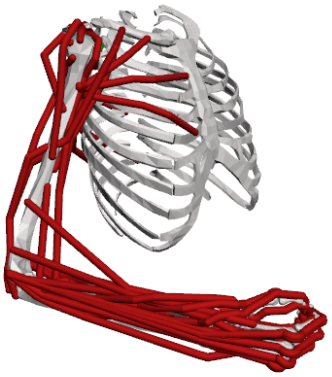

FIGURE 4.11: Human right arm musculoskeletal model MOBL-ARMS [158], with 50 muscles and 7 joints. Visualised using OpenSim software [176].

The human subject (male, 182 cm, 80 kg) and the MOBL-ARMS model belong to the same 50th percentile anthropomorphic group [179] so no model adaptation is necessary (ex. scaling [180]).

Figure 4.12 shows the recorded time evolution of robot's and human's carrying capacities and weight carried during the 10 cycles for each of the two experiments.

The figure shows (top graphs) that the human's carrying capacity is much more variable than the robot's one, peaking around 20kg in the recorded experiments and going as low as 5kg, emphasising the importance of measuring it in real-time, and supporting Concept 1 (C1) . Furthermore, the experiment's show that the human's carrying capacity is in many cases much higher than the robot's capacity, confirming the importance of creating the control strategies able to account for both of their physical abilities.

The second line of graphs on Figure 4.12 shows the time evolution of the sum of their carrying capacities, their common carrying capacity as one system. The graphs show that even though both the robot and the human, in multiple instances during the experiment runs, had carrying capacity lower than the object's weight G, their joint carrying capacity exceeds the object's weight during the whole time of the experiment. Therefore, the proposed method demonstrates the feasibility of using physical ability aware control strategies to improve the performance of the physical collaboration. This confirms Concept 2 (C2) by showing that, even though, neither the operator or the robot would have been able to carry the object on their own, by collaborating, they were able to execute this task.

It is worth noting that the objective of this experiment is to demonstrate the feasibility rather than guarantee universal applicability. In more general situations, there could be scenarios where the collective carrying capacity of the human and the robot is insufficient to support the object's weight ($f_{h,max} + f_{r,max} < G$). In such cases, the introduced control strategy prioritises ensuring that both actors capacities are not exceeded rather than the complete compensation for the object's weight.

The third line of graphs shows the time evolution of the weight carried by the robot and the human, while the bottom graphs on Figure 4.12, show the relative load of the robot l_r and the

[5]Video: https://youtu.be/wg4E62AkNnM

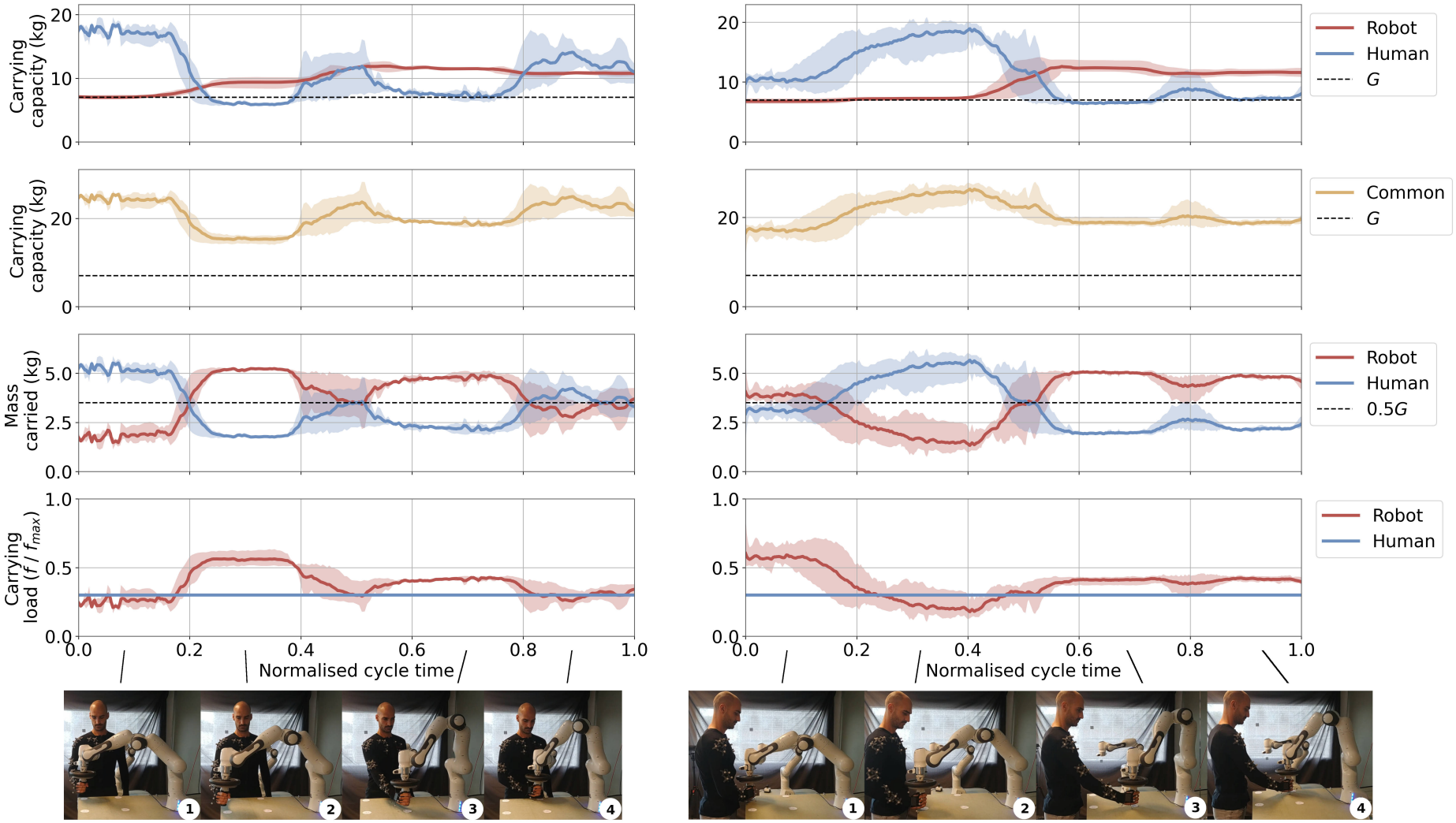

FIGURE 4.12: This figure shows the evolution of the carrying capacity and weight carried by the operator and the robot over the course of the via-point cycle, for two experiments. The plots show mean values and the variances of the curves calculated over 10 successive cycles. The graph regions belonging to the different via-points are separated by vertical lines. All the values are expressed in kilograms for easier readability.

human l_h calculated as

$$l_h = \frac{f_h}{f_{h,max}}, \qquad l_r = \frac{f_r}{f_{r,max}}$$

The graphs confirm that the proposed control strategy maintained the human's relative load at 30% ($l_h = 0.3$) of its capacity as proposed by the Assist-As-Needed (AAN) strategy, while the robot's relative load l_r depended on the human capacity. As the human's carrying capacity decreased robot's load l_r increased, and *vice-versa*. By having the real-time knowledge about the operator's carrying capacity, the robot was able to adapt its assistance level to maintain human's load constant, despite substantial changes in the operator's capacity profile and despite his changing placement placement with respect to the task, supporting the Concept 3 (C3) .

4.4 Discussion and perspectives

This chapter argues for the use of accurate real-time information about human's and robot's physical abilities for creating collaborative robot control strategies. To demonstrate this concept, the chapter proposes two proof-of-concept experimental scenarios in the context of human-robot physical collaboration.

As discussed in Section 4.2, the dual robot collaborative carrying experiment shows that the proposed strategy enables to fully exploit the robots' changing physical abilities in real-time. Additionally, the strategy enables ensuring that their safety is not compromised, even though the wight carried by the robots (12kg) is two times higher than their rated payload (2×3kg). Therefore, such real-time capacity-aware control strategies have the potential to enable leveraging the true physical abilities of robots more optimally and even potentially help make a step towards more minimal robot design.

Section 4.3 showcased the human-robot collaborative carrying experiment. For this scenario a new human-centred robot control strategy is proposed, leveraging the information about both of their physical abilities in real-time. The experiment shows that the strategy successfully distributes weight between the human and the robot, making sure that their abilities are not surpassed, despite their substantial changes happening in real-time. At the same time, the strategy ensures the operator's safety by maintaining the operator's relative load constant (at 30% of its capacity), inspired by the Assist-As-Needed (AAN) paradigm [68].

As the main focus of the experiment is to demonstrate the feasibility of the concept of capacity-aware collaborative robot control, it is important to note that the experiment is conducted with a single participant. Even though the benefits of similar AAN strategies have been experimentally confirmed in the rehabilitation literature [70, 174], their application in the collaborative scenarios is not well studied. Therefore, in order to go a step further and evaluate the utility of such human-centred strategies in more practical applications (and their parameters), a more extensive user study would be required.

Furthermore, the carrying task, used in the experiments, is relatively simple and essentially one dimensional. However the proposed capacity-aware approaches are not limited to such simple use-cases. Section 4.4.1 discusses the perspectives of these approaches to be used in more complex human-robot physical collaboration scenarios. Moreover, the experiments demonstrated that the computational efficiency of the ICHM algorithm allows for using a relatively detailed human musculoskeletal model of operators to evaluate their physical abilities in real-time. Section 4.4.2 discusses the benefits and limitations of using such detailed human models in the context of human-robot collaboration.

4.4.1 Perspective: Extension to more flexible assistive strategies

The proposed human-robot collaborative carrying experiment, as discussed in Section 4.3.3, has two main limitations. Firstly, it only accounts for forces in the vertical (z-axis) direction, limiting the task to one-dimensional ($m = 1$) space. This simplification reduces computational complexity but does not exploit full polytope geometry of the human's and robot's physical abilities. Secondly, the wrenches ($\boldsymbol{f}_h$) applied by the human operator are not measured in the experiment, simplifying the scenario by assuming all operator forces are in the z-axis direction. However, in the absence of wrench $\boldsymbol{f}_r$ measurements, the operator receives no assistance for forces orthogonal to the z-axis.

One of the promising approaches able to address these limitations is proposed by Petrič *et al.* [70]. The authors proposed an assistive strategy based on human's force ellipsoids, where the robot's task consists in making the operator's force capacity isotropic (equal in all the directions in space). The forces applied by the operator are measured in real-time and amplified by the robot. The amplification factor is determined with respect to the operator's force capacity in the direction of the applied force. The author's show that such strategy does not restrict the operator's movements while significantly reducing the effort. However, their work is based on force ellipsoids which are a coarse under-approximation of the true operator's force capacity. Moreover, their approach is created around a very specific planar exoskeleton setup in the rehabilitation context. Therefore, this section discusses an extension of this approach to the polytope representation of physical abilities and to more flexible collaboration scenarios.

Such human-centered assistive robot control strategy could enable providing the operator with adapted assistance in any spatial direction, by leveraging the real-time measurement of the operator's wrenches $\boldsymbol{f}_h$. Moreover, the method could allow for exploiting the full polytope geometry of human's and robot's wrench polytopes ($\mathcal{P}_{f,h}$ and $\mathcal{P}_{f,r}$). Finally, such method could allow to shape the operator's wrench capacity given a desired polytope $\mathcal{P}_{f,d}$, opening doors for more flexible collaboration behaviours.

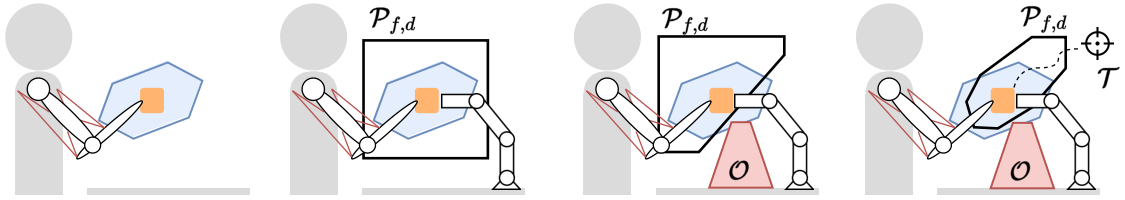

FIGURE 4.13: This figure illustrates the proposed assistive control strategy based on the desired polytope $\mathcal{P}_{f,d}$ modulation. The first figure on the left shows the human's wrench polytope $\mathcal{P}_{f,h}$ with no robotic assistance. The second figure shows the case where the robotic assistance leads to the isotropic rectangle shape of the desired polytope $\mathcal{P}_{d,f}$. The third figure shows the scenario where the desired polytope is modulated in order to prevent collision with the obstacles $\mathcal{O}$. The last figure shows the scenario where the desired polytope $\mathcal{P}_{f,d}$ shape takes in account the obstacles and guides the operator in the executing the task $\mathcal{T}$.

By setting an appropriate shape of the desired polytope $\mathcal{P}_{f,d}$, different collaborative behaviours can be obtained. For example, if no information about the task and the environment is available, then the shape $\mathcal{P}_{f,d}$ can be set to a m-dimensional hypercube. In this scenario, the robot's assistance makes the operator's force capacity isotropic (equal in all the directions of the space). On the other hand, if the information about the obstacles in the environment is available, the shape of the desired polytope $\mathcal{P}_{f,d}$ can be reduced in the directions of obstacles, preventing the operator to get in contact with them. Finally, if the operator's task requires executing a trajectory, known in advance, the shape of the desired polytope $\mathcal{P}_{f,d}$ can be chosen to guide the operator toward the successful task execution. Figure 4.13 illustrates the discussed

behaviours through of the desired polytope $\mathcal{P}_{f,d}$ modulation.

Implementing the adaptive force amplification approach

For a given operator wrench $\boldsymbol{f}_h$, characterised by magnitude f_h and direction vector $\boldsymbol{u}$, both operator and robot force capacity intervals can be defined: $f_h \in [f_{h,min},\ f_{h,max}]$ and $f_r \in [f_{r,min},\ f_{r,max}]$, as detailed in Section 4.3.1 and Section 4.2.1. Using the same vector $\boldsymbol{u}$, the maximum desired range $[f_{d,min},\ f_{d,max}]$ within the polytope $\mathcal{P}_{f,d}$ can be determined. Figure 4.14 illustrates the geometrical representation of these ranges.

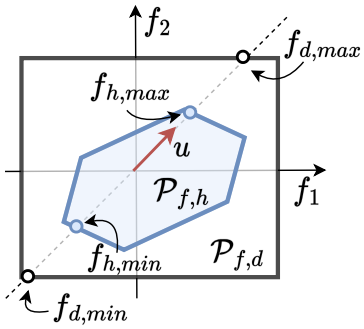

FIGURE 4.14: Finding the maximal ranges of human's and desired wrench capacity in direction $\boldsymbol{u}$.

The proposed assistive robot control strategy consists in amplifying the operator's wrench as follows

$$\boldsymbol{f}_r = \underbrace{\alpha f_h}_{f_r} \boldsymbol{u} \tag{4.19}$$

Here, α represents a scalar amplification factor and f_r is the magnitude of the robot's wrench applied in the direction $\boldsymbol{u}$. The final force applied jointly by the robot and the human is then

$$f_h + f_r = (1+\alpha) f_h \tag{4.20}$$

In order to amplify the operator's force capacity $f_{h,max}$ in the direction $\boldsymbol{u}$, and match the desired capacity $f_{d,max}$, the factor α can be determined using the ratio

$$\frac{f_h}{f_{h,max}} = \frac{(1+\alpha) f_h}{f_{d,max}} \tag{4.21}$$

The final expression for the amplification factor α becomes

$$\alpha = \frac{f_{d,max}}{f_{h,max}} - 1 \tag{4.22}$$

To calculate the amplification factor α (4.22), while at the same time ensuring that the robot's wrench capacity is not exceeded, a Quadratic Program (QP) can be formulated

$$\begin{aligned} \alpha = \arg\min_{\alpha} \ & \left|\left| \alpha - \frac{f_{d,max}}{f_{h,max}} + 1 \right|\right|^2 \\ s.t. \quad & f_r = \alpha f_h \\ & f_r \in [f_{r,min},\ f_{r,max}] \end{aligned} \tag{4.23}$$

For any human wrench ($\boldsymbol{f}_h = f_h \boldsymbol{u}$), the assistive robot control strategy (4.23) allows to calculate the optimal force amplification α, shaping the operator's wrench capacity $\mathcal{P}_{f,h}$ into the desired shape $\mathcal{P}_{f,d}$, while respecting the robot's wrench capacity $\mathcal{P}_{f,r}$.

A parallel with the impedance control

Consider that the operator and the robot jointly manipulate an object with a mass m. The wrenches they apply onto the object are combined and produce its motion in space

$$\boldsymbol{f}_h + \boldsymbol{f}_r = (1+\alpha)\boldsymbol{f}_h = m\ddot{\boldsymbol{x}} \tag{4.24}$$

where $\ddot{\boldsymbol{x}}$ is the object's Cartesian Space (CS) acceleration (including gravity) and α is the amplification factor described in the previous section. This equation can be rearranged in a form

$$\boldsymbol{f}_h = \frac{m}{1+\alpha}\ddot{\boldsymbol{x}} \tag{4.25}$$

permitting to see the proposed assistive approach as a special case of the impedance control, allowing to modulate the perceived mass m of the object to the operator. This approach could enable the intuitive modulation of the perceived mass of the object through the desired polytope $\mathcal{P}_{f,d}$, providing the operators with physical feedback about the task and their environment. Additionally, the object's mass can be modulated to account for the changing physical abilities of the operator $\mathcal{P}_{f,h}$ and the robot $\mathcal{P}_{f,r}$.

However, in the context of this thesis, the development of this approach stayed in the conceptual stage, leaving it as a potential avenue for future research and exploration.

4.4.2 Perspective: On using detailed musculoskeletal models

The ICHM algorithm, proposed in Section 3.5, presents a new and efficient way of evaluating different polytopes of physical abilities based on human's musculoskeletal models. This algorithm opens many possibilities for wider use of human physical ability polytopes in the area of the human-robot collaboration, by both reducing the computation time and enabling the use of more complete human models, better describing human subjects [177].

The algorithm's real-time ability may enable higher degree of human-centred robot control, where the operators' accurate real-time capacity, based on the detailed musculoskeletal models, could be used not only to enforce safety and improve engagement, but to provide the operator-specific assistance profiles. Such detailed musculoskeletal models could help preventing the development of the work related injuries, by ensuring that the operator's load well suited to its abilities as well as by including the information about his fatigue level [181] or his specific pathology. Furthermore, in more distant future, such human-centred approaches could be used to create scenarios in which the operator's load would not be adapted just to its current abilities, but modulated in a way to increase his abilities over time or potentially even enable the rehabilitation in the factory context.

However, even though the ICHM algorithm, and the potential collaborative robot control strategies using it, are independent of any musculoskeletal model, their accuracy relies entirely on the model's real-time representation of the operator. Therefore, for practical implementations, the key challenges are two-fold: evaluating human posture in the real-time, precise enough and in a minimally intrusive way, as well as performing the individual scaling and calibration of the used musculoskeletal models for a given operator [180].

In their recent work, Laisné *et al.* [182] have proposed a new promising method for calibrating the musculoskeletal models of human subjects, based in part on the works conducted in this thesis, particularly on the ICHM algorithm. The method consists in measuring the human subject's force capacity polytope in the laboratory setup and finding the optimal parameters of the musculoskeletal models to minimise the error between the recorded and simulated polytope. The method is based on the optimisation using Genetic algorithms and is applicable to relatively detailed musculoskeletal models, such as the model with 50 muscles and 7 joints developed by Holzbaur *et al.* [175].

Such techniques have a great potential to enable using detailed and well adapted models of the human operators and, in combination with the efficient tools for their analysis, enable high degree of personalised assistance to the operators, further improving their well-being and safety.

4.5 Conclusion

This chapter showcases the potential of employing polytope representations of robots' and humans' physical abilities to develop collaborative robot control strategies capable of adaptation to their changing abilities online. The chapter demonstrates that real-time calculation of these polytopes provides promising tools for creating flexible robot control strategies for different physical collaborating scenarios. Allowing the robot to adapt not just to the requirements of the task and its own abilities, but also to the changing abilities of the operators and their safety.

The chapter attempts to demonstrate three concepts, stating the potential impact of using real-time information about robot's and human's physical abilities for creating the adaptive robot control in collaborative scenarios. Concept 1 (C1) states that this real-time information can be exploited to better use each of the actors individual abilities. Concept 2 (C2) states that such real-time information can be used to better distribute the task load and allow them to collaboratively perform the task they would have been unable to do on their own. Finally, Concept 3 (C3) states that the human's real-time capacity information can be used to improve his safety and well-being while executing the task.

To support these concepts, the chapter brings two example collaborative scenarios, based on a common industrial task of collaborative carrying of a heavy object. The scenarios studied are: dual robotic arm collaborative carrying and human-robot collaborative carrying.

In the dual robot arm experiment two Franka Emika Panda robots carry an object with a mass of 12 kg through the common workspace, even though their rated payload, given by the manufacturer, is 3kg. Robots' carrying capacity is calculated in real-time using the algorithm VEPOLI^2, proposed in Section 3.4. This real-time information is then used to create the collaborative robot control strategy that adapts to their changing capacities in real-time.

The experiment shows that by calculating robots' carrying capacity in real-time, each one of the robots is able to surpass their rated abilities without compromising their safety, demonstrating Concept 1 (C1) . Furthermore, by using the real-time information about the carrying capacity of both robots, the proposed control distributes the weight of the object to the robots with respect to their changing physical abilities. As a result, the experiment shows that the two robots are able to compensate for the 12kg weight during the whole time of the experiment, even though neither of the robots would have been able to carry half of the object's weight on their own, demonstrating Concept 2 (C2) . This experiment has been published as a part of the scientific article **A. Skuric** *et al.* [A1].

In the second experiment, human operator and a Franka Emika Panda robot jointly carry 7kg object. Human's and robot's carrying capacity are calculated in real-time and used to create the collaborative robot control strategy adapting to their real-time changes. In order to determine the human's carrying capacity, a relatively detailed musculoskeletal model of the human right arm is used, with 50 muscles and 7 joints, developed by Holzbaur *et al.* [175]. Then, the newly developed algorithm ICHM, described in Section 3.5, is used to do the real-time calculations.

The experiment reveals that human carrying capacity exhibits greater variability compared to that of the robot. Interestingly, in several instances, the human demonstrated a higher carrying capacity than the robot. By having real-time information about the carrying capacities of both the human and the robot, the proposed control strategy effectively distributes the weight between them, while making sure that their both physical abilities are not exceeded. Even though neither of them would have been able to carry the object on their own, the proposed control strategy enables them to successfully carry out the task when collaborating physically, once again demonstrating Concept 2 (C2) .

Moreover, in addition to to making sure that human's physical ability is not surpassed during the task executions, to further demonstrate Concept 3 (C3) , an additional task was incorporated into the collaborative robot control strategy aiming to improve the operator's safety and his engagement in the task. This task consisted in maintaining a constant relative load for the operator throughout the experiment, corresponding to 30% of its carrying capacity. This concept was inspired by the Assist-As-Needed (AAN) strategy as described in Carmichael and Liu [68], commonly used in rehabilitation and assistance robotics. The human-robot collaborative carrying experiment has been published as a part of the scientific article **A. Skuric** *et al.* [A3].

In conclusion, this chapter advocates for creating robot control strategies based on the real-time and accurate information about the changing nature of task related physical abilities of robots and humans, based on polytopes. The chapter shows that such strategies enable employing their individual abilities fully without compromising their safety. Furthermore, when collaborating physically, such strategies enable distributing their task execution load with regard to their changing abilities, allowing them to execute tasks they would not have been able to do individually. Finally, the chapter shows that having the real-time and accurate information about human's physical abilities has a great potential to further improve operator's safety and well-being at the workplace.

Following two chapters explore the potential use-cases of the real-time physical ability polytope evaluation in different human-robot collaboration scenarios. Chapter 5, explores the possibilities of using polytopes for the real-time visual feedback to the operators. Chapter 6 proposes a new Cartesian Space (CS) trajectory planning strategy exploiting the polytope algebra. Finally, Chapter 7 presents a publicly available open-source software Python package `pycapacity`, providing tools for the efficient evaluation of polytope and ellipsoid based physical abilities of humans and robots.

Chapter 5

Real-time polytope visualisation for human-robot collaboration

As described in previous chapters, polytopes are one of the most accurate characterisations of different physical abilities of humans and robots. Commonly, they are valuable tools for numerous offline analysis applications, such as evaluating the performance of different robot designs with respect to the task requirements [31, 110, 183, 184]. In addition to using polytopes as quantitative indicators, these applications often employ different visual representations of polytopes to provide qualitative insights and additional visual information to operators. With the recent increase in the computing power and the development of more capable algorithms, such as ICHM and VEPOLI^2, the polytope characterisations of physical abilities make their way in online, interactive, applications as well. Leveraging these efficient algorithms, Chapter 4 shows how the real-time calculation of physical ability polytopes can be used to create advanced robot control strategies and enhance the physical collaboration of humans and robots.

However, efficient human-robot collaboration, in addition to flexible and human-centred robot control strategies, requires a high degree of operator's situation awareness [185] both in terms of robot's behaviour and its physical abilities in real-time. As described by Camblor *et al.* [186], human's situation awareness in an important factor in human-robot collaboration, having a direct influence on the operator's safety and the collaboration performance. Low situational awareness is often associated with higher risk of accidents [187], while better situational awareness leads to more efficient collaborations [186].

This chapter explores the idea of using the interactive visualisation of robot's physical ability polytopes to the operator, with the aim to improve the operator's understanding about robot's task, about robot's current state and its current physical abilities. As discussed in Chapter 2, polytopes can be efficiently transformed into triangulated meshes and visualised with most of the standard visualisation tools. One particularly interesting set of tools, providing an interactive and immersive approach to visualisation, are the Virtual Reality (VR) and Mixed Reality (MR) or Augmented Reality (AR) tools. AR and MR tools are particularly well suited for human-robot physical interaction as they allow the operator to act on the physical word directly, while augmenting his vision of the physical world with virtual features (information) [188]. VR tools, on the other hand, require the actions of the operator to be in the virtual world as well, which is more suitable to the robot teleportation scenarios [189].

Using these new set of tools, several works have been recently proposed in the literature visualising the physical ability polytopes to the operator in the human-robot collaboration context. Recently, Weistroffer *et al.* [80] proposed an approach, based on MR, that improves safety in collaborative workstations by visualising several different robot's physical ability characterisations to the operator, force polytopes being one of them. Zolotas *et al.* [79], on the other hand, focused more precisely on the constrained velocity polytopes, proposed by Long and Padır [34].

In their study the polytopes are visualised to the human subjects while teleoperating the robot using the VR tools. The authors study several polytope visualisation strategies. The study results show that the pure triangulated mesh visualisation, in the case of their task, is not the most intuitive to the operators.

One of the possible reasons why subjects found polytopes not intuitive might be the abundance of information within the polytope. Polytopes can have many hundreds of vertices and faces, which can increase the operator's cognitive load required to isolate the important information. Additionally, determining the appropriate (intuitive) physical ability to be visualised to the operator is a challenging scientific problem, as this highly depends on the nature of the task and potentially on the preferences of the operator as well. This is especially true since the physical ability polytopes often represent different abstract physical quantities, such as forces or accelerations. In order to be visualised they are scaled and transformed to the Cartesian Space (CS) (space of CS positions), which possibly adds to the operator's confusion.

To answer some of these challenges, this chapter brings a new polytope formulation able to incorporate multiple robot's actuation limits within one single polytope representation, while remaining easy to interpret. This new polytope formulation represents the CS reachable by the robot's within a given time horizon and is particularly aimed to be used as a visualisation tool to the operators. Furthermore, the chapter brings a preliminary work on the development of the testing setup for evaluating the efficiency of different polytopes and their visualisation modalities for sharing information with the operators.

The chapter is structured as follows. Section 5.1 introduces the convex polytope based approximation approach of the robot's reachable space within the horizon, while its approximation accuracy and execution time are studied in Section 5.1.3. Section 5.2 brings a preliminary work on non-convex approximation of this reachable space in order to provide a more accurate information to the operator. Finally, Section 5.3 brings the preliminary work on visualising the polytopes to the operator using the AR tools.

5.1 Approximating robot's reachable space with convex polytopes

This section introduces a new polytope characterisation of robot's physical abilities aimed particularly for the informative and easy to interpret visualisation to the operators. In order provide the informative insight to the operator, the proposed polytope formulation unifies the influences of multiple robot's actuator limits, such as Joint Space (JS) positions, velocities and torques, as well as the influence of the robot's kinematics and dynamics. At the same time, its geometrical interpretation remains intuitive, representing the prediction of the Cartesian Space (CS) positions reachable by the robot.

The set of all the CS positions (ex. end-effector CS positions) the robot can reach, given its geometry and the limitations of its actuator's positions, is a well studied problem in the literature and is often called robot's reachable workspace [13–15]. Different reachable workspace characterisations are common visual tool in robotics, often specified in the manufacturer's data-sheets [190, 191] using images such as on Figure 5.1. However, the geometry of reachable workspaces is often very complex, which can be challenging to characterise mathematically, as well as to exploit when it comes to practical applications. Furthermore, the methods used to calculate the reachable workspaces are often based on Interval analysis [192] or different extensive sampling strategies [14, 20]. Such techniques are often developed for robot design and analysis applications and their computational efficiency does not allow for interactive applications. Finally, robot's reachable workspace is evaluated in advance for the entirety of

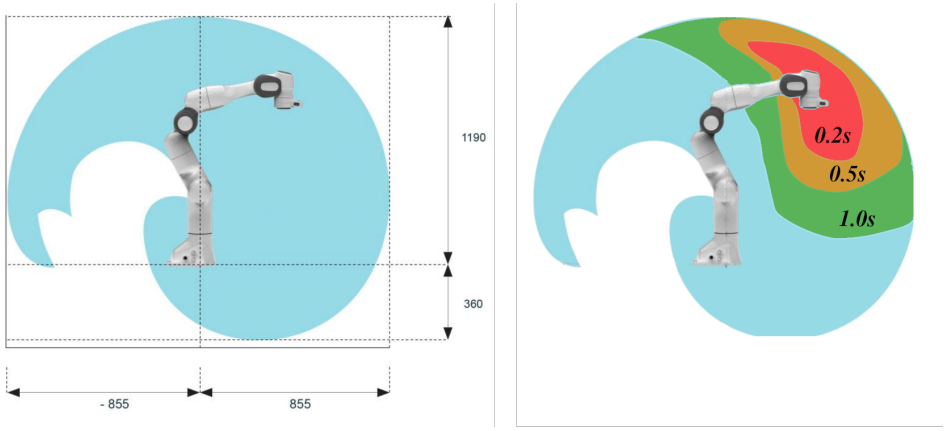

FIGURE 5.1: Figure on the left shows a section of the reachable workspace for a Franka Emika Panda robot given in the manufacturer's datasheet [191], as the area shown in blue. Figure on the right conceptually illustrates, in different colours, the area of this workspace reachable within the horizon times of 0.2s, 0.5s and 1s.

robot's states, therefore it lacks the instantaneous information about the robot's current state and its current abilities.

Instead of characterising the robot's entire workspace, the tools from Reachability Analysis [193, 194] allow for a more interactive characterisation of the robot's reachable space: robot's reachable space within a time horizon. Such characterisations represent the space of robot's reachable positions (ex. end-effector CS positions) within a certain time horizon, given the robot's current state and different assumptions on its physical abilities. Figure 5.1 provides an illustration of the complete reachable workspace of a Franka Emika Panda robot, as well as the reachable space within a several different horizon times.

As robotic systems are highly nonlinear and featuring complex kinematics and dynamics, the exact characterisation of their reachable spaces is very complex to characterise and calculate. Therefore, approximation techniques are necessary in order to yield practical solutions [195]. Pereira and Althoff [196] have developed an set based approach, later improved by Schepp *et al.* [197], to approximate the human arm reachable space, modelled as a serial robotic manipulator. Their approach approximates the reachable space using a set of spheres and cylinders. However, in many cases the approximation of the reachable space using these shapes is impractical sue to their nonlinear nature.

In this section, an approximation approach of the robot's reachable space using convex polytopes is proposed, leveraging several of their key characteristics. Since the reachable space is often low dimensional ($\leq$3D) these polytopes can be easily visualised, having the structure of a triangulated mesh. Additionally, convex polytopes can be represented as a set of linear inequalities $A\boldsymbol{x} \leq \boldsymbol{b}$, which may be directly used with different optimisation techniques. Additionally, this set of linear inequalities can be intuitively extended with different environmental and user defined constraints. Furthermore, operations over polytopes such as Minkowski sum and intersection are well defined and efficient to calculate, enabling for the intuitive extension of the proposed method to account for robot's link geometry. Finally, the efficiency of polytope enumeration techniques for low dimensional polytopes has a potential to make this metric real-time capable.

As discussed in Chapter 2, convex polytopes are widely used to characterise different task space physical abilities in robotics (ex. force, acceleration, velocity capacity). However, to the best of our knowledge they have not yet been used to characterise the reachable space of a robotic

manipulator. The proposed approach is partially inspired by the works of Long and Padir [33] on the constrained manipulability (velocity) polytopes.

The section has the following structure. In Section 5.1.1 the formal definition of the approach is introduced, as well as its extension with the environmental constraints and the link geometry. Section 5.1.3 brings the analysis of the accuracy of the convex polytope approximation, as well as the execution time analysis. In Section 5.1.3.4, the results of the accuracy and execution time analysis of the approach are given and Section 5.1.4 brings the discussion on main limitations.

5.1.1 Reachable space polytope definition

The dynamics equation of a serial robot in Joint Space (JS) can be expressed as

$$M(\boldsymbol{q})\ddot{\boldsymbol{q}} + \underbrace{C(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q})}_{\boldsymbol{\tau}_d} = \boldsymbol{\tau} \tag{5.1}$$

where $M \in \mathbb{R}^{n\times n}$ is the mass matrix, $C \in \mathbb{R}^{n\times n}$ is a matrix grouping Coriolis and centrifugal effects and $\boldsymbol{g} \in \mathbb{R}^n$ is the gravity vector. $\boldsymbol{\tau} \in \mathbb{R}^n$ is the applied joint torque vector and $\boldsymbol{\tau}_d \in \mathbb{R}^n$ is the equivalent joint torque vector due to the nonlinear effects of the robot's movement and gravity. Vectors $\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}} \in \mathbb{R}^n$ are robot's JS position, velocity and acceleration respectively.

For a robot with n actuated degrees of freedom (DOF), the robot's joint torque $\boldsymbol{\tau}$, velocity $\dot{\boldsymbol{q}}$ and joint angles $\boldsymbol{q}$ are n-dimensional vectors, limited by the robot's hardware[1]

$$\boldsymbol{\tau} \in [\boldsymbol{\tau}_{min}, \boldsymbol{\tau}_{max}], \quad \dot{\boldsymbol{q}} \in [\dot{\boldsymbol{q}}_{min}, \dot{\boldsymbol{q}}_{max}], \quad \boldsymbol{q} \in [\boldsymbol{q}_{min}, \boldsymbol{q}_{max}] \tag{5.2}$$

For a given moment in time t_o and corresponding robot's state $\{\boldsymbol{q}_o, \dot{\boldsymbol{q}}_o\}$, the affine relationship between joint torques $\boldsymbol{\tau}$ and accelerations $\ddot{\boldsymbol{q}}_o$ can be expressed as

$$\ddot{\boldsymbol{q}}_o = M^{-1}(\boldsymbol{q}_o)(\boldsymbol{\tau} - \boldsymbol{\tau}_d(\boldsymbol{q}_o, \dot{\boldsymbol{q}}_o)) = M_o^{-1}(\boldsymbol{\tau} - \boldsymbol{\tau}_{d,o}) \tag{5.3}$$

Considering a constant joint acceleration $\ddot{\boldsymbol{q}}_o$ during a given horizon length t_h, an approximation of the robot's joint velocity $\dot{\boldsymbol{q}}_h$ and position $\boldsymbol{q}_h$, at the end of horizon, can be calculated using numerical integration (forward Euler method)

$$\dot{\boldsymbol{q}}_h = M_o^{-1} t_h(\boldsymbol{\tau} - \boldsymbol{\tau}_{d,o}) + \dot{\boldsymbol{q}}_o, \qquad \boldsymbol{q}_h = M_o^{-1}\frac{t_h^2}{2}(\boldsymbol{\tau} - \boldsymbol{\tau}_{d,o}) + \dot{\boldsymbol{q}}_o t_h + \boldsymbol{q}_o \tag{5.4}$$

This linear numerical integration (5.4) considers the robot's dynamics (5.1), and the applied joint torque $\boldsymbol{\tau}$, fixed during the horizon time t_h. Such assumption is reasonable only for short horizon times t_h which in term represents a limitation of the proposed method.

The relationship between the m-dimensional task space velocity $\dot{\boldsymbol{x}}$ and acceleration $\ddot{\boldsymbol{x}}$ of a certain frame on the robot (for example end-effector frame) and the n-dimensional JS equivalents is defined through the corresponding Jacobian matrix $J(\boldsymbol{q}) \in \mathbb{R}^{m\times n}$ and its time derivative $\dot{J}(\boldsymbol{q}) \in \mathbb{R}^{m\times n}$

$$\dot{\boldsymbol{x}} = J\dot{\boldsymbol{q}}, \qquad \ddot{\boldsymbol{x}} = J\ddot{\boldsymbol{q}} + \dot{J}\dot{\boldsymbol{q}} \tag{5.5}$$

Finally, given the horizon of interest t_h, the predicted Cartesian Space (CS) position $\boldsymbol{x}_h$ can then be expressed as

[1] There is a coupling between the velocity and torque limits of an actuator related to its power. Yet, it is common practice to consider a subset of these limits such as torque and velocity can be chosen independantly one from the other.

$$\begin{aligned} \boldsymbol{x}_h &= \ddot{\boldsymbol{x}}_o \frac{t_h^2}{2} && && + \dot{\boldsymbol{x}}_o t_h && + \boldsymbol{x}_o \\ &= J_o M_o^{-1} \frac{t_h^2}{2} \boldsymbol{\tau} && \underbrace{- J_o M_o^{-1} \frac{t_h^2}{2} \boldsymbol{\tau}_{d,o}}_{\Delta \boldsymbol{x}_{o,dyn}} && \underbrace{+ \dot{J}_o \dot{\boldsymbol{q}}_o \frac{t_h^2}{2} + \dot{\boldsymbol{x}}_o t_h}_{\Delta \boldsymbol{x}_{o,vel}} && + \boldsymbol{x}_o \end{aligned} \tag{5.6}$$

$\hat{\boldsymbol{x}}_h$

Where $\hat{\boldsymbol{x}}_h = \Delta \boldsymbol{x}_{o,dyn} + \Delta \boldsymbol{x}_{o,vel} + \boldsymbol{x}_o$ is a predicted position vector calculated by integrating the influences of the robot's joint movement $\{\dot{\boldsymbol{q}}_o, \boldsymbol{\tau}_{d,o}\}$ over the time horizon t_h. The first term describes the influence of the applied joint torques $\boldsymbol{\tau}$ on CS position reached, considered constant during the horizon t_h.

The convex polytope of reachable CS positions $\mathcal{P}_x$ can then be defined as a set of all the possible CS positions $\boldsymbol{x}_h$ at the end of the horizon t_h, achieved by any combination of joint torques $\boldsymbol{\tau}$ within robot's physical limitations (5.2), given the robot's current state $\{\dot{\boldsymbol{q}}_o, \boldsymbol{q}_o\}$ and $\boldsymbol{\tau}_{d,o}$.

$$\begin{aligned} \mathcal{P}_x = \{ \boldsymbol{x}_h \in \mathbb{R}^m \quad | \quad & \boldsymbol{x}_h = J_o M_o^{-1} \frac{t_h^2}{2} \boldsymbol{\tau} + \hat{\boldsymbol{x}}_h, \\ & \boldsymbol{\tau} \in [\boldsymbol{\tau}_{min}, \boldsymbol{\tau}_{max}], \\ & M_o^{-1} t_h (\boldsymbol{\tau} - \boldsymbol{\tau}_{d,o}) + \dot{\boldsymbol{q}}_o \in [\dot{\boldsymbol{q}}_{min}, \dot{\boldsymbol{q}}_{max}], \\ & M_o^{-1} \frac{t_h^2}{2} (\boldsymbol{\tau} - \boldsymbol{\tau}_{d,o}) + \dot{\boldsymbol{q}}_o t_h + \boldsymbol{q}_o \in [\boldsymbol{q}_{min}, \boldsymbol{q}_{max}] \} \end{aligned} \tag{5.7}$$

Polytope $\mathcal{P}_x$ formulation (5.7) can also be seen as projection of the joint torque polytope $\mathcal{P}_\tau$ to the lower m dimensional task space

$$\mathcal{P}_x = \{ \boldsymbol{x}_h \in \mathbb{R}^m \mid \boldsymbol{x}_h = J_o M_o^{-1} \frac{t_h^2}{2} \boldsymbol{\tau} + \hat{\boldsymbol{x}}_h, \quad \boldsymbol{\tau} \in \mathcal{P}_\tau \} \tag{5.8}$$

where the joint torque polytope $\mathcal{P}_\tau$ integrates all the joint actuator limits (5.2) of the robot

$$\begin{aligned} \mathcal{P}_\tau = \{ \boldsymbol{\tau} \in \mathbb{R}^n \quad | \quad & \boldsymbol{\tau} \in [\boldsymbol{\tau}_{min}, \boldsymbol{\tau}_{max}], \\ & M_o^{-1} t_h (\boldsymbol{\tau} - \boldsymbol{\tau}_{d,o}) + \dot{\boldsymbol{q}}_o \in [\dot{\boldsymbol{q}}_{min}, \dot{\boldsymbol{q}}_{max}], \\ & M_o^{-1} \frac{t_h^2}{2} (\boldsymbol{\tau} - \boldsymbol{\tau}_{d,o}) + \dot{\boldsymbol{q}}_o t_h + \boldsymbol{q}_o \in [\boldsymbol{q}_{min}, \boldsymbol{q}_{max}] \} \end{aligned} \tag{5.9}$$

The visualisation potential of the proposed polytope formulation $\mathcal{P}_x$ is showcased on Figure 5.2, Figure 5.3, and Figure 5.4, as well as in the publicly available video[2]. The influences of the robot's actuator constraints, robot movement $(\dot{\boldsymbol{q}}_o, \boldsymbol{\tau}_{d,o})$ and different horizon lengths t_h, is captured through the changing shape and size of the reachable space polytope $\mathcal{P}_x$.

Video

The following sections extend this polytope formulation with the carried payload, environmental constraints and the robot's link geometry. Section 5.1.2 then describes an efficient approach to finding the $\mathcal{H}$ and $\mathcal{V}$-representation of the reachable space polytope $\mathcal{P}_x$.

[2]Video: https://youtu.be/JwZgrUp095Y

[3]More information about the Panda robot at https://frankaemika.github.io/docs/

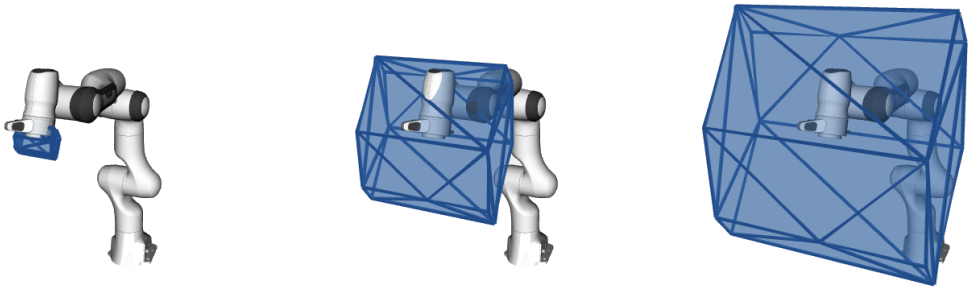

FIGURE 5.2: Three images show the comparison of the size of the reachable space polytope $\mathcal{P}_x$ of a Franka Emika Panda[3] robot's end-effector for 3 horizons (left to right) t_h = 0.05, 0.15 and 0.25s, for the same configuration.

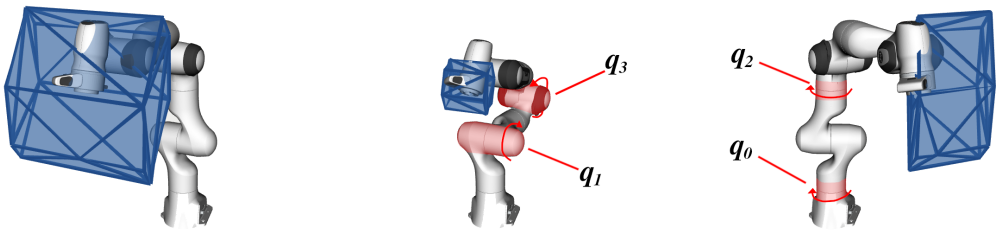

FIGURE 5.3: Reachable space polytope $\mathcal{P}_x$ for a Franka Emika Panda robot's end-effector for 3 configurations (t_h=0.15s). On the left, the robot is in its initial position $\boldsymbol{q} = [0, 0, 0, -\pi/2, 0, 3\pi/5, 0]$, close to the center of the joint ranges. In the middle figure the robot has joints q_1 and q_3 are at their limits, $\boldsymbol{q} = [0, -1.59, 0, -2.9, 0, 3\pi/5, 0, 0]$. On the right, robot's joints q_0 and q_2 are at their limits $\boldsymbol{q} = [-2.72, 0, -2.72, -\pi/2, 0, 13\pi/5, 0, 0]$, preventing the robot to rotate around the z axis in one direction.

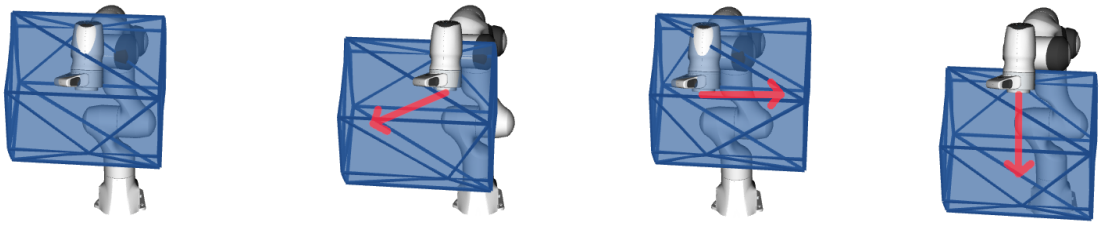

FIGURE 5.4: Reachable space polytope $\mathcal{P}_x$ for a Franka Emika Panda robot's end-effector for 4 different end-effector velocities $\dot{\boldsymbol{x}}_o = J\dot{\boldsymbol{q}}_o$ (from left to right) $\dot{\boldsymbol{x}}_o = [0, 0, 0]$, $[0, -1, -0.5]$, $[0, 1, 0]$ and $[0, 0, -1] m/s$.

5.1.1.1 Influence of the carried object

Consider a robot that carries a payload, an object with a mass m_{object} and inertia i_{object}, attached to its end-effector. This object has an influence on the robot's dynamics, modifying the mass matrix M, coriolis matrix C and the gravity vector $\boldsymbol{g}$ [198]. The augmented dynamical model of the robot has reduced acceleration capabilities due to the added effort necessary for the object's movements.

Figure 5.5 shows the influence of different object masses m_{object}, on the resulting reachable space polytope $\mathcal{P}_x$ for the Franka Emika Panda robot.

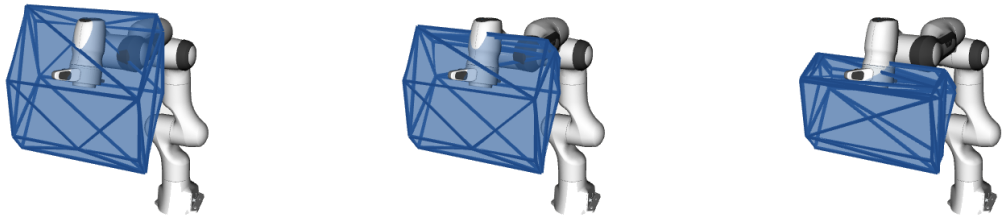

FIGURE 5.5: Reachable space polytope $\mathcal{P}_x$ for a Franka Emika Panda robot for three different carried object masses (left to right) m_{object} = 0, 2 and 5kg, horizon time used t_h=0.15s.

5.1.1.2 Integration of the environment

If the robot's environment can be defined as a set of convex inequalities

$$A_e \boldsymbol{x} \leq b_e \tag{5.10}$$

these constraints can be directly transformed to the constraints of the joint torque $\boldsymbol{\tau}$ using the equation (5.6)

$$A_e \boldsymbol{x}_h = A_e J_o M_o^{-1} \frac{t_h^2}{2} \boldsymbol{\tau} + A_e \hat{\boldsymbol{x}}_h \leq b_e \tag{5.11}$$

and then included in the $\mathcal{P}_x$ calculation (5.7)

$$\begin{aligned} \mathcal{P}_x = \{ \boldsymbol{x}_h \in \mathbb{R}^m \quad | \quad & \boldsymbol{x}_h = J_o M_o^{-1} \frac{t_h^2}{2} \boldsymbol{\tau} + \boldsymbol{x}_h^*, \\ & \boldsymbol{\tau} \in [\boldsymbol{\tau}_{min}, \boldsymbol{\tau}_{max}], \\ & M_o^{-1} t_h (\boldsymbol{\tau} - \boldsymbol{\tau}_{d,o}) + \dot{\boldsymbol{q}}_o \in [\dot{\boldsymbol{q}}_{min}, \dot{\boldsymbol{q}}_{max}], \\ & M_o^{-1} \frac{t_h^2}{2} (\boldsymbol{\tau} - \boldsymbol{\tau}_{d,o}) + \dot{\boldsymbol{q}}_o t_h + \boldsymbol{q}_o \in [\boldsymbol{q}_{min}, \boldsymbol{q}_{max}], \\ & A_e J_o M_o^{-1} \frac{t_h^2}{2} \boldsymbol{\tau} + A_e \hat{\boldsymbol{x}}_h \leq b_e \quad \} \end{aligned} \tag{5.12}$$

Figure 5.6 showcases the extension of the reachable space polytope $\mathcal{P}_x$ with the environmental constraints, influencing directly its shape. The same approach can be used to visualise the robot's task space constraints to the operator, enforced by the robot control. For example in the case of robot teleoperation, the polytope $\mathcal{P}_x$ can be used to visualise to the operator the space reachable by the robot while including virtual walls [199].

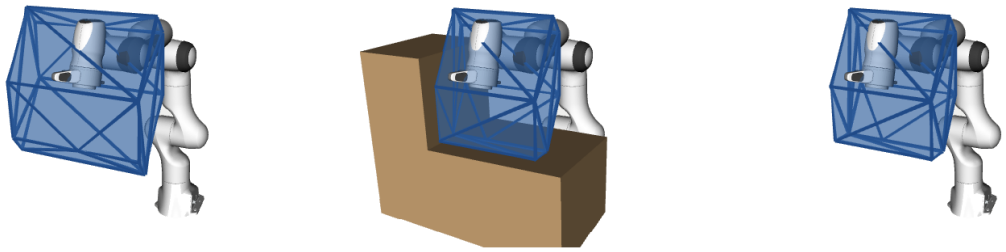

FIGURE 5.6: Resulting reachable space polytope $\mathcal{P}_x$ for a Franka Emika Panda robot's end-effector when integrating the environmental constraints, horizon time used is t_h=0.15s. Environment is defined as $z \geq 0.5$m and $y \geq -0.2$m.

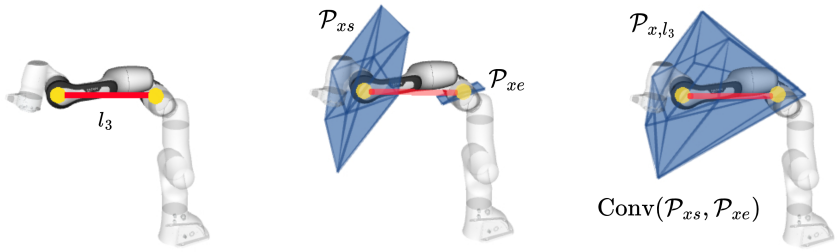

FIGURE 5.7: The figure shows the construction of the reachable space polytope of the Panda robot's link 3, $\mathcal{P}_{x,l_3}$. The first image shows the robot's link 3 modelled as a line segment l_3 (in red). The polytopes $\mathcal{P}_{xs}$ and $\mathcal{P}_{xe}$ calculated in its vertices (yellow) are shown in the middle. $\mathcal{P}_{x,l_3}$, calculated as their Convex-Hull, is shown on the right. Robot is in its initial configuration, and the horizon time used is 150ms.

5.1.1.3 Integration of robot's link geometry

Leveraging the efficient tools from polytope algebra, the proposed reachable space approximation can be extended to the calculation of the reachable space of robot's links as well.

For example, if a robot link l_i is modelled as a line segment, the polytope $\mathcal{P}_x$ can be calculated at the start $\mathcal{P}_{xs}$ and the end $\mathcal{P}_{xe}$ of the line l_i. Then, the reachable space of this idealised robot link l_i can then be calculated as the Convex-Hull (Conv $(\cdot)$) of the polytopes $\mathcal{P}_{xs}$ and $\mathcal{P}_{xe}$

$$\mathcal{P}_{x,l_i} = \text{Conv}\left(\mathcal{P}_{xs},\ \mathcal{P}_{xe}\right) \tag{5.13}$$

Figure 5.7 shows the construction of the reachable space polytope of the Panda robot's link 3, modelled as a line segment.

If, instead of a straight line, the space captured by a robot's link l_i is expressed as a convex set of constraints or a polytope ($\mathcal{H}$-representation)

$$\mathcal{L}_i = \left\{ \boldsymbol{x}_{l_i} \mid A_{l_i} \boldsymbol{x}_{l_i} \leq b_{l_i} \right\} \tag{5.14}$$

the reachable space polytope $\mathcal{P}_{x,l_i}$ can be extended to account for this link geometry by first finding the vertices ($\mathcal{V}$-representation) of the polytope $\mathcal{L}_i$, for example using one of the standard polytope transformation methods described in Section 3.3.1. Then the reachable space polytope $\mathcal{P}_x$ can be evaluated for each one of the N_v vertices, while the full link polytope $\mathcal{P}_{x,\mathcal{L}_i}$ can be found by computing their Convex-Hull

$$\mathcal{P}_{x,\mathcal{L}_i} = \text{Conv}\left(\mathcal{P}_{x1},\ \ldots\ , \mathcal{P}_{xN_v}\right) \tag{5.15}$$

By evaluating the polytope $\mathcal{P}_{x,\mathcal{L}_i}$ for each robot's link one can obtain an efficient approximation of the reachable space of the entire robot. One example of constructing such reachable space is shown on Figure 5.8.

The link polytopes $\mathcal{P}_{x,\mathcal{L}_i}$ can be further used to calculate the convex envelope $\mathcal{P}_{env}$ of the robot's reachable space as well, by calculating the their Convex-Hull.

$$\mathcal{P}_{env} = \text{Conv}\left(\mathcal{P}_{x,\mathcal{L}_1},\ \mathcal{P}_{x,\mathcal{L}_2},\ \ldots\right) \tag{5.16}$$

Video

An interactive visualisation of the reachable space approximation for the robot's links as well as its convex envelope can be found in the publicly available video[4].

[4]Video: https://youtu.be/1J2UrMC2uP0

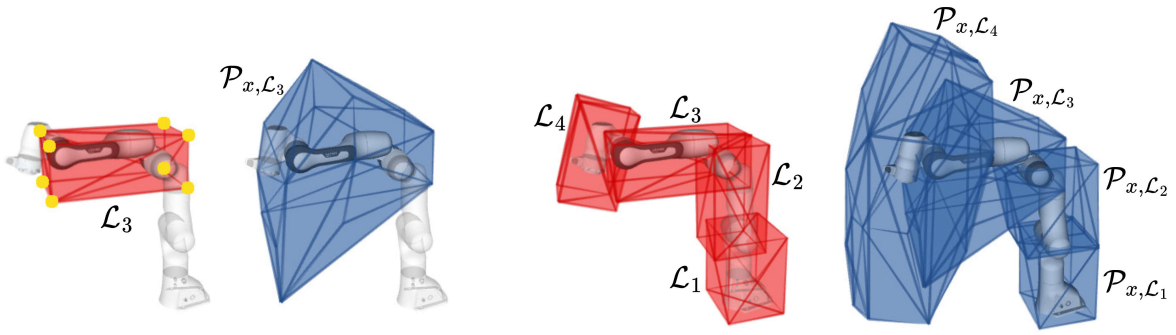

FIGURE 5.8: First two images show the construction of the reachable space polytope of the panda robot's link 3, $\mathcal{P}_{x,\mathcal{L}_3}$. The left figure shows the link 3, modelled as a box $\mathcal{L}_3$ (in red). Followed by the visualisation of its reachable space polytope $\mathcal{P}_{x,\mathcal{L}_3}$, calculated as a Convex-Hull of the polytopes of each one of its 8 vertices (in yellow). Last two images show 4 enveloping spaces $\mathcal{L}_i$ for each one of the robot's links, and their reachable space polytopes $\mathcal{P}_{x,\mathcal{L}_i}$. Robot is in its initial configuration, and the horizon time used is 150ms.

5.1.2 Finding $\mathcal{H}$ and $\mathcal{V}$-representation of the reachable space polytope

In order to be used with standard applications, such as to be visualised to the operator, this polytope has to be transformed to the set of the vertices ($\mathcal{V}$-representation) or the set of half-planes ($\mathcal{H}$-representation) representing its faces.

The reachable space polytope $\mathcal{P}_x$ belongs to the family of problems

$$\mathcal{P}_x = \{\boldsymbol{x} \in \mathbb{R}^m \quad | \quad \boldsymbol{x} = B\boldsymbol{\tau} + \boldsymbol{b}_x, \quad H_\tau \boldsymbol{\tau} \leq \boldsymbol{d}_\tau \quad \}, \quad \boldsymbol{\tau} \in \mathbb{R}^n, \quad n \geq m \tag{5.17}$$

where the inequality constraint $H_\tau \boldsymbol{\tau} \leq \boldsymbol{d}_\tau$ encapsulates all the robot constraints (5.2) as well as the environmental constraints (5.11), and the matrix $J_o M_o^{-1} \frac{t_h^2}{2}$ becomes the projection matrix B. Vector $\boldsymbol{\tau}$ corresponds to the joint torque vector, $\boldsymbol{x}$ corresponds to the positions at the end of the horizon $\boldsymbol{x}_h$ and the bias vector $\boldsymbol{b}_x$ corresponds to $\hat{\boldsymbol{x}}_h$.

This polytope has the projection formulation with polytope input set, as described in Section 3.2.1. The set of constraints $H_\tau \boldsymbol{\tau} \leq \boldsymbol{d}_\tau$ forms a $\mathcal{H}$-representation of a polytope $\mathcal{P}_\tau$ in the n-dimensional joint torque space

$$\mathcal{P}_\tau = \{\boldsymbol{\tau} \in \mathbb{R}^n \quad | \quad H_\tau \boldsymbol{\tau} \leq \boldsymbol{d}_\tau \quad \} \tag{5.18}$$

This polytope is then projected to the, usually much lower dimensional ($m \leq 3$), Cartesian Space (CS), forming the reachable space polytope $\mathcal{P}_x$ (5.17).

As described more in detail in Section 3.3.3, and condensed in Table 3.2, there are multiple ways to find the vertices and faces of this polytope. Table 3.2 shows that if finding the $\mathcal{H}$-representation of the polytope $\mathcal{P}_x$, the most straight-forward methods are the Fourier-Motzkin Elimination (FME)[141] or Equality-Set Projection (ESP)[144]. However, when searching for its $\mathcal{V}$-representation, no standard exact method can be directly applied. As described in Section 3.3.3.1, finding the $\mathcal{V}$-representation requires different multi-step approaches.

In order to avoid using the multi-step approaches and to find both the $\mathcal{H}$ and $\mathcal{V}$-representations of the polytope $\mathcal{P}_x$ at the same time, Iterative Convex-Hull Method (ICHM) is used. ICHM method, described in detail in Section 3.5, is defined for a generic family of sets

$$\mathcal{P} = \{ \boldsymbol{x} \in \mathbb{R}^m \mid A\boldsymbol{x} = B\boldsymbol{y} + \boldsymbol{b}, \quad H_y \boldsymbol{y} \leq \boldsymbol{d}_y \} \tag{5.19}$$

Using the ICHM algorithm for enumerating the reachable space polytope is rather straightforward, by setting its matrix A to identity $A = I_{m\times m}$, matrix B becomes the projection matrix P, the inequality constraint $H_y\boldsymbol{y} \leq \boldsymbol{d}_y$ then becomes the set of the constraints $H_\tau\boldsymbol{\tau} \leq \boldsymbol{d}_\tau$ and the bias $\boldsymbol{b}$ becomes the bias $\boldsymbol{b}_x$.

5.1.3 Analysing the approximation performance

The aim of the proposed polytope approximation of the robot's reachable space is to provide the real-time insight to the operators into the robot's state and its current physical abilities. Nevertheless, as this approach relies on approximating the actual reachable space, the accuracy of information provided to operators requires further validation. This section, therefore, brings a numerical analysis of the proposed approximation method's accuracy.

Defining metrics of interest for accuracy analysis is highly dependent of the applications these methods will be used on. If the application requires an under-approximation of the reachable set or an over-approximation, the metrics of interest will not be the same. In this section, the proposed polytope accuracy is analysed using three different quantitative metrics, each one providing an insight in different characteristics and limitations of the method.

Additionally, as this approximation method has a potential to be used for real-time applications, execution time analysis is performed as well.

In the extent of the proposed experiments, the considered reachable space is the reachable space of the robot's end effector.

5.1.3.1 Numerical analysis of the approximation accuracy

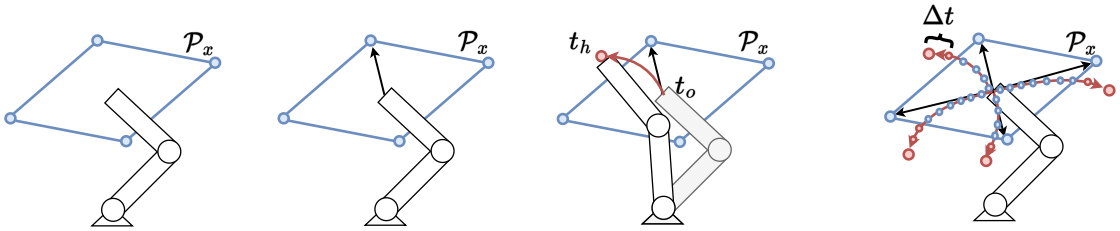

FIGURE 5.9: Images show the steps (left to right) of the nonlinear simulation procedure on a simplified 2 link planar robot. First the polytope $\mathcal{P}_x$ of the end-effector is determined. For each $\boldsymbol{x}_{v_i}$ vertex of the $\mathcal{P}_x$, the joint torque $\boldsymbol{\tau}_{v_i}$, generating this vertex, is applied to the robot simulation (figures in the middle). All the robot's end-effector positions $\boldsymbol{x}_k$ in all the simulation time steps Δt are saved for further analysis.

The main limitation of the proposed convex polytope $\mathcal{P}_x$ approach to the approximation of the robot's reachable space, and its main source of the approximation error, is the consideration of robot's dynamics and kinematics to be linear. Therefore, to analyse the approximation accuracy of the proposed approach, it is compared to the robot's nonlinear simulation.

Each vertex $\boldsymbol{x}_{v_i}$ of the reachable space polytope $\mathcal{P}_x$, calculated for the robot's state $\{\boldsymbol{q}_o, \dot{\boldsymbol{q}}_o\}$, is generated by a certain joint torque vector $\boldsymbol{\tau}_{v_i}$, considered constant during the time horizon t_h. Assuming the linear robot model, the joint torques $\boldsymbol{\tau}_{v_i}$, generating the vertices $\boldsymbol{x}_{v_i}$, correspond to the subset of vertices of the joint torque polytope $\mathcal{P}_\tau$ (5.9), that produce the largest possible robot's displacements in Cartesian Space (CS).

Therefore, to evaluate the accuracy of the proposed method, the difference in reached space produced by the constant liner model (polytope $\mathcal{P}_x$) and time varying nonlinear model of the robot is evaluated, for the same set of applied joint torques $\boldsymbol{\tau}_{v_i}$. Figure 5.9 illustrates the nonlinear simulation procedure.

A simple discrete nonlinear robot dynamics simulation, subject to the constraints (5.2), is employed. For each joint torque vector $\boldsymbol{\tau}_{v_i}$, a simulation is carried out using the equations (5.1-5.4)

$$\begin{aligned} \ddot{\boldsymbol{q}}_k &= M^{-1}(\boldsymbol{q}_k)\left(\boldsymbol{\tau}_{v_i} - C(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k)\dot{\boldsymbol{q}}_k + \boldsymbol{g}(\boldsymbol{q}_k)\right) \\ \dot{\boldsymbol{q}}_{k+1} &= \ddot{\boldsymbol{q}}_k \Delta t + \dot{\boldsymbol{q}}_k \\ \boldsymbol{q}_{k+1} &= \ddot{\boldsymbol{q}}_k \frac{\Delta t^2}{2} + \dot{\boldsymbol{q}}_k \Delta t + \boldsymbol{q}_k \end{aligned} \tag{5.20}$$

For each $\boldsymbol{\tau}_{v_i}$ considered, $N = t_h/\Delta t$ steps are taken within the horizon time t_h, where Δt is the simulation sampling time ($\Delta t = 5ms$ used in the experiments). Each simulation step (5.20) updates the robot's dynamics model (matrices M, C and vector $\boldsymbol{g}$) and calculates the resulting state $\{\boldsymbol{q}_{k+1}, \dot{\boldsymbol{q}}_{k+1}\}$, while considering the joint torque vector $\boldsymbol{\tau}_{v_i}$ constant during the complete roll-out (N steps).

The CS position in each step $\boldsymbol{x}_k$ is calculated by evaluating the robot's forward kinematics

$$\boldsymbol{x}_k = f_{fk}\left(\boldsymbol{q}_k\right) \tag{5.21}$$

Figure 5.9 illustrates the nonlinear simulation procedure. For each of the n_v vertices $\boldsymbol{x}_{v_i}$ of the polytope $\mathcal{P}_x$, the simulation (5.20) is performed, and all the CS positions $\boldsymbol{x}_k$ of the robot in each of the $N = t_h/\Delta t$ sample times are retained for the further analysis.

$$\mathcal{X} = \{\underbrace{\boldsymbol{x}_{0,0} \ \dots \ \boldsymbol{x}_{0,N}}_{\boldsymbol{\tau}_{v_1}}, \ \underbrace{\boldsymbol{x}_{1,0} \ \dots \ \boldsymbol{x}_{1,N}}_{\boldsymbol{\tau}_{v_1}}, \ \dots, \ \underbrace{\boldsymbol{x}_{n_v,0} \ \dots \ \boldsymbol{x}_{n_v,N}}_{\boldsymbol{\tau}_{v n_v}}\}$$

Based on the set of robot's CS positions reached in the numerical simulations $\mathcal{X}$, and the calculated polytope $\mathcal{P}_x$, this section proposes three different approximation accuracy metrics. Figure 5.10 shows the geometrical interpretation of the proposed metrics.

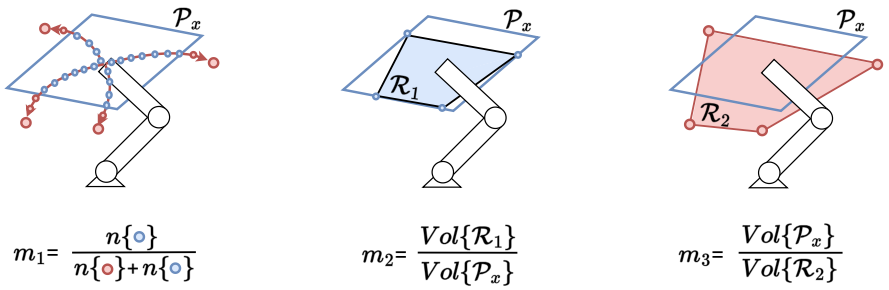

FIGURE 5.10: Depiction of the defined quantitative metrics for a simple 2 link planar robot. The notation $n\{\cdot\}$ represents the number of points.

Metric m_1 First metric, m_1, is defined as a ratio of number of the robot's CS positions $\boldsymbol{x}_k \in \mathcal{X}$, reached in the simulations, inside the polytope $\mathcal{P}_x$ with respect to the overall number of simulated positions.

$$m_1 = \frac{|\mathcal{X} \cap \mathcal{P}_x|}{|\mathcal{X}|} = \frac{|\mathcal{X} \cap \mathcal{P}_x|}{N \cdot n_v} \tag{5.22}$$

The notation $|\cdot|$ represents the number of elements of the set. The metric m_1 gives an insight on how well the polytope $\mathcal{P}_x$ encapsulates the real reachable space determined with simulations. This metric is designed to asses the confidence in the polytope $\mathcal{P}_x$. The closer this metric is to 1, the less chance that the robot can exit the space bounded by $\mathcal{P}_x$.

Metric m_2 The second metric, m_2, is defined as the ratio of the volumes of the polytope $\mathcal{P}_x$ and the Convex-Hull of the subset of the reached points $\boldsymbol{x}_k \in \mathcal{X}$ that are inside of $\mathcal{P}_x$.

$$m_2 = \frac{\mathcal{V}ol\,(\mathcal{R}_1)}{\mathcal{V}ol\,(\mathcal{P}_x)}, \qquad \mathcal{R}_1 = \mathrm{Conv}\,(\mathcal{X} \cap \mathcal{P}_x) \tag{5.23}$$

The metric m_2 quantifies the quality of the approximation providing an insight into the ratio of the polytope $\mathcal{P}_x$ that is actually attained by the robot in the simulations. A low score ($m_2 << 1$) of this metric indicates that the large part of the $\mathcal{P}_x$ is not actually reachable.

Metric m_3 The third metric, m_3, is defined as the ratio of the volumes of the polytope $\mathcal{P}_x$ and the Convex-Hull of all the points $\boldsymbol{x}_k \in \mathcal{X}$, reached by the robot in the simulations.

$$m_3 = \frac{\mathcal{V}ol\,(\mathcal{P}_x)}{\mathcal{V}ol\,(\mathcal{R}_2)}, \qquad \mathcal{R}_2 = \mathrm{Conv}\,(\mathcal{X}) \tag{5.24}$$

The metric m_3 gives insight about the ratio of the volumes of the polytope $\mathcal{P}_x$ and the simulated reachable space of the robot $\mathcal{R}_2$. This metric assesses the quality of the volume of the $\mathcal{P}_x$ as a metric. The further this metric from 1 the less confidence one has in the volume of the $\mathcal{P}_x$.

5.1.3.2 Benchmark comparison

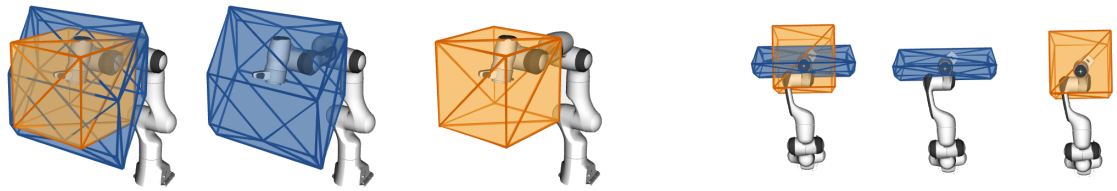

FIGURE 5.11: Comparison of the end-effector reachable space approximation using the convex polytope $\mathcal{P}_x$ (blue) and the cube $\mathcal{C}_x$ (orange) for the Panda robot with the horizon time t_h=200ms. They are both evaluated in robot's initial position and the figures show two different views.

To benchmark the accuracy of convex polytope approach $\mathcal{P}_x$, it is compared against the coarse approximation of the robot's end-effector reachable space using fixed CS velocity $\dot{\boldsymbol{x}}$ and acceleration $\ddot{\boldsymbol{x}}$ limits specified by robot manufacturer.

$$\ddot{\boldsymbol{x}} \in [\ \ddot{\boldsymbol{x}}_{min}, \ddot{\boldsymbol{x}}_{max}], \qquad \dot{\boldsymbol{x}} \in [\ \dot{\boldsymbol{x}}_{min}, \dot{\boldsymbol{x}}_{max}] \tag{5.25}$$

The CS limitations (5.25) assume that the robot's acceleration $\ddot{\boldsymbol{x}}$ and velocity $\dot{\boldsymbol{x}}$ capacity is constant. As described in Section 2.1, this not true since the robot's movement capacity depends highly on robot's state $\{\boldsymbol{q}, \dot{\boldsymbol{q}}\}$ [200]. However, the limits (5.25) are very convenient and widely used in many applications. A simple convex space $\mathcal{C}_x$ (cube in 3D) of reachable end-effector positions given the CS limitations (5.25) can be calculated as

$$\begin{aligned} \mathcal{C}_x = \{\ \boldsymbol{x}_h \in \mathbb{R}^m \mid \boldsymbol{x}_h = \ddot{\boldsymbol{x}}_o \frac{t_h^2}{2} + \dot{\boldsymbol{x}}_o t_h + \boldsymbol{x}_o, \\ \ddot{\boldsymbol{x}}_o \in [\ \ddot{\boldsymbol{x}}_{min}, \ddot{\boldsymbol{x}}_{max}], \\ \ddot{\boldsymbol{x}}_o t_h \in [\ \dot{\boldsymbol{x}}_{min}, \dot{\boldsymbol{x}}_{max}]\ \} \end{aligned} \tag{5.26}$$

Figure 5.11 shows the visual comparison of the two approaches: $\mathcal{P}_x$ and $\mathcal{C}_x$ calculated for one pose of the Franka Emika Panda robot.

5.1.3.3 Implementation details

The proposed analysis of the reachable space polytope accuracy is performed using the numerical simulation of the collaborative robotic manipulator Franka Emika Panda with 7 degrees of freedom. The robot's Joint Space (JS) limits as well as the CS limits are taken from the manufacturer's official datasheet [191]. These values are publicly available[5] and are listed in Table 5.1 and Table 5.2.

Limits	q_0	q_1	q_2	q_3	q_4	q_5	q_6
$\boldsymbol{q}_{min}$ $[rad]$	-2.8973	-1.7628	-2.8973	-3.0718	-2.8973	-0.0175	-2.8973
$\boldsymbol{q}_{max}$ $[rad]$	2.8973	1.7628	2.8973	-0.0698	2.8973	3.7525	2.8973
$\dot{\boldsymbol{q}}_{max}$ $[rad/s]$	2.175	2.175	2.175	2.175	2.61	2.61	2.61
$\boldsymbol{\tau}_{max}$ $[Nm]$	87	87	87	87	12	12	12

TABLE 5.1: Franka Emika Panda robot JS actuator limits[5]. The lower limits of the JS velocity and torque are symmetric to their upper limits.

Limits	Translation	Orientation
$\dot{\boldsymbol{x}}_{max}$	1.7 m/s	2.5 rad/s
$\ddot{\boldsymbol{x}}_{max}$	13 m/s^2	25 rad/s^2

TABLE 5.2: Franka Emika Panda robot CS kinematic limits[5]. The lower limits of the CS velocity and acceleration are symmetric to their upper limits.

Robot modeling and kinematics as well as simulations of robot dynamics are built using the Python implementation of the `roboticstoolbox` [201] package. The evaluation of the reachable space polytope $\mathcal{P}_x$ and $\mathcal{C}_x$ is carried out using the ICHM algorithm's efficient Python implementation within the `pycapacity`[6] package, described more in detail in Chapter 7. The ICHM's approximation accuracy used for all the evaluations is δ=1mm.

Eight different horizon lengths t_h are chosen for the experiments

$$t_h = [0.05s,\ 0.15s,\ 0.25s,\ 0.5s,\ 0.75s,\ 1.0s,\ 1.5s,\ 2.0s]$$

and the simulation sample time has been set to Δt=5ms.

The proposed method's implementation as well as the code used for the experiments is open-source and can be found in the GitLab repository[7]. The GitLab repository additionally contains a Robot Operating System (ROS) [167] implementation of the proposed method for more interactive evaluation. Furthermore, an efficient implementation of the reachable space polytope can be found as a part of `pycapacity` Python package as well. All the simulations are run on a computer equipped with a 1.90GHz Intel i7-8650U processor and 32Gb of RAM memory.

5.1.3.4 Results of the numerical analysis

This section brings the results of the numerical analysis of the computation time and the accuracy of the proposed reachable space approximation.

Execution time

Figure 5.12 shows the polytope $\mathcal{P}_x$ execution time averaged over 1000 random robot configurations for each of the 8 horizon lengths. From the figure it can be seen that the execution

[5]Full datasheet available at: https://frankaemika.github.io/docs/control_parameters.html
[6]https://auctus-team.github.io/pycapacity/
[7]GitLab: https://gitlab.inria.fr/auctus-team/people/antunskuric/reachable_space/

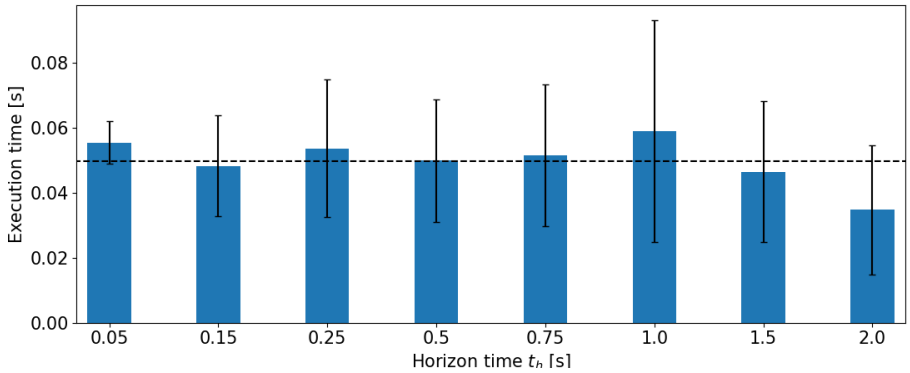

FIGURE 5.12: Execution time of the polytope $\mathcal{P}_x$ enumeration using ICHM algorithm. Average and standard deviation calculated over 1000 executions for each of the 8 horizon lengths.

time is relatively consistent through all the horizon lengths, with an average value around 50 ms. The constant execution time is expected because the horizon length t_h does not have a big influence on the geometrical complexity of the polytope (number of vertices and faces). Rather than on the complexity, as shown on Figure 5.2, the horizon time t_h has an influence on the scale of the polytope.

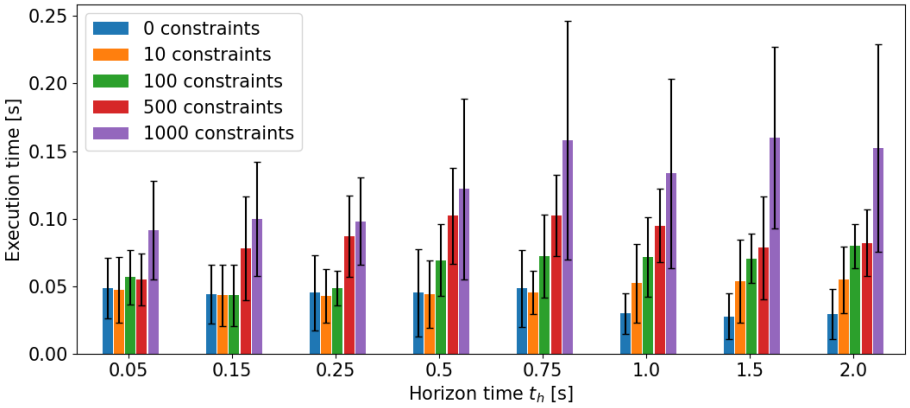

FIGURE 5.13: Average execution time and standard deviation calculated over 1000 executions for each of the 8 horizon lengths, with different number of added environmental constraints, ranging from 0 to 1000.

Figure 5.13 shows the ICHM execution time for the 8 horizon lengths with different numbers of environmental constraints $A_e\boldsymbol{x} \leq \boldsymbol{b}_e$. All the environmental constraints are generated randomly and all the results are averaged over 1000 random robot configurations for each horizon length t_h and all 5 different numbers of environmental constraints: 0 (no environment constraints), 10, 100, 500 and 1000. Results show that up to 10 added environmental constraints the execution time of the method does not change significantly, having the average execution time around 50ms. With 100 environmental constraints the average execution time increases around 20% to around 70ms, for 500 the average execution is 100ms and with 1000 constraints the average execution time more than doubles, to 150ms.

The execution time analysis of the ICHM algorithm has shown that the average polytope $\mathcal{P}_x$ enumeration takes between 50 and 150ms, which opens many doors for potential real-time applications.

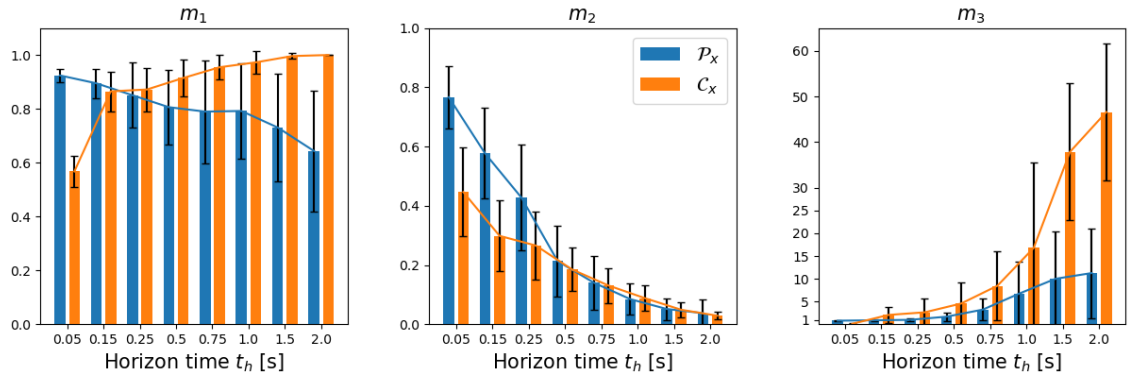

FIGURE 5.14: Accuracy measures calculated for $\mathcal{P}_x$ and $\mathcal{C}_x$, averaged over 100 random poses of the Franka Emika Panda robot for each of 8 horizon lengths.

The average execution time of the nonlinear simulation for t_h=50ms and Δt=5ms is around 60s, and it grows to about 240s for t_h=2s. The cube $\mathcal{C}_x$ average execution time stays constant for all horizon lengths at around 2ms.

Accuracy analysis

Figure 5.14 shows the evolution of the three metrics m_1, m_2 and m_3 with respect to the horizon length t_h for the $\mathcal{P}_x$ and $\mathcal{C}_x$, averaged over 100 random robot configurations.

The first graph on the Figure 5.14 shows the metric m_1. It can be seen that the ratio of simulated robot positions $\boldsymbol{x}_k \in \mathcal{X}$ inside the polytope $\mathcal{P}_x$ decreases with the increasing horizon length t_h, while it increases for the $\mathcal{C}_x$. For larger horizon lengths the robot is able to move far from the initial joint configuration $\boldsymbol{q}_o$ at the beginning of the horizon, making the assumption of linearity of the robot model, made for calculating $\mathcal{P}_x$, largely inaccurate. The approach $\mathcal{C}_x$, on the other hand, improves with the increase in horizon length, since the space bounded by the cube $\mathcal{C}_x$ increases linearly with the horizon length t_h, at some point it encapsulates the whole workspace of the robot. However, it can be seen that even though the ratio m_1 decreases, the polytope $\mathcal{P}_x$ still contains more than half ($\geq$60%) of the simulated robot's reached positions $\boldsymbol{x}_k \in \mathcal{X}$. Finally, it can also be seen that the variance of m_1 increases considerably, lowering the confidence of the polytope $\mathcal{P}_x$ for longer horizons.

Evolution of the metric m_2 is shown on the middle graph of Figure 5.14 . It can be seen that the volume of all the reached robot positions inside polytope $\mathcal{P}_x$ and $\mathcal{C}_x$ decreases considerably with the increase of the horizon time t_h. For the horizon times under $t_h \leq 0.25$s the majority ($\geq$50%) of the reachable space polytope $\mathcal{P}_x$ has been reached by the robot simulations. However for the longer horizon times $t_h \geq 0.5$s the reached volume ratio m_2 drops under 20% of the total volume of the $\mathcal{P}_x$. In the case of $\mathcal{C}_x$, the majority ($\geq$50%) of the volume of the $\mathcal{C}_x$ is never reached by the robot for all tested horizon lengths t_h.

The right graph of Figure 5.14 shows the evolution of the metric m_3. The graph shows that the volume of the polytope $\mathcal{P}_x$ and the cube $\mathcal{C}_x$ becomes several times higher than the volume of $\mathcal{R}_2$ with the increase in the horizon time t_h. For the horizon time of t_h=2s the average volume of $\mathcal{P}_x$ is 12 times larger than the volume of $\mathcal{R}_2$ and the volume $\mathcal{C}_x$ is more than 50 times larger than $\mathcal{R}_2$. At the same time the average reached volume $\mathcal{R}_1$ inside $\mathcal{P}_x$ and $\mathcal{C}_x$ consists of under 10% of their volume (metric m_2), which potentially makes both polytope $\mathcal{P}_x$ and $\mathcal{C}_x$ impractical for many of applications.

However, for the shorter horizon times $t_h \leq 0.25$s the ratio m_3 is very close to 1 for $\mathcal{P}_x$, which means that the reachable space polytope's $\mathcal{P}_x$ volume corresponds well to the actual reached

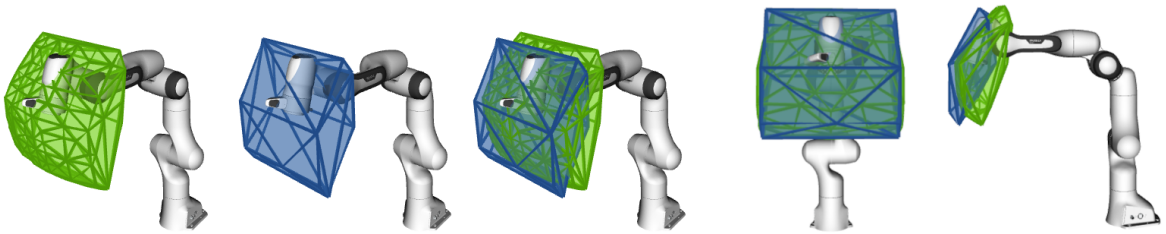

FIGURE 5.15: Comparison of the end-effector reachable space approximation polytope $\mathcal{P}_x$ (blue) and the reachable space obtained using non-liner simulation $\mathcal{R}_2$ (green), from different viewpoints. Both spaces are evaluated for one robot's configuration with the horizon of $t_h = 150$ms.

volume of the robot $\mathcal{R}_2$. Figure 5.15 shows the comparison of the sets $\mathcal{P}_x$ and $\mathcal{R}_2$, for one configuration of a Franka Emika Panda robot.

Overall, this numerical analysis shows that the reachable space approximation based on the convex polytope $\mathcal{P}_x$ works reasonably well for low horizon times $t_h \leq 250$ms. For longer horizon times $t_h > 250$ms the assumption of the linearity of the system produces a large error in the reachable space estimation. In those cases, polytope $\mathcal{P}_x$ has volume several times higher than the real reachable space of the robot (shown by the metric m_3), most of which is not reachable by the robot at all (showed by the metric m_2). Finally, the results show that the polytope $\mathcal{P}_x$, even though a coarse approximation, has better performance across all the defined metrics then the $\mathcal{C}_x$ based on the constant CS limits, at the same time having smaller average volume for the all the horizon times. These results remind, once again, that robot's CS capacity is not constant and by modeling their evolution, even in a coarse manner, a substantial improvement in the estimation accuracy can be reached.

5.1.4 Discussion on limitations and potential applications

Approximating the reachable space of the robot using convex polytopes $\mathcal{P}_x$ is an efficient and relatively fast way of gaining information about the potential robot's position within the horizon of interest. Even though this approximation approach is neither an under or an over approximation of the true reachable space of the robot, as shown in the numerical analysis, for shorter horizon times $t_h \leq 250$ms this metric is reasonably precise. Due to its efficiency, easy and intuitive integration of different environmental constraints, possibility to consider the robot's link geometry and the fact that its output has the form of a convex set of inequality constraints (convex polytope $\mathcal{P}_x$), it has a potential to be a useful tool for robot performance and safety analysis. However there are several important limitations of this approach.

5.1.4.1 Main limitations

First of all, the reachable space of a robot is highly non-convex and nonlinear. As shown in the numerical analysis Section 5.1.3.4, approximating this space with convex polytopes is relatively precise only for small horizon times $t_h \leq 250$ms.

Secondly, robot dynamics is highly nonlinear and time variant as well. When calculating the reachable space polytope $\mathcal{P}_x$, as formulated in Section 5.1.1, the robot model is linearised in the robot's joint configuration $\boldsymbol{q}_o$ and velocity $\dot{\boldsymbol{q}}_o$ at the beginning of the horizon and this linearised model is considered constant during the horizon length. The consideration of constant robot model and its linearity, is only valid for small robot displacements around the current robot state, which is again a strong assumption and only valid for short horizon times.

Finally, the polytope formulation from the Section 5.1.1, considers only constant joint torques $\boldsymbol{\tau}$ applied to the robot during the whole horizon length t_h. The polytope $\mathcal{P}_x$ calculates the joint torque vectors $\boldsymbol{\tau}$ that will produce the largest possible displacements in Cartesian Space (CS) given the robot and environment constraints. This calculation is based on the robot's linearised model at the beginning of the horizon, which in term, once again, limits the precision of the approximation to the small displacements around the initial robot state, preventing longer horizon lengths t_h, as confirmed in the analysis Section 5.1.3.4.

5.1.4.2 Potential applications

One promising field of applications for the polytope based reachable space metric is the human-robot interaction domain. The polytope is essentially a triangulated mesh and can be easily visualised, which could be an easy and intuitive way to give insight to the human operators about robot's real-time capabilities. This metric could potentially be used to increase human operators awareness about the robot's state during the co-manipulation by visualising the polytope $\mathcal{P}_x$ in real-time and even potentially increase the interaction safety. On the other hand, it has a potential to be used for the haptic control applications, as human operators might not be always aware of how close the robot is to the limitation of its capabilities or what is the possible space the robot can be in while teleoperating.

Furthermore, the numerical analysis has shown that for shorter horizon times $t_h \leq 250$ms the volume of the polytope $\mathcal{P}_x$ is comparable to the volume of the simulated reached space R_2, as shown on the Figure 5.14. The volume of the $\mathcal{P}_x$ could potentially be used as a performance metric for robotic workspace analysis and for applications such as robot design.

Finally, as the polytope $\mathcal{P}_x$ can be represented as a set of convex inequalities $H\boldsymbol{x} \leq \boldsymbol{d}$, this metric could potentially be used as a set of constraints for robot control. For example in case of the Model Predictive Control (MPC) [202] where, in many cases, robot's dynamics is already considered linear and constant during given horizon time, which corresponds well with the assumptions of the polytope $\mathcal{P}_x$ formulation.

5.2 Preliminary work: Non-convex approximation of the robot's reachable space

One of the main limitations of the convex polytope based approach, in terms of the impact on its estimation accuracy, is the assumption of linearity of the robot's kinematics

$$\boldsymbol{x} = f_{fk}\left(\boldsymbol{q}_o + \Delta\boldsymbol{q}\right) \approx \boldsymbol{x}_o + J(\boldsymbol{q}_o)\Delta\boldsymbol{q}$$

throughout the horizon time t_h. This assumption can have an undesired effect where large parts of the polytope $\mathcal{P}_x$ can be outside of the robot's workspace, unattainable to the robot regardless of the horizon time. This is particularly noticeable in the cases where the operator is interested in visualising the robot's reachable space for longer horizon times ($t_h > 0.5s$).

Many different methods are available in the literature addressing this issue, namely in the field of robot's workspace analysis. This field brings a vast body of literature on different methods that characterise the set of the robot's reachable positions and orientations, given its actuator limits and its nonlinear kinematics, while being able to guarantee certain bounds on the approximation accuracy. These methods are well known tools for robot design and workplace placement [13, 15], and they are often based on Interval analysis [192] or different exhaustive sampling strategies [14]. However, as these methods are developed for the design and analysis applications, their computational efficiency does not allow for interactive applications.

Therefore, this section brings a preliminary work on a sampling-based approach able to efficiently account for the nonlinear aspects of robot kinematics and improve the accuracy of the reachable space approximation, especially for longer horizon times. However, it is important to acknowledge that both this method and the preceding polytope approximation method lack guarantees on their approximation error.

5.2.1 Sampling-based reachable space approximation strategy

When considering a kinematic robot model in which Joint Space (JS) velocity $\dot{\boldsymbol{q}}$ and position $\boldsymbol{q}$ adhere to constraints

$$\dot{\boldsymbol{q}} \in [\dot{\boldsymbol{q}}_{min}, \dot{\boldsymbol{q}}_{max}], \quad \boldsymbol{q} \in [\boldsymbol{q}_{min}, \boldsymbol{q}_{max}] \tag{5.27}$$

for any given robot configuration $\boldsymbol{q}_o$ and horizon time t_h, the set of all the robot's reachable joint positions $\boldsymbol{q}$ can be specified

$$\mathcal{Q} = \{\boldsymbol{q} \in \mathbb{R}^n \mid \boldsymbol{q} = \boldsymbol{q}_o + \underbrace{\dot{\boldsymbol{q}} t_h}_{\Delta \boldsymbol{q}}, \quad \boldsymbol{q} \in [\boldsymbol{q}_{min},\ \boldsymbol{q}_{max}],\ \dot{\boldsymbol{q}} \in [\dot{\boldsymbol{q}}_{min},\ \dot{\boldsymbol{q}}_{max}]\} \tag{5.28}$$

where $\Delta \boldsymbol{q}$ represents the JS displacement from the initial position $\boldsymbol{q}_o$ during the time horizon t_h. As each of the n components of the joint position $\boldsymbol{q}$ are independent, the set $\mathcal{Q}$ can be geometrically represented as a hyperrectangle in robot's JS.

The set $\mathcal{Q}$ (hypterrectangle) can then be transformed to the task space using the robot's forward kinematics $f_{fk}(\cdot)$

$$\mathcal{R}_x = \{\boldsymbol{x} \in \mathbb{R}^m \ \mid \ \boldsymbol{q} \in \mathcal{Q}, \quad \boldsymbol{x} = f_{fk}(\boldsymbol{q})\} \tag{5.29}$$

obtaining the set of all the reachable task (Cartesian) space positions $\mathcal{R}_x$.

As a comparison, the approach described in the previous section considers the robot's kinematics to be linear during the horizon time, which yields a convex set of the task space reachable space positions that can be represented by a polytope $\mathcal{P}_x$

$$\mathcal{P}_x = \{\boldsymbol{x} \in \mathbb{R}^m \ \mid \ \boldsymbol{q} = \boldsymbol{q}_o + \Delta \boldsymbol{q} \in \mathcal{Q}, \quad \boldsymbol{x} = \boldsymbol{x}_o + J(\boldsymbol{q}_o)\Delta \boldsymbol{q}\} \tag{5.30}$$

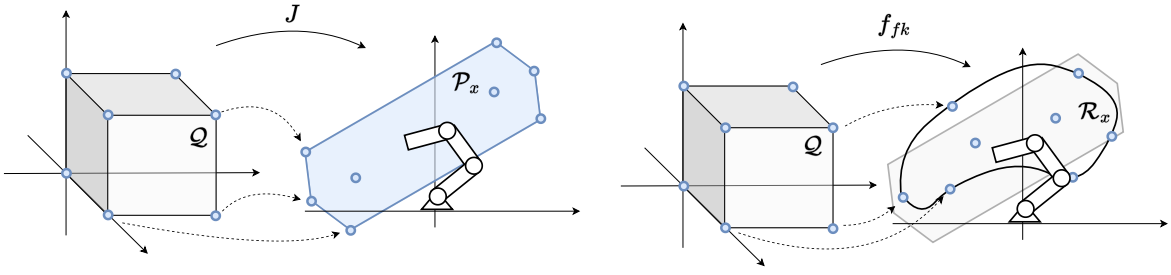

FIGURE 5.16: Simple example of the robot with $n = 3$ joints and planar $m = 2$ task space. The figures illustrate the difference between the linear $\mathcal{P}_x$ and nonlinear $\mathcal{R}_x$ projection of the hyperrectangle $\mathcal{Q}$ to task space.

Figure 5.16 illustrates the difference between the polytope $\mathcal{P}_x$ and the nonlinear reachable space $\mathcal{R}_x$. As shown on the figure, the polytope $\mathcal{P}_x$ is an approximation of the reachable space $\mathcal{R}_x$, which is, in general case, nonlinear and non-convex.

Furthermore, the vertices of convex polytope $\mathcal{P}_x$ are a subset of projected vertices of the the hyperrectangle $\mathcal{Q}$. However, in the case of the set $\mathcal{R}_x$, which uses the nonlinear projection through the robot's forward kinematics, the projected vertices of the hyperrectangle $\mathcal{Q}$, are

either projected to its interior or they become a part of the surfaces bounding the robot's reachable space. In general case the projected vertices are no longer the extreme points of the non-convex reachable space $\mathcal{R}_x$.

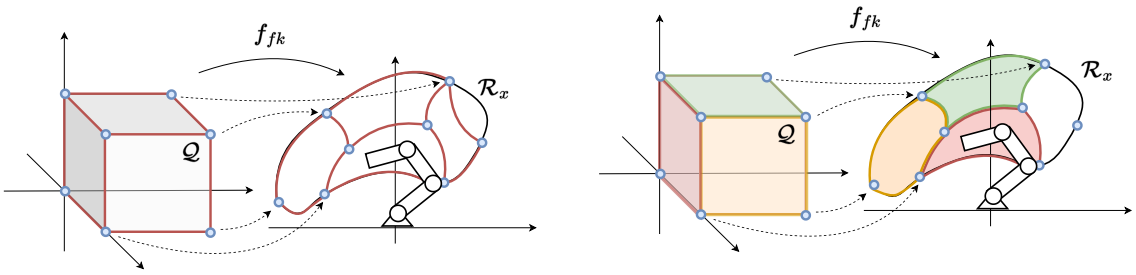

FIGURE 5.17: The figures illustrate the property of the critical points to be the intersection of the $r = 1$ dimensional faces and $r = 2$ dimensional faces both in JS and when projected to the task space.

These projected vertices are also called critical points [203]. As the forward kinematics $f_{fk}(\cdot)$ is a continuous function, these critical points represent the intersections of the edges and faces of the set $\mathcal{Q}$ projected in the task space, as illustrated on Figure 5.17. Therefore, starting at any of the critical points, any other point in the space $\mathcal{R}_x$ can be reached (including the boundaries) by navigating the projected edges and faces of the set $\mathcal{Q}$.

Leveraging this property, this work proposes a simple approximation approach of the space $\mathcal{R}_x$, based on sampling of the edges and faces of the set $\mathcal{Q}$. This approach consists in sampling the r dimensional faces ($r = 0$ vertices, $r = 1$ edges, $r = 2$ facets, etc.) of the hyperrectangle $\mathcal{Q}$ and projecting them to the task space using the forward kinematics function $f_{fk}(\cdot)$. This approach allows for an efficient sampling of $\mathcal{R}_x$, where its approximation can be then obtained by triangulating the obtained sample points.

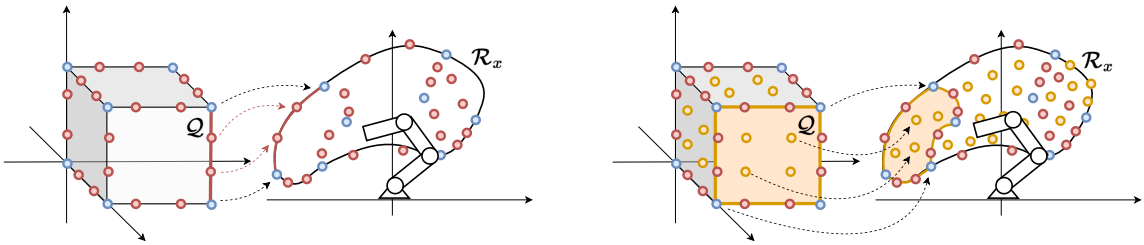

FIGURE 5.18: The figures illustrate the proposed sampling-based strategy, by choosing to sample $r = 1$ dimensional faces of the hypercube $\mathcal{Q}$ on the left and the $r = 2$ dimensional faces on the right, while using uniform sampling with $s = 4$ samples per axis.

The proposed approach is more computationally efficient than the standard sampling methods, as it does not require sampling the complete n dimensional hyperrectangle $\mathcal{Q}$, rather its r dimensional faces, where $r \leq n$. Figure 5.18 illustrates the sampling strategy.

The proposed approach can be divided in three steps:

1. Sample all the r dimensional faces of the hyperrectangle $\mathcal{Q}$
2. Project them to the task space using $f_{fk}(\cdot)$
3. Triangulate the obtained points

Face dimension and sampling complexity

Determining the optimal parameters for implementation, including the dimension r of the faces to be sampled and the sampling strategy, profoundly influences both the accuracy of the approximation and the execution time. These considerations are important as they describe the trade-off between approximation accuracy and time efficiency.

Increasing the dimension r of the sampled faces and elevating the sampling resolution holds the potential to improve approximation accuracy. However, these improvements may come at the cost of extended execution times.

The number of r dimensional faces of the n dimensional hyperrectangle can be found using the equation

$$n_f = 2^{n-r}\binom{n}{r}$$

Moreover, in the context of uniform sampling with s samples along each axis, the number of samples per r dimensional face has an exponential relationship

$$n_s = s^r$$

As a result, the overall number of sampled points, given by $N_s = n_f n_s$, is exponentially related to the parameters r and s. Therefore the appropriate values for r and s should be determined with respect to the application's requirements, in terms of the approximation accuracy and the computational complexity.

Triangulating the non-convex space

Furthermore, as the reachable space $\mathcal{R}_x$ is not convex, the triangulation of sampled points can be executed in multiple ways. If the application requires it, the sampled points can be triangulated using different Convex-Hull algorithms or Delaunay triangulation [204], in order to convexify the obtained space. On the other hand, numerous non-convex triangulation methods are available in the geometric algebra, such as Alpha-shapes [205] triangulation. Such techniques represent better the true reachable space, but are more computationally complex. For example, in a case of a set of n 3D points, the Alpha-shape algorithm's complexity is $O(n^2)$ [205], while the Convex-Hull algorithms usually have a complexity of $O(n \log n)$ [74].

Open-source implementation

A publicly available Python implementation of the proposed sampling-based reachable space approximation method can be found on GitLab[8] as well as a short video[9] demonstration of the method running in real-time. The implemented method uses the library `pinocchio` [166] for robot's forward kinematics while the non-convex Alpha-shape triangulation is implemented using `CGAL` [206] library.

Video

Figure 5.19 shows the visualisation of the approximation of the reachable space $\mathcal{R}_x$ using the proposed sampling method for different horizon times, while Figure 5.20 shows several different joint configurations for the same horizon times.

[8] GitLab: https://gitlab.inria.fr/auctus-team/people/antunskuric/example/reachable_workspace
[9] Video: https://youtu.be/s_bjTfJhDag

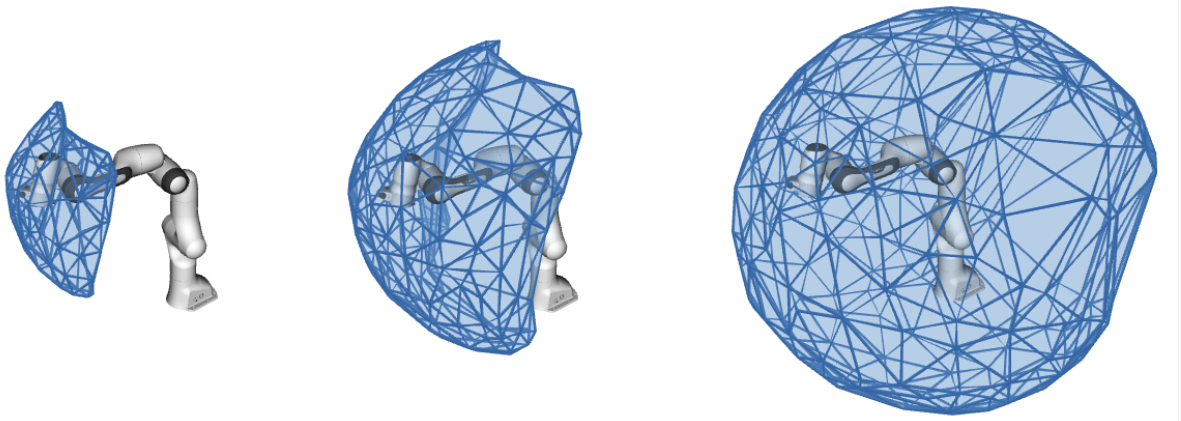

FIGURE 5.19: Figure showing the approximation of the reachable space $\mathcal{R}_x$ of a Franka Emika Panda robot in one configuration for three different horizon times: $t_h = 100$ms, 200ms and 500ms (from left to right). Figures show that the approximated space is highly non-convex and for longer horizon times it tends to approach the robot's full workspace [191]. The chosen sampling parameters are $r = 3$, uniform sampling with $s = 5$ and the execution time was around 500ms.

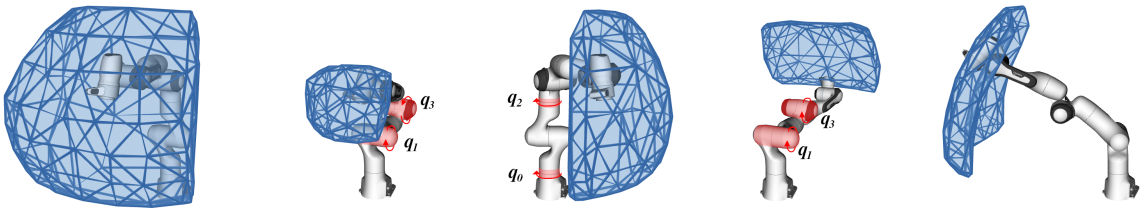

FIGURE 5.20: Figure showing the approximation of the reachable space $\mathcal{R}_x$ of a Franka Emika Panda robot in five different configurations for the same horizon time: $t_h = 150$ms. The first on the right shows the robot in its initial configuration, while the three following figures show the robot in configurations where two out of its seven joints are in their position limits, where the influence of robot's joint position limits can be seen through its reachable space size and shape, similar to Figure 5.3. The figure on the right, demonstrates the non-convex nature of the robot's reachable space, showing the robot in its near-singular configuration. In this configuration its movement capacity is high in tangential directions and low in radial ones. The chosen sampling parameters are $r = 2$, uniform sampling with $s = 3$ and the execution time was around 50ms.

5.2.2 Discussion on limitations

The proposed sampling-based approach holds potential for improving reachable space approximation accuracy and provide the operator with more accurate representation the robot's physical abilities. However, it has several limitations.

The proposed method lacks formal guarantees on approximation error and, in its current form, it does not represent neither an under nor an over-approximation of the true reachable space $\mathcal{R}_x$. Furthermore, while all sampled points projected to the task space are attainable by the robot, once triangulated, the obtained space might include areas that are unattainable.

The approach's accuracy and computational efficiency heavily depends on parameter choices, like the dimension of sampled faces r and the sampling strategy. Additionally, computational efficiency of forward kinematics influences overall effectiveness as well.

Future refinements are needed to focus on addressing these limitations and to enhance the method's applicability and reliability, and in that way the quality of the information transmitted to the operator.

5.3 Preliminary work: Interactive polytope visualisation platform

In the context of this thesis an initial work on the development of a testing platform for the interactive visualisation of different polytopes is conducted. The long-term aim of the testing platform is to be enable comparing different modalities of polytope visualisation to the operators and help to understand and evaluate their effectiveness in different human-robot collaboration scenarios. In this preliminary work the focus is put on real-time interactive visualisation of polytopes using the Augmented Reality (AR) devices. More specifically the initial setup is built around a Microsoft HoloLens 2 device, as one of the widely used, of-the-shelf and affordable AR devices.

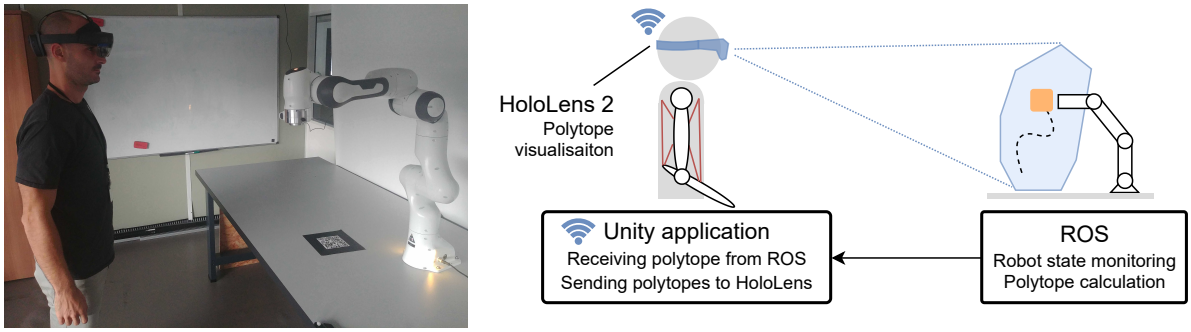

FIGURE 5.21: Figure shows the conceptual overview of the developed platform. Robot Operating System (ROS) is used to gather information about the robot's current state, which is then used to calculate different polytopes of physical abilities and prepare them for visualisation. The polytopes in form of meshes are sent to the Unity application which updates the virtual world. The updated virtual world (with polytopes) is then visualised in real-time to the operator using Microsoft HoloLens AR headset.

Achieving the real-time visualisation of the physical ability polytopes with AR devices, while making sure that the polytopes are visualised at the right place (ex. robot's end-effector), and synchronised with the robot's movements, involves several key steps. It requires real-time monitoring of the robot's state in order to calculate the appropriate polytopes and transform them to the form of triangulated meshes. The polytopes are then communicated to AR devices, which ultimately provide the operator with the real-time interactive visualisation.

The architecture of the implemented system is illustrated in Figure 5.21. ROS is used to gather real-time data on the robot's state, while `pinocchio` [166] library is used to calculate the robot's dynamics and kinematics. The computation of various polytopes was implemented using the `pycapacity` Python package. Once the polytopes are transformed to the form of triangulated meshes, they are transmitted to the Unity framework. The Unity application updates the virtual world with the new data and transmits the polytopes in real-time to the Microsoft HoloLens 2. This AR device efficiently visualises the polytopes to the operator, at the appropriate position on the real robot (ex. robot's end-effector), providing an immersive visualisation experience for the operator.

Some components of this setup have been created through an informal collaboration with the Institute for Experiential Robotics at Northeastern University, building upon their recent work described in Zolotas *et al.* [79]. Additionally, a significant portion of the platform's development has been undertaken during Claire Houziel's internship, she is pursuing her master's degree in "Robotics and Rehabilitation" at Sorbonne University.

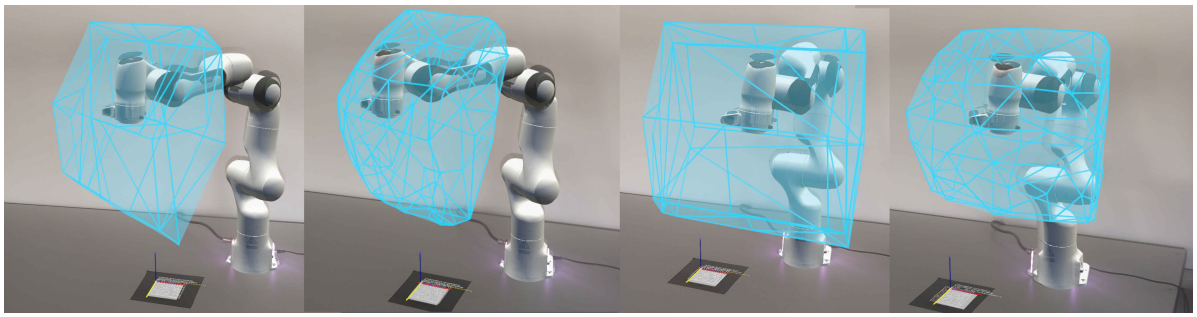

FIGURE 5.22: Figure shows the screenshots taken from the operator's perspective using Microsoft HoloLens headset. Images show two different robot's configurations and for each one two approximations of the robot's reachable space are shown: convex polytope approach described in Section 5.1 and non-convex approach from Section 5.2. Convex polytope is calculated with the horizon time of $t_h = 200$ms, while the non-convex approximation is calculated for $t_h = 100$ms.

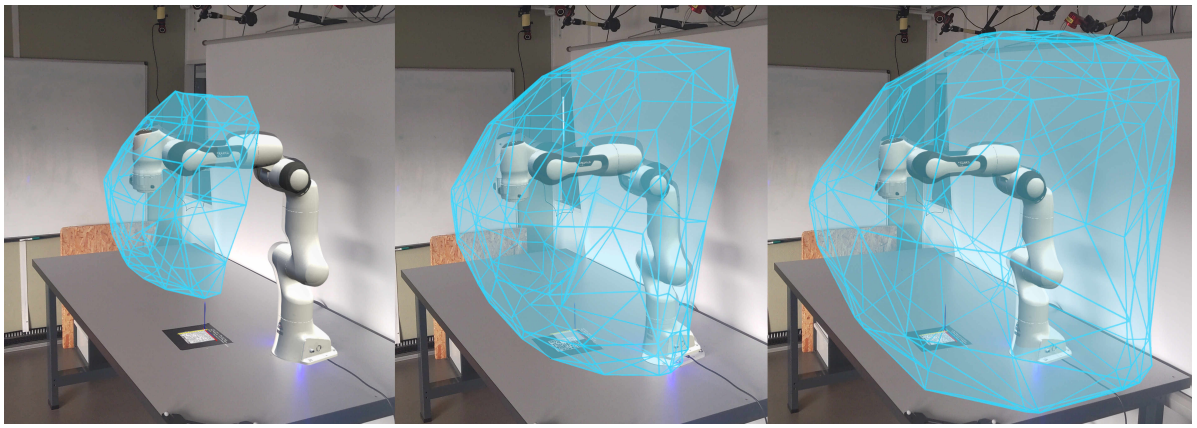

FIGURE 5.23: Figure shows screenshots form the Microsoft HoloLens headset. The non-convex approximation of the robot's reachable space, described in Section 5.2 , is calculated for one robot's configuration and three different horizon times $t_h = 100$, 200 and 300ms (from left to right).

The implemented setup allows for interactive visualisation of different polytope metrics in real-time, which serves as a base for the future work that will concentrate on the evaluation of their effectiveness for sharing information with the operator. Figure 5.22 and Figure 5.23 show visualisations of several physical ability polytopes of the Franka Emika Panda robot using the developed AR platform. Additionally, a publicly available video demonstration of the interactive visualisation of polytope metrics can be found using link[10].

Video

5.4 Conclusion

This chapter discusses the concept of utilising polytopes as a visualisation tool to inform operators about the robot's real-time state and its changing physical capabilities. By providing dynamic and interactive visualisations of polytope based metrics, this approach holds the potential to significantly enhance operators' situational awareness. Not only enhancing operator safety by allowing them to better understand the robot's capabilities and limitations, but also having the capacity to improve overall collaboration performance.

[10]Video: https://youtu.be/0O9b9iBzWyA

Polytopes offer a convenient visualisation method due to their capacity to be transformed into triangulated meshes, a format compatible with most standard visualisation tools. Among these tools, Virtual Reality (VR) and Augmented Reality (AR) devices emerge as particularly promising platforms, offering immersive and interactive visualisation experiences. Nevertheless, the challenge lies in determining which specific physical ability polytope to present to operators and selecting an appropriate visualisation mode. This task is challenging, primarily due to the intrinsic complexity of polytopes, containing a large number of faces and vertices that may overwhelm operators' cognitive capacity when extracting pertinent information. Furthermore, the common formulations of physical ability polytopes often characterise abstract physical quantities, like accelerations or forces, which may not intuitively translate to operators' understanding once displayed. The convergence of these challenges highlights the need for careful consideration of both the choice of physical ability polytope and the visualisation mode to ensure effective communication of critical information to operators.

This chapter, in Section 5.1.1, introduces a novel polytope formulation for robotic manipulators, which provides an approximation of the manipulator's reachable space within a given horizon time. This polytope formulation is particularly suitable for operator visualisation, as unlike traditional polytope formulations, it represents the set of robot's reachable positions, thereby simplifying interpretation for operators. Additionally, it has the capacity to integrate actuator constraints and robot's movement capacity at the same time. Moreover, it can be intuitively extended to include the environmental constraints as well as the robot's link geometry. Finally, with the execution time of around 50ms, for standard 7DOF collaborative robot, it has a potential to be used for interactive visualisation. However, it's important to note that this formulation relies on a linearized robot model, consequently limiting its applicability to scenarios with shorter horizon times. The analysis conducted in Section 5.1.3 shows that the accuracy of the approximation significantly diminishes for horizon times exceeding 250 milliseconds. The proposed convex polytope approximation approach have been published as a part of a scientific article **A. Skuric** *et al.* [A5].

To enhance the approximation accuracy, particularly for longer horizon times, and to provide the operator with more accurate long-term information, Section 5.2 introduces a preliminary work on the sampling-based reachable space approximation strategy. This method takes into account the robot's nonlinear kinematics and allows the non-convex characterisation of the robot's reachable spaces. Therefore, the proposed method has a potential to offer a more accurate characterisation of robot's reachable space than the convex polytope based approach. However, like the convex polytope based approach, this strategy lacks formal guarantees on its approximation accuracy, necessitating further research to enhance reliability and information quality conveyed to operators.

Finally, in the effort to evaluate the effectiveness of the real-time polytope visualisation to the operators, Section 5.3 presents the preliminary work on the development of the testing setup based on AR tools. In the context of this thesis an initial setup is developed capable of visualising different polytopes to the operator using Microsoft HoloLens. The polytopes are visualised at the appropriate location on the real-robot (ex. end-effector), updated in real-time and synchronised with the movements of the robot. This setup presents the foundation for the future work on evaluating the information sharing potential of different polytope formulations and different visualisation modalities, in the context of the human-robot collaboration.

Following chapter, Chapter 6, proposes a new Cartesian Space (CS) trajectory planning approach exploit robot's full movement capacity, while at the same time remaining reactive to the potential changes in the environment. Chapter 7 presents the publicly available open-source software Python package `pycapacity`. The package provides the efficient implementation of algorithms for evaluating polytope and ellipsoid based physical abilities of humans and robots.

Chapter 6

Polytopes for time-efficient and reactive Cartesian Space trajectory planning

Chapter 2 shows that polytopes are an accurate characterisation of physical abilities for both robots and humans. Moreover, efficient polytope algebra tools potentially open doors for their use in real-time applications, as demonstrated in Chapter 3.

One such application is showcased in Chapter 4, demonstrating the potential of using real-time evaluation of human's and robot's physical ability polytopes in the context of the human-robot physical interaction. The polytopes are used for creating more flexible robot control strategies that adapt to the changing physical abilities of both humans and robots. A different polytope application scenario is brought in Chapter 5, proposes the use of the physical ability polytopes as visual communication tools. Such real-time visualisation has a potential to provide insight to the operators about the robot's current states and its current physical abilities in real-time.

This chapter brings the application of the polytope representation of robot's movement capacity (its physical ability to generate movement: velocities, accelerations, etc.) for efficient and reactive planning of robot's trajectories in Cartesian Space (CS). As discussed in Section 2.1, robot's CS movement capacity is robot's state dependent and can change significantly while executing different trajectories.

Leveraging the efficient tools form polytope algebra, this chapter proposes a new trajectory planning approach consisting in evaluating the robot's movement capacity in real-time and using it to adapt the planned trajectory to account for its changes. By re-planning in each step of the trajectory execution, the proposed approach is able to fully exploit robot's constantly changing movement capacity and at the same time be reactive to the potential changes in the trajectory itself, induced by the robot's environment. These properties are particularly interesting in the human-robot collaboration context, as the collaborative robots are often limited in performance, relying on the efficient use of their abilities, and working in human environments which is commonly unstructured and highly dynamical.

The motivation for the proposed approach is described in Section 6.1, followed by Section 6.2, which brings the intuition about the difficulties of planning robot's trajectories in CS. The efficient method for evaluating robot's CS movement capacity is described in Section 6.3, while the proposed real-time re-planning approach is introduced in Section 6.4.

The performance of the proposed planning method is compared against the state-of-the-art trajectory planning methods in Section 6.7. An applicative experiment of the proposed method in the context of collaborative waste sorting is brought in Section 6.8.

6.1 Motivation: Time-optimal and reactive trajectories

The field of collaborative robotics has seen a unprecedented growth in recent years with the development of safer and cheaper robots with promising applications in industry, research, and even everyday life [207]. However, as the complexity of the environments in which these robots operate increases, planning for robot trajectories becomes a significant challenge. The robots need to dynamically adapt to changing environmental conditions and tasks, as well as the presence of humans. At the same time, in order to be safe, collaborative robots tend to be smaller and relatively limited in performance compared to more traditional industrial robots [169]. Therefore, it is becoming increasingly important to utilise the physical abilities of these robots fully in order for their applications to become viable, rather than investing in larger, more expensive industrial robots.

When it comes to planning robot motions, the most common approach is to decouple path and trajectory planning [208]. First, the robot's geometric path is found, accomplishing certain task and potentially avoiding the obstacles in the environment. Then the optimal sequence of movements along this path is calculated in order to optimise certain criteria, the most common ones used in the literature being minimum energy, minimum jerk and minimum execution time. Among these criteria, the one that aims to exploit full robot's movement capacity corresponds to the minimum time criteria, resulting in time-optimal (or minimum time) trajectories [209].

Traditional time-optimal trajectory planning techniques, developed for industrial robots, find trajectories with the highest possible speeds, within robot's and task constraints, along pre-defined paths. These methods are often defined in robot's Joint Space (JS) calculating the necessary optimal motions of each one of the robot's joints. Time-optimal algorithms, such as ones proposed Bobrow *et al.* [210] and recently by Pham and Pham [211], assume full in advance knowledge about both robot's path and the environment. However in collaborative scenarios, environment is often dynamic and conditions may change rapidly, requiring real-time changes in the trajectory and the task. To account for these changes the trajectory needs to be adapted in real-time, however time-optimal approaches have long execution times due to their computational complexity, making such implementations unpractical.

On the other end of the spectrum, the real-time capable trajectory planning approaches often abstract the robot as a generator of ideal movements without accounting for its true capacity. These methods are often defined in robot's task space, the space of robot's CS positions and orientations. With this simplification, they have short execution times and can quickly adapt to any changes in the robot's path, environment or the task itself. Such algorithms, as proposed by Macfarlane and Croft [212], Haschke *et al.* [213] or Svarny *et al.* [214], are often based on S-curve trajectories [215] with an appropriate order, producing smooth trajectories that respect a set of fixed velocity, acceleration, jerk and sometimes even snap (jerk derivative) limits enforced by the task itself and robot's movement capacity that is assumed to be constant [216]. However, the robot's CS movement capacity is not constant and can vary significantly during the trajectory execution. Therefore such approaches are not capable of fully exploiting the robot's capacity, resulting in trajectories that might underestimate or sometimes even overestimate its movement capacity. Figure 6.1 shows an example of this effect on a Franka Emika Panda collaborative robot.

To bridge this gap, several approaches were proposed recently, aiming to exploit full robot's movement capacity while allowing for reactive adaptations online. These methods often decouple time-optimality and reactivity problems and frequently involve varying degrees of robot model simplification. For instance, Palleschi *et al.* [217] proposed a decoupling method that plans for time-optimal robot motions while guaranteeing human safety. The approach consists in offline pre-calculation of the time-optimal trajectory that exploits full robot's movement

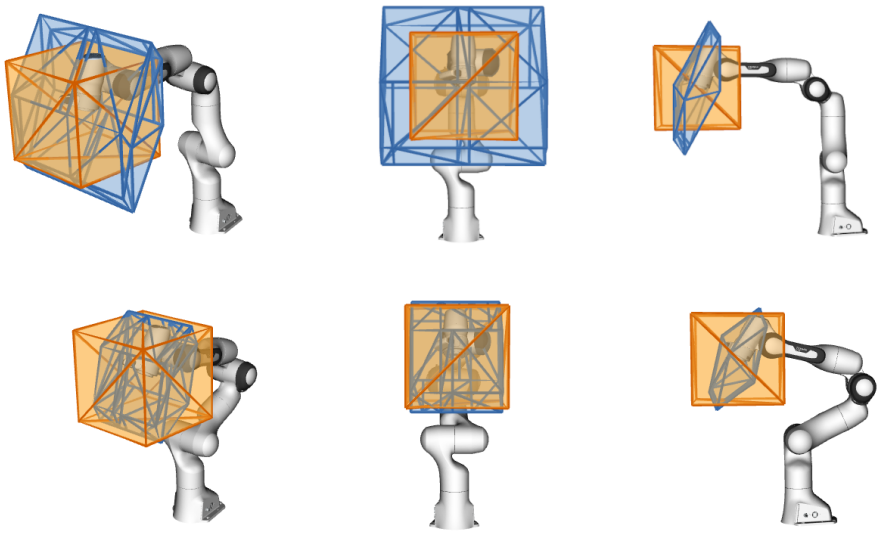

FIGURE 6.1: End-effector linear velocity limits as computed using the fixed Cartesian Space (CS) limits provided by the manufacturer (orange) and using an estimation based on the JS velocity limits and the current configuration of the robot (blue) (scale: 1 m.s^{-1}/ 10 cm). The true robot's velocity capacity are in some directions higher and in the others significantly lower than the manufacturer's limits, depending on the configuration.

capacity, while the reactivity is ensured online using the reactive re-planning approach. A different decoupling approach that uses Dynamic Motion Primitives (DMP) is proposed by Springer *et al.* [218]. Their method calculates the DMP models of the time-optimal point-to-point trajectories of the robot in-advance, trying to exploit its full movement capacity. Then these DMP based trajectories are exploited in real-time to adapt to changing targets. However, even though the offline trajectories calculated by these decoupled methods are time-optimal, their real-time adaptations might lose this property. In order to avoid the decoupling, Zhang *et al.* [219] proposed a real-time method for time-optimal trajectory planning based on simplifying the robot's model. The approach uses machine learning to learn the simplified robot's dynamical model offline. The simplified model reduces the computational complexity of the time-optimal planning and allows for real-time reactive re-planning. However, this method requires a substantial degree of in-advance computation in order to work properly.

Optimal control methods like Model Predictive Control (MPC) [202] offer a promising avenue for generating reactive and time-optimal trajectories in real-time. MPC approaches determine the optimal action to be executed by the robot, given its current state and predictions of its future states, obtained using the robot's model. MPC updates its prediction of the robot's future states in each step of the trajectory execution, taking in account the changes in robot's current state and its environment. Therefore, they can account for the evolving movement capacity of a robot and react to the changes in its environment [220][221]. However, as robot models are highly nonlinear, the implementation of these methods remains relatively complex and challenging, often relying on nonlinear optimisation techniques [222, 223].

Hence, this chapter makes a step in this direction, proposing a trajectory planning approach capable of producing time-efficient and reactive trajectories by re-planning in real-time. The proposed approach evaluates the robot's CS movement capacity in real-time, using the efficient tools from polytope algebra, and re-plans the time-optimal trajectory in each time-step. The time-optimal planning is based on Trapezoidal Acceleration Profile (TAP) [216] planning, which computational efficiency enables real-time execution. By evaluating robot's CS movement capacity and re-planning the time-optimal trajectory in each step of the trajectory execution,

the approach is capable of adapting to the changes in robot's movement capacity without requiring any pre-computation. Additionally, the real-time re-planning enables reacting to potential changes in the robot's trajectory induced by its environment, making it well-suited for dynamic environments where conditions change rapidly such as human-robot collaboration.

Experimental results, described in Section 6.7, demonstrate the effectiveness of the proposed approach on a Franka Emika Panda collaborative robot. The proposed method is compared against a state-of-the art offline time-optimal method TOPP-RA [211], where the results show that the proposed algorithm generates the trajectories that have comparable execution time (even shorter in some cases) than offline method. The method is furthermore compared to the standard reactive approach based on TAP planning with fixed CS movement capacity given by the manufacturer. The analysis confirms that the proposed method achieves faster and more precise trajectories than the standard reactive approach with fixed capacity assumption.

Finally the reactivity of the proposed approach is demonstrated in a mock-up experiment in the context of collaborative waste sorting, described in Section 6.8. In this experiment the proposed planning method is used to generate time-efficient and adaptable robot's trajectories in order to pick the waste items introduced by the operator and place them in the appropriate sorting bins.

The intuition about robot's movement capacity aware trajectory planning in CS is given in Section 6.2. The description of the efficient method for robot's CS movement capacity evaluation is described in Section 6.3. Then, Section 6.4 gives an overview of the Cartesian space TAP planning and the description of the real-time planning strategy. Section 6.5 discusses several potentially undesirable effects of the proposed method and proposes the approaches to overcome them. The experimental validation of the proposed method is brought in Section 6.7 and Section 6.8. Finally, the limitations and perspectives of the proposed method are discussed in Section 6.9.

6.2 Problem statement: Capacity aware trajectory planning

Time-optimal trajectory planning consists in finding the robot's movements with the highest possible speeds, within robot's and task constraints, along predefined paths. Therefore it requires a thorough understanding of the robot's movement (ex. velocity, acceleration, jerk) capacity along the specified path.

Robot's actuation capabilities are usually expressed in the n dimensional Joint Space (JS). Where, for each one of the n joint actuators, the limits are specified by manufacturers in the form of independent min-max ranges of different joint variables. As opposed to the robot's actuator limits, the desired path to follow is often defined in the task space, often $m = 3$ dimensional space of Cartesian Space (CS) positions or $m = 6$ if orientations are specified as well. In the context of this chapter, robot's paths are considered to be CS point-to-point straight line path from one pose X_a to another X_b (for example expressed as an homogeneous transform matrix in $SE(3)$). CS straight line paths are commonly used as they represent the shortest paths in the CS between the two CS positions, and in the context of human-robot interaction they have shown to produce predictable robot's motions for the operators [224]. Additionally, they can be used to construct more complex paths by approximating them with a set of straight line segments.

Traditional approaches to the time-optimal trajectory planning consist in transforming the path to JS, where they find the fastest movements of the robot's joints following the path and respecting the actuator limits. This approach is convenient because the robot's movement capacity in JS can reasonably be considered constant and each joint limit can be specified

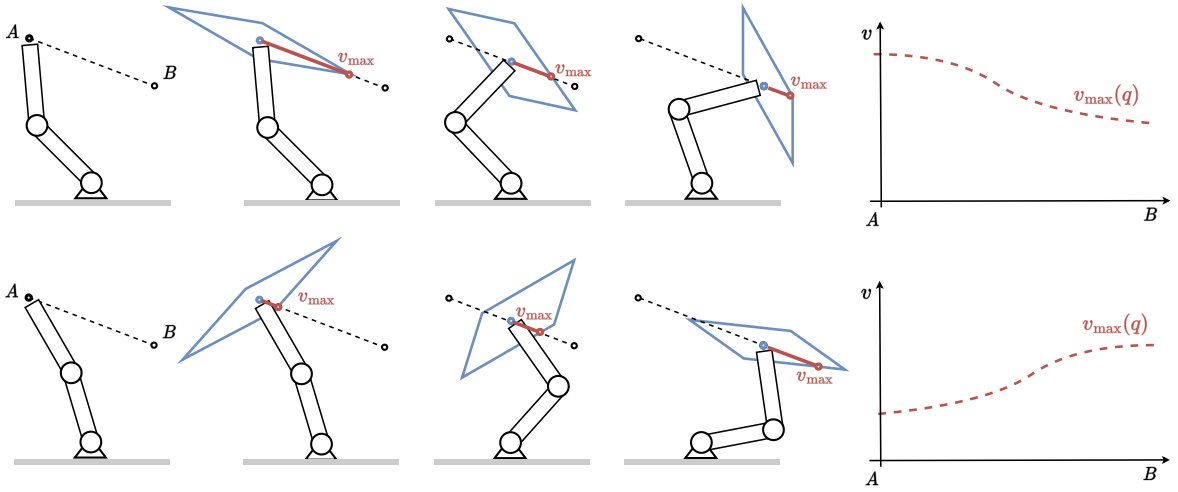

FIGURE 6.2: The figure illustrates the robot's CS linear velocity capacity evolution while following a CS straight line path, from point A to B, using two different JS paths (up and down). The robot's CS velocity capacity polytope is shown in blue, while its capacity to generate velocity in the path direction is shown in red. The path is shown in black dashed lines. The graph on the right shows the evolution of the robot's velocity capacity in the path direction while executing the trajectory. The figure shows that the robot's movement capacity in the path direction can evolve significantly during the trajectory execution, as well as that it depends highly on the JS path taken.

as an independent min-max range. However, the main limitation of this approach is that JS is usually higher dimensional than CS ($n > m$), making the mapping between the CS and JS not unique. In practice this means that there are multiple JS paths that accomplish the same CS path. Each one of these JS paths has different properties which makes finding the optimal JS path a challenging problem of its own, and one with a substantial computational cost. Additionally, having chosen an appropriate JS path, the robot's redundant degrees of freedom are fixed even though they do not participate in movement generation, preventing it from executing other tasks and further limiting this approach.

Complementary approach consists in transforming the robot's joint actuator limits to the CS and planning the time-optimal trajectory in CS directly. Such method results in the CS trajectory which respects all the robot's actuator limits without explicitly choosing the JS path. As discussed in Chapter 2, once the robot's actuator limits are transformed to CS, they form the polytopes of robot's CS movement capacity. The robot's movement polytopes are highly dependant on its state and can evolve significantly during the trajectory execution. Therefore, changing robot's CS movement capacity presents one of the main challenges of time-optimal trajectory planning in CS. Figure 6.2 illustrates robot's changing CS velocity capacity along the trajectory for two different JS paths executing the same CS path.

A common approach to simplify this issue is to consider robot's CS movement capacity constant during the trajectory execution. For example by choosing a fixed set of limits that correspond to the worst-case robot's movement capacity along the path. Such approach is, by definition, not capable of fully exploiting the robot's movement capacity, underestimating considerably robot's actual abilities. Furthermore, finding the worst-case underestimation of the robot's movement capacity often involves characterising all the robot's JS configurations possible for the same path, which is not always practically possible and results in long execution times. A more common approach, especially in the industry, is much more empirical. It involves manually finding the appropriate set of fixed CS limits by specifying certain percentage of the robot's CS limits predefined by the manufacturers [190][225]. This tuning approach has to be

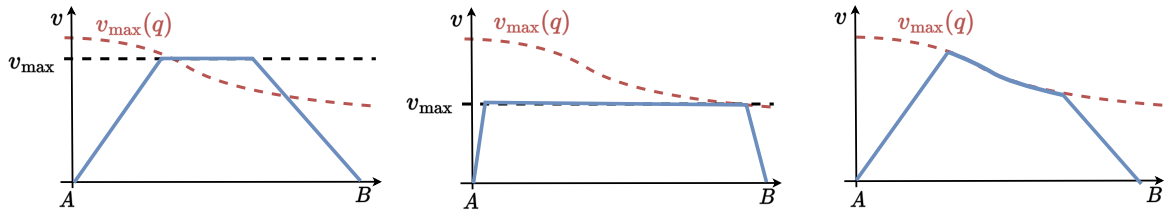

FIGURE 6.3: The figure illustrates three different strategies for planning for CS trajectory with respect to their consideration of the robot's velocity capacity limits in the path direction. The planned trajectories are shown in blue while the robot's true velocity capacity when executing the trajectory is shown in red dashed lines. Figures on the left and in the middle represent the planning methods which assume fixed velocity capacity during the trajectory execution (black dashed line). Strategy on the left, due to the robot's changing capacity, at the beginning of the trajectory underestimates robot's ability and towards the end overestimates robot's abilities, resulting in infeasible trajectory. Strategy in the middle uses the worst-case velocity capacity on the path as its limit, resulting in feasible trajectory however significantly underestimating robot's abilities. Strategy on the right shows an example of a true time-optimal trajectory exploiting fully robot's changing velocity capacity.

repeated for each trajectory of interest in order to find a satisfactory trade-off between robot's speed and the tracking error. Manual tuning can be time consuming and, in many cases, results in sub-optimal trajectories planed without any real information about the robot's true motion capacity.

Therefore, choosing any fixed set of CS limits leads to either overestimation of the robot's movement capacity or its underestimation, as shown on Figure 6.3. The overestimation can lead to planning for trajectories that are unfeasible for the robot, while underestimation leads to sub-optimal robot's trajectories, that do not exploit its full movement potential. Planning for a true time-optimal trajectory would involve accounting for the robot's changing movement abilities along the trajectory. However, as the robot's movement capacity depends on its joint states, evaluating the robot's movement capacity along the path in advance would require transforming the CS path to JS. Such approach would lose most of the benefits of CS trajectory planning: resulting in high computational cost, requiring making choices about the appropriate JS path and fixing the robot's redundant degrees of freedom.

Rather than trying to account for the robot's changing movement capacity in the offline planning stage, this chapter proposes a method that calculates the robot's movement capacity in real-time and adapts the time-optimal trajectory with the updated limits in each step. The proposed method calculates the robot's movement capacity, in the path direction, using efficient tools from polytope algebra in real-time. Trapezoidal Acceleration Profile (TAP) planning is then used to recalculate the time-optimal trajectory with the updated movement capacity is each step. In this way, this CS trajectory planning method is able to adapt to the robot's changing movement capacity, while not making any a priori choices about the robot's JS path. Additionally, the real-time re-planning allows the method to be reactive to the potential changes in the desired robot's path induced by the robot's environment.

The real-time method for robot's CS movement capacity evaluation in the path direction is described in Section 6.3. While Section 6.4 presents the real-time TAP based trajectory re-planing method. Section 6.7 then brings a comparative study of the performance of the proposed method against the state-of-the-art JS time-optimal method TOPP-RA [211], as well as against the CS planning approach considering fixed limits.

6.3 Evaluating robot's Cartesian Space movement capacity in real-time

Robot's joint actuator limits are usually given by the manufacturer as ranges of the feasible joint positions $\boldsymbol{q} \in \mathbb{R}^n$, velocity $\boldsymbol{q} \in \mathbb{R}^n$, acceleration $\dot{\boldsymbol{q}} \in \mathbb{R}^n$ and jerk $\ddot{\boldsymbol{q}} \in \mathbb{R}^n$

$$\dddot{\boldsymbol{q}} \in [\dddot{\boldsymbol{q}}_{min}, \dddot{\boldsymbol{q}}_{max}], \quad \ddot{\boldsymbol{q}} \in [\ddot{\boldsymbol{q}}_{min}, \ddot{\boldsymbol{q}}_{max}], \quad \dot{\boldsymbol{q}} \in [\dot{\boldsymbol{q}}_{min}, \dot{\boldsymbol{q}}_{max}], \quad \boldsymbol{q} \in [\boldsymbol{q}_{min}, \boldsymbol{q}_{max}] \tag{6.1}$$

The true Joint Space (JS) acceleration $\ddot{\boldsymbol{q}}$ and jerk $\dddot{\boldsymbol{q}}$ limits are a consequence of the constrained nature of the robot's actuator torques $\boldsymbol{\tau}$ and torque derivatives $\dot{\boldsymbol{\tau}}$, and have a polytope form. In this work, these limits are considered to be given in a form of independent ranges (6.1). However, the proposed approach is not restricted by this assumption and can be extended to account for the polytope limits as well.

The mapping between the robot's n dimensional JS and m dimensional Cartesian Space (CS) velocity, acceleration and jerk for a certain fixed frame of interest (ex. end-effector frame) is nonlinear and dependant on robot's state $\{\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}\}$

$$\begin{aligned} \dot{\boldsymbol{x}} &= J(\boldsymbol{q})\dot{\boldsymbol{q}} \\ \ddot{\boldsymbol{x}} &= J(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{J}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} \\ \dddot{\boldsymbol{x}} &= J(\boldsymbol{q})\,\dddot{\boldsymbol{q}} + 2\dot{J}(\boldsymbol{q}, \dot{\boldsymbol{q}})\ddot{\boldsymbol{q}} + \ddot{J}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})\dot{\boldsymbol{q}} \end{aligned} \tag{6.2}$$

A certain joint configuration $\boldsymbol{q}$, the joint velocity $\dot{\boldsymbol{q}}$, acceleration $\ddot{\boldsymbol{q}}$ and jerk $\dddot{\boldsymbol{q}}$ are mapped to CS using the Jacobian matrix $J(\boldsymbol{q}) \in \mathbb{R}^{m \times n}$ and its time derivatives $\dot{J}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \in \mathbb{R}^{m \times n}$ and $\ddot{J}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) \in \mathbb{R}^{m \times n}$.

JS kinematic limits (6.1) are expressed in a form of an interval for each of the robot's n joints (degrees of freedom), forming n dimensional hyperrectangles. For any given robot state $\{\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k, \ddot{\boldsymbol{q}}_k\}$, CS kinematic limits can be calculated by projecting these JS n dimensional hyperrectangles (6.1) into the m dimensional CS using the expressions (6.2). The resulting CS limits will have a form of convex polytopes.

As described in Section 2.1.2 and Section 2.1.3, for a certain robot state $\{\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k, \ddot{\boldsymbol{q}}_k\}$ the convex polytopes $\mathcal{P}_v$, $\mathcal{P}_a$ and $\mathcal{P}_j$ of achievable CS velocities $\dot{\boldsymbol{x}}$, accelerations $\ddot{\boldsymbol{x}}$ and jerks $\dddot{\boldsymbol{x}}$ can be expressed as

$$\mathcal{P}_v(\boldsymbol{q}_k) = \{\dot{\boldsymbol{x}} \in \mathbb{R}^m \mid \dot{\boldsymbol{x}} = J(\boldsymbol{q}_k)\dot{\boldsymbol{q}} + \boldsymbol{b}_v, \quad \dot{\boldsymbol{q}} \in [\dot{\boldsymbol{q}}_{min}, \dot{\boldsymbol{q}}_{max}]\} \tag{6.3a}$$

$$\mathcal{P}_a(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k) = \{\ddot{\boldsymbol{x}} \in \mathbb{R}^m \mid \ddot{\boldsymbol{x}} = J(\boldsymbol{q}_k)\ddot{\boldsymbol{q}} + \boldsymbol{b}_a, \quad \ddot{\boldsymbol{q}} \in [\ddot{\boldsymbol{q}}_{min}, \ddot{\boldsymbol{q}}_{max}]\} \tag{6.3b}$$

$$\mathcal{P}_j(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k, \ddot{\boldsymbol{q}}_k) = \{\dddot{\boldsymbol{x}} \in \mathbb{R}^m \mid \dddot{\boldsymbol{x}} = J(\boldsymbol{q}_k)\,\dddot{\boldsymbol{q}} + \boldsymbol{b}_j, \quad \dddot{\boldsymbol{q}} \in [\dddot{\boldsymbol{q}}_{min}, \dddot{\boldsymbol{q}}_{max}]\} \tag{6.3c}$$

where $\boldsymbol{b}_a, \boldsymbol{b}_j \in \mathbb{R}^m$ are the bias acceleration and jerk produced by the effect of current joint velocity $\dot{\boldsymbol{q}}_k$ and acceleration $\ddot{\boldsymbol{q}}_k$, while $\boldsymbol{b}_v \in \mathbb{R}^m$ is a zero vector, added to unify the formulations.

$$\begin{aligned} \boldsymbol{b}_v &= \boldsymbol{0} \\ \boldsymbol{b}_a &= \dot{J}(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k)\dot{\boldsymbol{q}}_k \\ \boldsymbol{b}_j &= 2\dot{J}(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k)\ddot{\boldsymbol{q}}_k + \ddot{J}(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k, \ddot{\boldsymbol{q}}_k)\dot{\boldsymbol{q}}_k \end{aligned} \tag{6.4}$$

Additionally, as the $\boldsymbol{b}_j$ term produced by the second derivative of the Jacobian matrix $\ddot{J}\dot{\boldsymbol{q}}_k$ will produce relatively small effects on final value of $\boldsymbol{b}_j$, it is neglected $\ddot{J}\dot{\boldsymbol{q}}_k \approx 0$. Finally, the achievable CS velocity, acceleration and jerk, given current robot state $\{\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k, \ddot{\boldsymbol{q}}_k\}$ can be expressed as

$$\dot{\boldsymbol{x}} \in \mathcal{P}_v(\boldsymbol{q}_k), \quad \ddot{\boldsymbol{x}} \in \mathcal{P}_a(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k), \quad \dddot{\boldsymbol{x}} \in \mathcal{P}_j(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k, \ddot{\boldsymbol{q}}_k) \tag{6.5}$$

As point-to-point CS paths are effectively one dimensional, instead of using the complete polytope definitions (6.3a - 6.3c), a more efficient solution is to use the polytopes to find the robot's movement capacity in the path direction. Since the path is one dimensional the calculated movement capacity can be expressed in the form or min-max ranges (intervals) for each one of the CS variables. The following section proposes an efficient approach to finding the projection of the CS polytopes of robot's movement capacity in the path direction.

6.3.1 Finding movement capacity in the path direction

Characterising the robot's movement (velocity, acceleration and jerk) capacity in the path direction, requires finding the maximal (and minimal) values of the robot's Cartesian Space (CS) movement variable in the path direction that is still within its movement capacity polytope. In other words, geometrically, this corresponds to finding the intersection of the polytope with the line corresponding to the path direction. Therefore, this section proposes an efficient Linear Program (LP) based approach for characterising this intersection.

For any point-to-point CS path, an m dimensional CS unit vector $\boldsymbol{c}$, pointing in the direction of the path can be found. Then any CS vector $\boldsymbol{y} \in \mathbb{R}^m$ can be projected on the path direction using the scalar product

$$s = \boldsymbol{c}^T \boldsymbol{y} \tag{6.6}$$

where $s \in \mathbb{R}$ is a scalar. Analogously, the projection of the vector $\boldsymbol{y}$ on the path's complementary (orthogonal) space can be expressed using [226, p. 431]

$$\boldsymbol{y}_\perp = \underbrace{(I_{m\times m} - \boldsymbol{c}\boldsymbol{c}^T)}_{N} \boldsymbol{y} \tag{6.7}$$

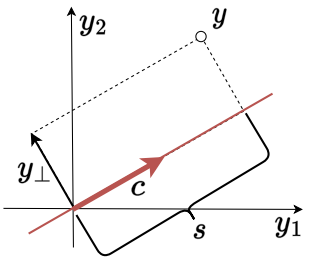

FIGURE 6.4: Figure shows the projection s of the CS variable in the path direction $\boldsymbol{c}$ as well as its components $\boldsymbol{y}_\perp$ normal to the path.

where $\boldsymbol{y}_\perp \in \mathbb{R}^m$ is a m dimensional vector containing the components of the vector $\boldsymbol{y}$ orthogonal to the path. Mathematically, the matrix $N = (I_{m\times m} - \boldsymbol{c}\boldsymbol{c}^T)$ corresponds to the projector to the null-space $\mathcal{K}er\left(\boldsymbol{c}^T\right)$ of the vector $\boldsymbol{c}$. A simple check can then be devised to verify if a certain CS vector $\boldsymbol{y}$ is in the path direction. In order for vector $\boldsymbol{y}$ to be in the path direction, respecting the equation (6.6), it should not have components in the path null-space, or in other words $N\boldsymbol{y} = \boldsymbol{0}$.

Remark. *The expression for null-space projector matrix N, defined in equation (6.7), is valid only if the vector $\boldsymbol{c}$ is a unit vector. If the vector $\boldsymbol{c}$ is not normalised the equivalent expression becomes $N = I_{m\times m} - \frac{\boldsymbol{c}\boldsymbol{c}^T}{\boldsymbol{c}^T\boldsymbol{c}}$ [226, p. 431].*

With the known path direction $\boldsymbol{c}$ and the path normal space projector N, the maximal value of the Cartesian variable $\boldsymbol{y} \in \mathbb{R}^m$, in the path direction $\boldsymbol{c}$, within its range expressed as a convex polytope $\mathcal{P}_y$, can be found by solving the LP [157]

$$\begin{aligned} \max_{\boldsymbol{y}} \ & \boldsymbol{c}^T \boldsymbol{y} \\ s.t. \ & N\boldsymbol{y} = \boldsymbol{0}, \\ & \boldsymbol{y} \in \mathcal{P}_y \end{aligned} \tag{6.8}$$

whereas the minimum can be found by minimising it. Figure 6.5 illustrates geometrically this procedure.

By substituting the generic Cartesian variable $\boldsymbol{y}$ with Cartesian velocity $\dot{\boldsymbol{x}}$, acceleration $\ddot{\boldsymbol{x}}$ and jerk $\dddot{\boldsymbol{x}}$ and its polytope $\mathcal{P}_y$ with their respective polytopic limits (6.5), LP formulation (6.8)

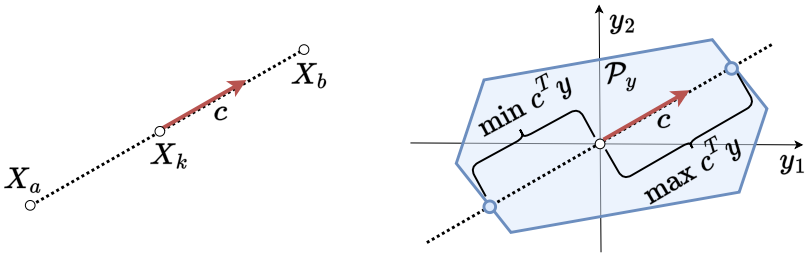

FIGURE 6.5: Figure demonstrating the proposed approach to finding the maximal CS capacity in the path direction. Robot's path is shown on the left and where the path direction vector $\boldsymbol{c}$ is shown in red. Polytope $\mathcal{P}_y$ of the CS capacity is shown in blue on the right. The maximal and minimal capacity in the path direction can then be found by minimising and maximising the LP defined in (6.8).

can be directly used to calculate the limits of the velocity $\dot{s}$, acceleration $\ddot{s}$ and jerk $\dddot{s}$ in the trajectory direction $\boldsymbol{c}$.

Cartesian velocity example When searching for maximal Cartesian velocity $\dot{s}_{max}$, assuming known robot configuration $\boldsymbol{q}_k$ and trajectory direction vector $\boldsymbol{c}$, the Cartesian velocity along the trajectory $\dot{s}$ can be calculated as a projection of the Cartesian velocity $\dot{\boldsymbol{x}}$

$$\dot{s} = \boldsymbol{c}^T \dot{\boldsymbol{x}} = \boldsymbol{c}^T J(\boldsymbol{q}_k)\dot{\boldsymbol{q}} + \boldsymbol{c}^T \boldsymbol{b}_v \tag{6.9}$$

Integrating this relationship into the LP formulation (6.8) yields

$$\begin{aligned} \dot{s}_{max} = \max_{\dot{\boldsymbol{q}}} \; & \boldsymbol{c}^T J(\boldsymbol{q}_k)\dot{\boldsymbol{q}} + \boldsymbol{c}^T \boldsymbol{b}_v \\ s.t. \quad & NJ(\boldsymbol{q}_k)\dot{\boldsymbol{q}} = -N\boldsymbol{b}_v, \\ & \dot{\boldsymbol{q}} \in [\dot{\boldsymbol{q}}_{min}, \dot{\boldsymbol{q}}_{max}] \end{aligned} \tag{6.10}$$

The equivalent LP expressions for finding the maximal acceleration and jerk are obtained by substituting $\dot{\boldsymbol{q}}$ and $\boldsymbol{b}_v$ with $\ddot{\boldsymbol{q}}$, $\boldsymbol{b}_a$ and $\dddot{\boldsymbol{q}}$, $\boldsymbol{b}_j$. The minimal values are found by minimising the same problem. Using the proposed procedure, for any path direction $\boldsymbol{c}$ and a given robot state $\{\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k, \ddot{\boldsymbol{q}}_k\}$, the robot's movement capacities (velocity $\dot{s}$, acceleration $\ddot{s}$ and jerk $\dddot{s}$) in the path direction can be expressed as

$$\begin{aligned} \dot{\boldsymbol{s}} &\in [\dot{\boldsymbol{s}}_{min}(\boldsymbol{q}_k), \;\; \dot{\boldsymbol{s}}_{max}(\boldsymbol{q}_k)] \\ \ddot{\boldsymbol{s}} &\in [\ddot{\boldsymbol{s}}_{min}(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k), \;\; \ddot{\boldsymbol{s}}_{max}(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k)] \\ \dddot{\boldsymbol{s}} &\in [\dddot{\boldsymbol{s}}_{min}(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k, \ddot{\boldsymbol{q}}_k), \;\; \dddot{\boldsymbol{s}}_{max}(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k, \ddot{\boldsymbol{q}}_k)] \end{aligned} \tag{6.11}$$

Therefore, determining the robot's instantaneous movement capacity in the path direction can be obtained very efficiently, by solving a sequence of 6 LP problems and allowing for real-time execution.

It is worth noting that the robot's movement capacity limits, as expressed in (6.11), assume that the robot's velocity, acceleration and jerk limits are independent and do not influence one another, which is not usually the case (ex. if one of the robot's joints reaches its maximal velocity, it will no longer be able to accelerate in that direction). However, these limits are used for Trapezoidal Acceleration Profile (TAP) planning which inherently accounts for this issue and finds the trajectory respecting all the set constraints at the same time.

6.3.2 Integrating task induced Cartesian space limits

In many cases, when planning for Cartesian Space (CS) movement, the task requires limiting maximal CS velocity, acceleration and jerk. For example, when executing trajectories in the human vicinity, it is common practice to limit the robot's CS velocity $\dot{\boldsymbol{x}}$ to minimise the potential impact forces due to the robot's moment of inertia [169] and kinetic energy [227]. On the other hand, when designing trajectories of robotic manipulation of liquids, in order to prevent its sloshing or spilling, translation acceleration $\ddot{\boldsymbol{x}}$ continuity needs to be ensured [228], which in term corresponds to introducing CS jerk $\dddot{\boldsymbol{x}}$ limits.

Therefore, if an additional task specific limit of a CS variable $\boldsymbol{y}$ (velocity, acceleration or jerk) is required that can be expressed as an interval

$$\boldsymbol{y} \in [\boldsymbol{y}_{min},\ \boldsymbol{y}_{max}] \tag{6.12}$$

or more generally in a form of polytope defined by a set of inequality constraints

$$\mathcal{C} = \{\boldsymbol{y} \mid A\boldsymbol{y} \leq \boldsymbol{b}\} \tag{6.13}$$

Then the Linear Program (LP) problem (6.8) can be extended to account for the polytope (6.13)

$$\begin{aligned} \max_{\boldsymbol{y}}\ & c^T \boldsymbol{y} \\ s.t.\ & N\boldsymbol{y} = \boldsymbol{0}, \\ & \boldsymbol{y} \in \mathcal{P}_y \cap \mathcal{C} \end{aligned} \tag{6.14}$$

Cartesian velocity example Following the same example of maximal velocity $\dot{s}_{max}$ along the trajectory from (6.10), if an additional CS limits are imposed by the task

$$\mathcal{C}_v = \{\dot{\boldsymbol{x}} \mid A\dot{\boldsymbol{x}} \leq \boldsymbol{b}\} \tag{6.15}$$

the maximal velocity $\dot{s}_{max}$ along the trajectory that respects both task constraint (6.15) and robot's instantaneous movement capacity (6.3a), can be found by extending the LP formulation (6.10)

$$\begin{aligned} \dot{s}_{max} = \max_{\dot{\boldsymbol{q}}}\ & \boldsymbol{c}^T J(\boldsymbol{q}_k)\dot{\boldsymbol{q}} + \boldsymbol{c}^T \boldsymbol{b}_v \\ s.t.\quad & NJ(\boldsymbol{q}_k)\dot{\boldsymbol{q}} = -N\boldsymbol{b}_v, \\ & AJ(\boldsymbol{q}_k)\dot{\boldsymbol{q}} \leq \boldsymbol{b} - A\boldsymbol{b}_v, \\ & \dot{\boldsymbol{q}} \in [\dot{\boldsymbol{q}}_{min}, \dot{\boldsymbol{q}}_{max}] \end{aligned} \tag{6.16}$$

The lower limit $\dot{s}_{min}$ can be found by minimising the same problem.

6.3.3 Scaling robot's Cartesian space limits

When it comes to planning robot trajectories, it is a common practice to consider only a part (certain percentage) of the specified robot limits. This is partially due to the fact that more dynamic robot movements require higher actuation capacity and leave less margin to account for potential tracking error, in many cases resulting in impaired tracking performance and potentially even raise different safety concerns.

When the robot's Cartesian Space (CS) kinematic limits are assumed constant, the simplest form of scaling can be done by multiplying with scalar factors $\alpha_v, \alpha_a, \alpha_j \in [0, 1]$.

$$\begin{aligned} \dot{\boldsymbol{x}} &\in [\alpha_v \dot{\boldsymbol{x}}_{min},\ \alpha_v \dot{\boldsymbol{x}}_{max}] \\ \ddot{\boldsymbol{x}} &\in [\alpha_a \ddot{\boldsymbol{x}}_{min},\ \alpha_a \ddot{\boldsymbol{x}}_{max}] \\ \dddot{\boldsymbol{x}} &\in [\alpha_j \dddot{\boldsymbol{x}}_{min},\ \alpha_j \dddot{\boldsymbol{x}}_{max}] \end{aligned} \tag{6.17}$$

allowing the robots velocity $\dot{\boldsymbol{x}}$, acceleration $\ddot{\boldsymbol{x}}$ and jerk $\dddot{\boldsymbol{x}}$ not to exceed certain percentage of the specified limits.

However, if the robot's CS kinematic capacity is not considered constant, but a result of the robot's current state $\{\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}\}$, and its Joint Space (JS) kinematic limits (6.1), then the scaling using the scalars $\alpha_v, \alpha_a, \alpha_j \in [0, 1]$ can done in the JS

$$\begin{aligned} \dot{\boldsymbol{q}} &\in [\alpha_v \dot{\boldsymbol{q}}_{min},\ \alpha_v \dot{\boldsymbol{q}}_{max}] \\ \ddot{\boldsymbol{q}} &\in [\alpha_a \ddot{\boldsymbol{q}}_{min},\ \alpha_a \ddot{\boldsymbol{q}}_{max}] \\ \dddot{\boldsymbol{q}} &\in [\alpha_j \dddot{\boldsymbol{q}}_{min}, \text{`}\alpha_j \dddot{\boldsymbol{q}}_{max}] \end{aligned} \tag{6.18}$$

These new modulated JS limits can then be used to calculate the polytopes $\mathcal{P}_v$,$\mathcal{P}_a$ and $\mathcal{P}_j$ using equations (6.3a - 6.3c), and in term scale the robot's CS movement capacity in path direction (6.11).

6.4 Evolving capacity aware Cartesian Space trajectory planning

In order to fully exploit the robot's changing movement capacity while executing the trajectory, this work proposes a real-time re-planning strategy. The proposed trajectory planning method leverages the computational efficiency of Trapezoidal Acceleration Profile (TAP) or S-curve velocity profiles [216, Chapter 9.2.2.2][229, 230], a classical approach to finding the Cartesian Space (CS) time-optimal or minimum-time trajectories [209]. In each step of the trajectory execution, the proposed method evaluates the robot's movement capacity in the path direction, using the efficient method described in Section 6.3.1, and recalculates the new time-optimal trajectory using TAP planning on the remaining path.

Section 6.4.1 brings a brief introduction to the basics of the TAP planning, while Section 6.4.2 provides more details about the real-time re-planning approach.

6.4.1 Trapezoidal acceleration profile basics

Cartesian Space (CS) point-to-point straight-line paths connect two CS poses X_a and X_b (for example expressed as homogeneous matrices in $SE(3)$) of the desired robot's frame (ex. end-effector frame) [216, Chapter 9.2.1]. These paths can be expressed as $\mathcal{P}(s)$, where $s \in [0, d]$ is a scalar position on the path with a length d [231, 232].

When is comes to finding the time-optimal time evolution of the robot's CS pose $X(t)$ following the straight line path between X_a and X_b, one of the most common

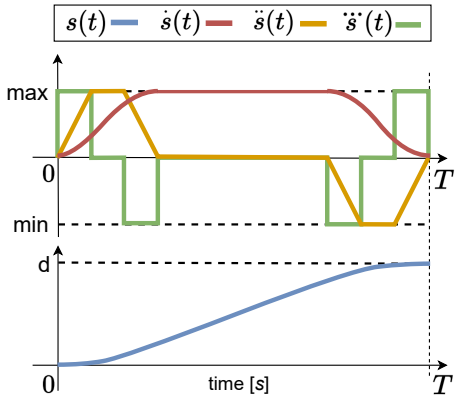

FIGURE 6.6: TAP example, all the initial and final conditions are set to 0.

approaches in the literature are the Trapezoidal Acceleration Profile (TAP) planning techniques [216, Chapter 9.2.2.2]. TAP trajectory planning finds the time-optimal evolution of the path position variable $s(t)$ respecting the limits on all its derivatives

$$\dot{s} \in [\dot{s}_{min}, \dot{s}_{max}], \quad \ddot{s} \in [\ddot{s}_{min}, \ddot{s}_{max}], \quad \dddot{s} \in [\dddot{s}_{min}, \dddot{s}_{max}] \tag{6.19}$$

as well as the starting and end conditions

$$\dot{s}(0) = \dot{s}_0 \quad \dot{s}(T) = \dot{s}_T \quad \ddot{s}(0) = \ddot{s}_0 \quad \ddot{s}(T) = \ddot{s}_T \tag{6.20}$$

Where T is the trajectory duration, $\dot{s}_0$ and $\ddot{s}_0$ represent the velocity and acceleration at the beginning of the path $\mathcal{P}(s)$, while the $\dot{s}_T$ and $\ddot{s}_T$ represent their final values at the end of the trajectory. One example of the TAP profile is shown on Figure 6.6.

When planing for the geometric paths $\mathcal{P}(s)$ in CS, where the CS poses X_a and X_b belong to $SE(3)$, it is common practice to separate the translation $\mathcal{P}_t(s_t)$ and orientation $\mathcal{P}_r(s_r)$ component of the path $\mathcal{P}(s)$[233]. The translation component of the path can then be expressed as

$$\mathcal{P}_t(s_t) = s_t \boldsymbol{u}_t, \quad s_t \in \left[0, ||X_b^t - X_a^t||_2\right] \tag{6.21}$$

where X_a^t and X_b^t are the translation parts of the poses X_a and X_b and $\boldsymbol{u}_t$ is the unit vector pointing from X_a^t to X_b^t. For the orientation, the common approach is to use the axis-angle representation of the rotation, and specify the difference in orientation between X_a and X_b as an angle θ around the axis $\boldsymbol{u}_r$. Then the geometric path corresponding to the orientation $\mathcal{P}_o(s_r)$ can be written as

$$\mathcal{P}_r(s_r) = s_r \boldsymbol{u}_r, \quad s_r \in [0, \theta] \tag{6.22}$$

$\mathcal{P}_r(s_r)$ and $\mathcal{P}_t(s_t)$ represent a relative change in the orientation and translation over the course of the trajectory from the initial pose X_a. The desired CS pose $X(t)$ can be calculated using an homogeneous transformation matrix H, constructed from $\mathcal{P}_r$ and $\mathcal{P}_t$

$$X(t) = X_a H\Big(\mathcal{P}_r\big(s_r(t)\big),\ \mathcal{P}_t\big(s_t(t)\big)\Big) \tag{6.23}$$

and optimal CS velocity and acceleration can be found

$$\dot{\boldsymbol{x}}(t) = \begin{bmatrix} \dot{s}_t(t)\boldsymbol{u}_t \\ \dot{s}_r(t)\boldsymbol{u}_r \end{bmatrix}, \quad \ddot{\boldsymbol{x}}(t) = \begin{bmatrix} \ddot{s}_t(t)\boldsymbol{u}_t \\ \ddot{s}_r(t)\boldsymbol{u}_r \end{bmatrix} \tag{6.24}$$

The translation path $\mathcal{P}_t(s_t)$ and the orientation path $\mathcal{P}_r(s_r)$ direction can be calculated as

$$\boldsymbol{c}_t = [\boldsymbol{u}_t^T, \boldsymbol{0}_{3\times1}]^T, \qquad \boldsymbol{c}_r = [\boldsymbol{0}_{3\times1}, \boldsymbol{u}_r^T]^T \tag{6.25}$$

where $\boldsymbol{u}_t \in \mathbb{R}^3$ is the unit vector pointing from the CS pose X_a^t to X_b^t and $\boldsymbol{u}_r \in \mathbb{R}^3$ represents the axis of the rotation between the poses.

Once both translation and orientation TAP trajectories are found resulting in optimal $s_r(t)$ and $s_t(t)$, to synchronise the two movements, their time duration is matched, making both trajectories last the same time T, the time taken by the longer of the two trajectories $T = \max\{T_r, T_t\}$.

The inherent challenge of TAP planning for the robot's motion in the CS is that its movement capacity limits (6.19) are robot's state dependent, and over the course of the trajectory the robot's movement capacity can change significantly. Therefore, no set of fixed CS limits as defined in (6.19) will be able to exploit the robot's full movement capabilities.

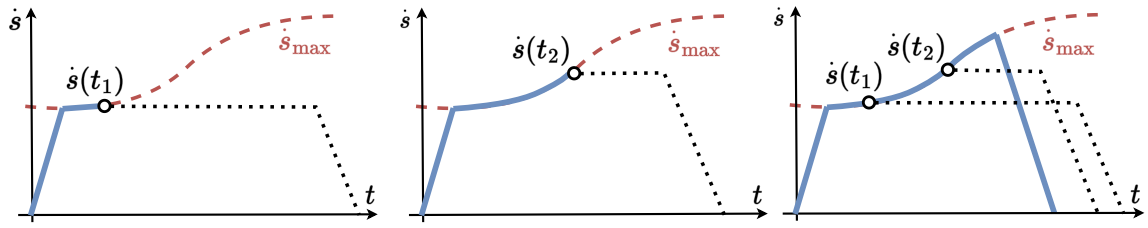

FIGURE 6.7: Velocity curve, updated in the real time with instantaneous maximal values. At each time-step t_k during the trajectory execution (ex. t_1 and t_2) the Trapezoidal Acceleration Profile (TAP) planning calculates the remaining trajectory considering that maximal velocity $\dot{s}_{max}$ constant till the end of the movement (black dotted lines). As the maximal velocity $\dot{s}_{max}$ increases during the trajectory execution (red dashed line), the final trajectory duration is considerably shorter.

6.4.2 Allowing for real-time updates of CS capacity

To address the issue of constantly changing kinematic limits (6.19), during robot trajectory execution, a real-time re-planning strategy is proposed. This approach consists in evaluating the jerk $\dddot{s}$, acceleration $\ddot{s}$ and velocity $\dot{s}$ limits at each step of the trajectory execution, using the proposed method in Section 6.3.1. The updated limits are then considered constant until the end of the movement and used in conjunction with the TAP algorithm to calculate the time-optimal profile of the remaining trajectory. Figure 6.7 illustrates the proposed approach on the example of the velocity $\dot{s}$ curve.

In each time step k, corresponding to the moment in time t_k and robot's state $\{\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k, \ddot{\boldsymbol{q}}_k\}$, the instantaneous limits on the path variables $s_r(t)$ and $s_t(t)$ are calculated using the method described in Section 6.3.1. The new updated limits have a form

$$\begin{aligned} \dot{\boldsymbol{s}}_i &\in [\dot{\boldsymbol{s}}_{i,min}(\boldsymbol{q}_k),\ \dot{\boldsymbol{s}}_{i,max}(\boldsymbol{q}_k)] \\ \ddot{\boldsymbol{s}}_i &\in [\ddot{\boldsymbol{s}}_{i,min}(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k),\ \ddot{\boldsymbol{s}}_{i,max}(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k)] \\ \dddot{\boldsymbol{s}}_i &\in [\dddot{\boldsymbol{s}}_{i,min}(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k, \ddot{\boldsymbol{q}}_k),\ \dddot{\boldsymbol{s}}_{i,max}(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k, \ddot{\boldsymbol{q}}_k)] \end{aligned} \tag{6.26}$$

where i is either the translation t or the orientation r. The translation limits are calculated using the direction vector $\boldsymbol{c}_t$ and the rotation limits are calculated using $\boldsymbol{c}_r$, defined in (6.25).

Then, given the robot's position on the trajectory $s_t(t_k), s_r(t_k)$, the remaining length of the geometric path to the target position can be calculated as $d_k = s_{t,T} - s_t(t_k)$ and the remaining angle of rotation $\theta_k = s_{r,T} - s_r(t_k)$. Then the initial conditions for the new planning execution can be updated

$$\begin{aligned} s_t &\in [0, d_k], \quad \dot{s}_{t,0} = \dot{s}_t(t_k), \quad \ddot{s}_{t,0} = \ddot{s}_t(t_k), \\ s_r &\in [0, \theta_k], \quad \dot{s}_{r,0} = \dot{s}_r(t_k), \quad \ddot{s}_{r,0} = \ddot{s}_r(t_k) \end{aligned} \tag{6.27}$$

Finally, the updated TAP trajectory can then be calculated for the translation $s_t(t)$ and the orientation $s_r(t)$, resulting in an optimal robot trajectory given the updated robot's limits (6.26) and the initial conditions (6.27). The assumption of constant limits (6.26) for the duration of the remaining trajectory is a simplification that does not hold true in practice, as the robot's movement capacity can change significantly over time. However, this issue is addressed through real-time planning, which constantly updates the plan using the latest information on the robot's current ability. The TAP planning algorithm is well-suited for this approach, as it is highly efficient in computing optimal trajectories, allowing for real-time execution.

An example of the generated velocity $\dot{s}$ profile using real-time TAP re-planning, with constantly changing limits $\dot{s}_{max}$, is shown on Figure 6.7.

6.5 Compensating for real-time TAP planning negative effects

The proposed approach adapts to the constant changes of the robot's motion capabilities by planning the new Trapezoidal Acceleration Profile (TAP) trajectory at each time step t_k. For each planning iteration it considers constant robot's capabilities until the end of the trajectory. This assumption is reasonable as only the first step of each planned trajectory is used, in which the calculated robot's capabilities are valid. However, based on the prediction of robot's constant movement capacity, the TAP planning decides the phase of the trajectory: when to stop accelerating and when to start braking. Therefore, in some cases, especially when robot's capacity changes significantly towards the end of the trajectory, TAP planning can produce oscillations and an overshoot due to the switching between the modes of braking and accelerating.

There are two main scenarios that generate unwanted behaviour, both related to the robot's deceleration (braking) capacity. Section 6.5.1 describes the overshoot effect due to the robot's decreasing braking capacity towards the end of the trajectory, while Section 6.5.2 describes the oscillation effect produced by the robot's increasing braking capacity towards the end of the trajectory. The sections propose heuristics for minimising these effects and numerically validate the choice of their parameters.

6.5.1 Decreasing braking capacity: Overshoot effect

When the robot's deceleration capacity decreases towards the end of the trajectory, Figure 6.8, the planned trajectory will have an overshoot. Due to considering robot's capacity constant, robot's braking capacity is overestimated and when the Trapezoidal Acceleration Profile (TAP) planning decides to start braking, at the time step t_k, it is already too late, the robot is not able to stop before it reaches the target position.

This is an inherent challenge of real-time planning and cannot be solved without adding a degree of prediction of the robot's future capabilities. One such approach to mitigate the overshoot is to predict the robot's minimal braking capacity for the remaining Cartesian Space (CS) path $\mathcal{P}(s(t_i))$ until the end of the trajectory $t_i \in [t_k, T]$ and use this value for TAP planning.

$$\ddot{s}_{min} = \max\{\ddot{s}_{min}(t_i)\} \quad t_i \in [t_k,\ T] \tag{6.28}$$

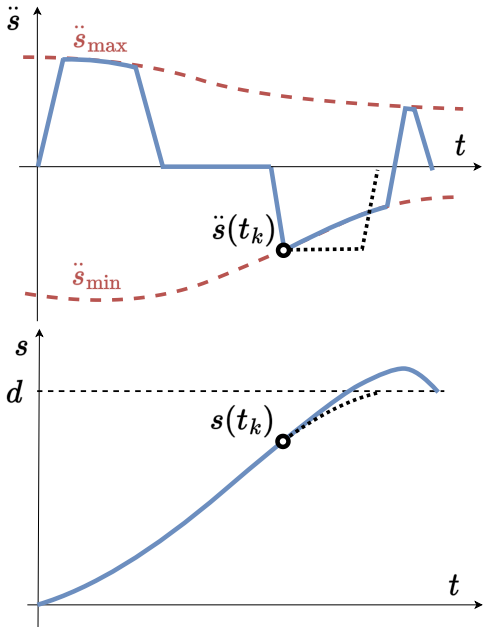

FIGURE 6.8: Figure shows the effect of diminishing braking capacity towards the end of the trajectory, where the end result is an overshoot.

This approach in many cases results in more conservative trajectories as the robot's braking capacity is underestimated, but at the same time it guarantees the overshoot removal. As the robot's braking capacity depends on its joint state $\{\boldsymbol{q}, \dot{\boldsymbol{q}}\}$, finding the minimal braking capacity (6.28) requires either knowing the exact robot's Joint Space (JS) path until the end of the trajectory, which is by definition not known, or

finding all the possible joint configurations the robot can be in on the remaining path, which is very long and not practical for most real-time applications.

In this work a sampling based approximation of (6.28) is proposed based on sampling the remaining the CS path (from the current pose X_k to the final pose X_b) into N posses $X_i \in SE(3)$. The most probable predictions JS configurations $\boldsymbol{q}_i$ at each of the poses X_i is proposed to be found as the robot's inverse kinematics solution the closest to the current pose $\boldsymbol{q}_k$. Finally, the prediction of (6.28) can be found as the minimal braking capacity among all the sampled ones $\ddot{s}_{i,min}$

$$\ddot{s}_{min} = \max\{\ddot{s}_{0,min}, \ \ldots, \ \ddot{s}_{N,min}\} \tag{6.29}$$

Experimental validation of compensation parameters

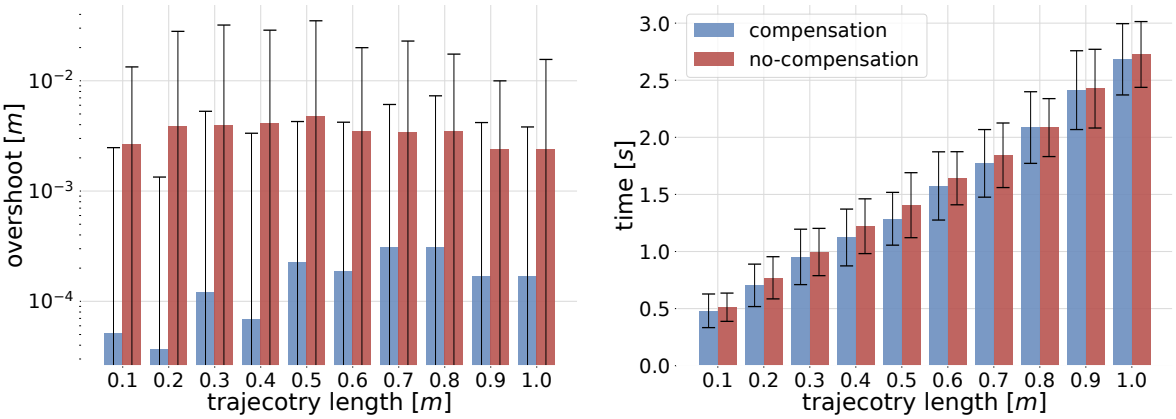

FIGURE 6.9: Plots showing the comparison of the average overshoot (left) and average execution time (right) of the proposed method with and without overshoot compensation. For each length ranging from $d = 0.1$ to $1m$, means and variances are calculated over 100 random trajectories.

In the experiments the remaining trajectory is sampled with $N = 2$ points, corresponding to the current position X_k in the time step t_k and the final position at the end of the trajectory X_T. Joint configuration $\boldsymbol{q}_T$ at X_T is found as the inverse kinematics solution the closest to the current configuration $\boldsymbol{q}_k$ and the joint velocity $\dot{\boldsymbol{q}}_T$ and acceleration $\ddot{\boldsymbol{q}}_T$ are considered to be $\dot{\boldsymbol{q}}_T, \ddot{\boldsymbol{q}}_T = \mathbf{0}$, as the robot will come to a stop at the target X_T. Finally the braking capacity used for TAP planning in the step t_k is the minimum of the two

$$\ddot{s}_{min} = \max\{\ddot{s}_{k,min}, \ \ddot{s}_{T,min}\} \tag{6.30}$$

Figure 6.9 presents a comparison between the proposed method's performance with and without overshoot compensation. To evaluate the methods, 1000 random translation trajectories (random fixed orientation) were generated in the robot's workspace, ranging in length from $d = 10cm$ to $d = 1m$. The experiments were conducted in simulation using a Franka Emika Panda robot, the implementation details are described in Section 6.6. The overshoot and execution time were recorded for each trajectory. The results indicate that the proposed method with overshoot compensation significantly reduces the expected overshoot compared to the method without compensation, going from $3mm$ average overshoot to $0.1mm$. Moreover, the compensation strategy does not have a negative impact on the trajectory execution time. On the contrary, it slightly reduces the average execution time, as shown in the figure.

The numerical analysis shows that even by taking in consideration only the final braking capacity of the robot $\ddot{s}_{T,min}$ the effects of the overshoot can be significantly reduced. Furthermore,

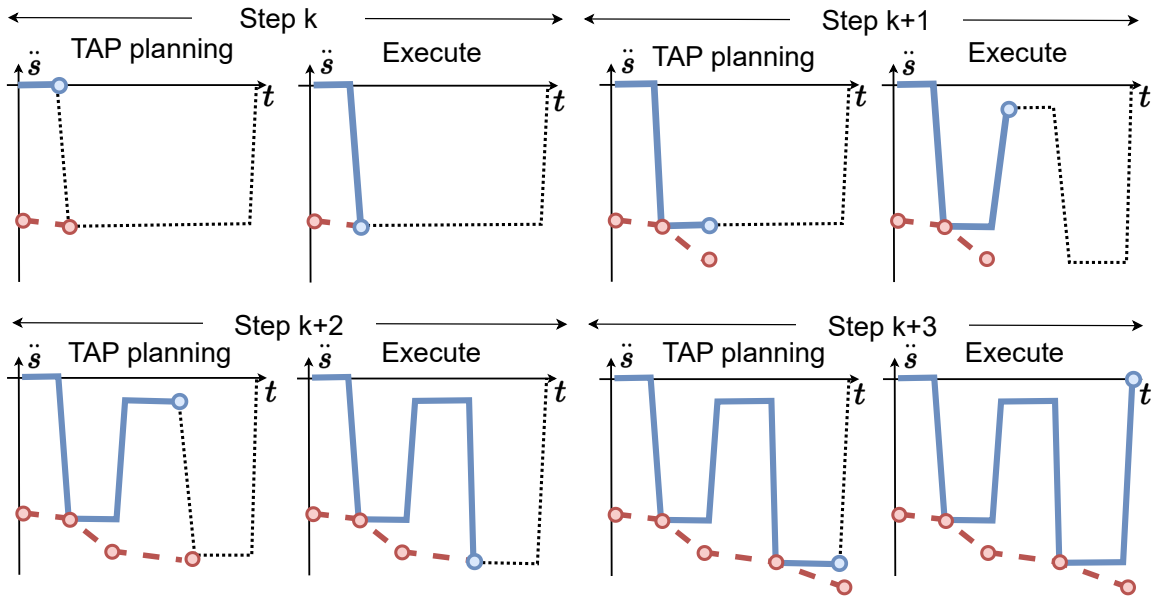

FIGURE 6.11: The figure shows braking stage of the TAP planning trajectory with the effect of augmenting braking capacity towards the end of the trajectory. The executed acceleration $\ddot{s}$ is shown in blue, the planned one is shown in black dotted lines and the maximal braking limit is shown in red dashed lines. Due to, in some cases, inverse proportional coupling between the robot's velocity $\dot{\boldsymbol{q}}$ and the braking capacity, the more the robot brakes, the more its velocity $\dot{\boldsymbol{q}}$ decreases and more the braking capacity increases. This effect produces an oscillatory behaviour, as shown in the sequence of figures. The robot starts braking in time step k. Because it was braking, its braking capacity increased in step $k+1$. With the increased braking capacity, TAP planning decides to decrease braking in step $n+1$ as it can brake stronger later.

this strategy can be implemented very efficiently since the robot's final braking capacity $\ddot{s}_{T,min}$ has to be evaluated only once per trajectory execution.

6.5.2 Increasing braking capacity: Oscillations effect

When the robot's breaking capacity increases along the trajectory, scenario shown on Figure 6.10, the proposed method can result in oscillatory behaviour. Due to considering robot's capacity constant, robot's braking capacity is underestimated and the decision of the Trapezoidal Acceleration Profile (TAP) planning to start braking, at the time step t_k, comes too soon. In the next step t_{k+1} the robot's breaking capacity increases which makes TAP planning take the decision to stop braking. The repetitions of this sequence create the oscillations of the planed acceleration profile and produces jerky trajectories.

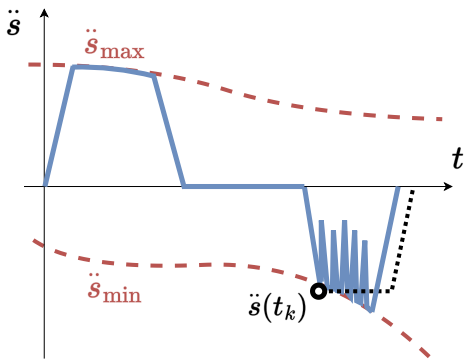

FIGURE 6.10: Figure shows the effect of increasing braking capacity towards the end of the trajectory, where the end result are oscillations.

The effect of these oscillations is greatly amplified as the robot's braking capacity $\ddot{s}_{min}$, in some cases, is inverse proportional to the robot's current joint velocity $\dot{\boldsymbol{q}}_k$ through the bias term $\boldsymbol{b}_a = \dot{J}(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k)\dot{\boldsymbol{q}}_k$. Lowering the joint velocity $\dot{\boldsymbol{q}}_k$, lowers the bias $\boldsymbol{b}_a$ and increases the braking capacity $\ddot{s}_{min}$. This effect creates a closed loop behaviour, where the more the robot brakes, the

more its braking capacity increases. Figure 6.11 illustrates few time steps of such oscillatory behaviour.

In order to smooth the acceleration profile and reduce the effect of coupling introduced by the bias $\boldsymbol{b}_a$, a simple strategy is proposed which consists in down-sampling the TAP planning. Instead of planning the TAP trajectory in each time step t_k, the trajectory is planned with the time step Δt_p, while linearly interpolating the trajectory between the planning steps $t \in [t_k, t_k + \Delta t_p]$.

$$s(t) = s_k + \frac{s_{k+1} - s_k}{\Delta t_p}(t - t_k), \quad t \in [t_k, t_k + \Delta t_p] \tag{6.31}$$

The same linear interpolation can be applied to the velocity $\dot{s}$ and acceleration $\ddot{s}$

$$\begin{aligned} \dot{s}(t) &= \dot{s}_k + \frac{\dot{s}_{k+1} - \dot{s}_k}{\Delta t_p}(t - t_k), \\ \ddot{s}(t) &= \dot{s}_k + \frac{\ddot{s}_{k+1} - \ddot{s}_k}{\Delta t_p}(t - t_k) \end{aligned} \tag{6.32}$$

where s_k,$\dot{s}_k$ are $\ddot{s}_k$ are path position, velocity and acceleration in the current step t_k and s_{k+1},$\dot{s}_{k+1}$ are $\ddot{s}_{k+1}$ are the their optimal values in the next planning step $t_{k+1} = t_k + \Delta t_p$ calculated by the TAP planning.

Between the TAP planning steps, the robot's movement capacity is considered constant and in that way the high-frequency oscillations induced by the bias term $\boldsymbol{b}_a$ are filtered. Furthermore, the longer the time between the planning Δt_p, the more the effects of the coupling are reduced. On the other hand, the robot's movement capacity is considered constant between the planning steps Δt_p. Since the robot's movement capacity can decrease between planning steps, the planned trajectory can potentially overestimate the robot's capacity in this period. This overestimation can therefore increase the robot's tracking error due to its inability to follow the planned trajectory. Therefore, when implementing this oscillation compensation strategy, a choice of the planning step Δt_p requires a trade-off to be made between the filtering the high-frequency oscillations and the allowable level of tracking error.

Experimental validation of compensation parameters

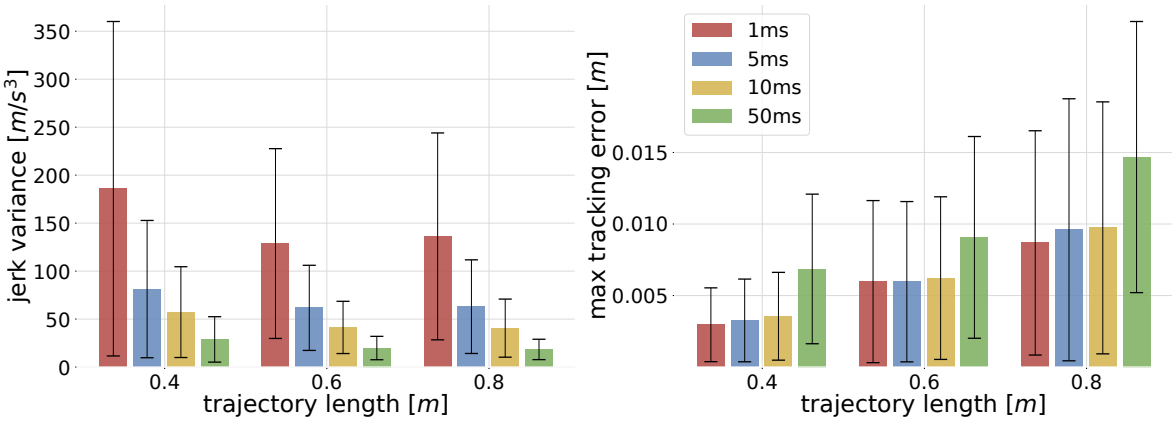

FIGURE 6.12: Plots showing the comparison of the jerk variance (left) and tracking error (right) of the proposed method with and without oscillation compensation, for trajectory lengths $d = 0.4$, 0.6 and $0.8m$. Means and variances are calculated over 100 random trajectories. Four down-sampling times are compared $\Delta t_p = 1, 5, 10$ and 50ms.

In order to find the optimal down-sampling time Δt_p an empirical study is conducted for 100 random trajectories with lengths $d = 40cm$, $60cm$ and $80cm$ in the robots workspace. The experiments are conducted in simulation using a Franka Emika Panda robot, the implementation details are described in Section 6.6. Four down sampling times are compared $\Delta t_p = 1ms$, $5ms$, $10ms$ and $50ms$. For each executed trajectory the maximal deviation from the path and the Cartesian Space (CS) jerk variance is evaluated.

Figure 6.12 demonstrates an inverse proportionality between the down sampling time and the jerk variance. Additionally, it reveals that as the planning time step Δt_p increases, there is a corresponding increase in the deviation from the desired path. Therefore, the choice of the appropriate planning time step Δt_p implies making a trade-off between the two. In the case of this work, the planning step $\Delta t_p = 10ms$ is chosen, resulting in a significant decrease in the jerk variance while having relatively low impact on the tracking error.

6.6 Experimental setup

All the experiments, both in simulation and in real world, are conducted on a Franka Emika Panda robot. The robot's kinematic limits in Joint Space (JS) and the Cartesian Space (CS) are obtained from Franka Emika's official datasheet [225]. These values are publicly available[1] and are listed in Table 6.1 and Table 6.2.

Limits	q_0	q_1	q_2	q_3	q_4	q_5	q_6
$\dot{\boldsymbol{q}}_{max}$ $[rad/s]$	2.175	2.175	2.175	2.175	2.61	2.61	2.61
$\ddot{\boldsymbol{q}}_{max}$ $[rad/s^2]$	15	7.5	10	12.5	15	20	20
$\dddot{\boldsymbol{q}}_{max}$ $[rad/s^3]$	7500	3750	5000	6250	7500	10000	10000

TABLE 6.1: Franka Emika Panda robot JS kinematic limits[1]. The lower limits are symmetric to the upper ones.

Limits	Translation	Orientation
$\dot{\boldsymbol{x}}_{max}$	1.7 m/s	2.5 rad/s
$\ddot{\boldsymbol{x}}_{max}$	13 m/s^2	25 rad/s^2
$\dddot{\boldsymbol{x}}_{max}$	6500 m/s^3	12500 rad/s^3

TABLE 6.2: Franka Emika Panda robot CS kinematic limits[1]. The lower limits are symmetric to the upper ones.

6.6.1 Robot control architecture

The schematic diagram of the proposed approach within the robot control paradigm is shown on Figure 6.13. Given the robot's current state $\{\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k, \ddot{\boldsymbol{q}}_k\}$ in step k, the approach first determines the robot's current CS movement capacity in the path direction using the approach described in Section 6.3.1, resulting in velocity $\dot{s}_{max}$, acceleration $\ddot{s}_{max}$ and jerk $\dddot{s}_{max}$ limits. Using the updated limits, the robot's current CS state X_k,$\dot{\boldsymbol{x}}_k$,$\ddot{\boldsymbol{x}}_k$ and the target state X_t,$\dot{\boldsymbol{x}}_t$,$\ddot{\boldsymbol{x}}_t$, the proposed method calculates the new optimal trajectory using TAP planning, as described in Section 6.4.2. Once the optimal TAP trajectory is found, desired states X_d,$\dot{\boldsymbol{x}}_d$,$\ddot{\boldsymbol{x}}_d$ are sent to the inverse velocity kinematics layer of the control architecture.

[1]Full datasheet available at: https://frankaemika.github.io/docs/control_parameters.html

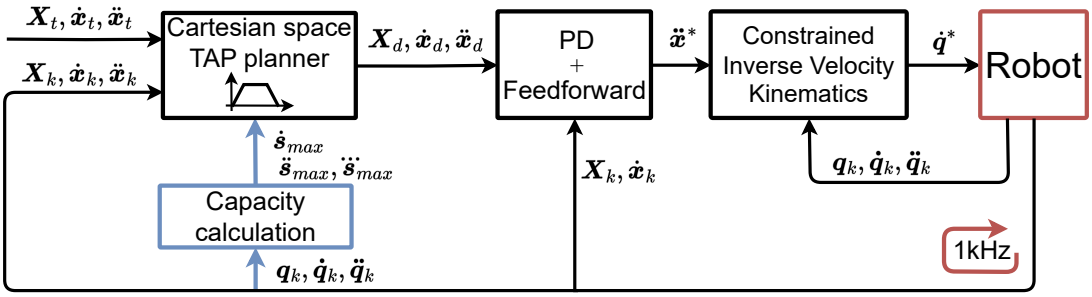

FIGURE 6.13: Proposed method schematic overview in the context of the robot control, elements in blue present the extension of a standard Cartesian Space (CS) based Trapezoidal Acceleration Profile (TAP) planning.

In this work, the robot control strategy for real-time CS trajectory following is formulated as a Quadratic Program (QP) and solved in each control loop.

$$\begin{aligned} \ddot{\boldsymbol{q}}_{opt} = \arg\min_{\ddot{\boldsymbol{q}}} \quad & \overbrace{||\ddot{\boldsymbol{x}}^* - J(\boldsymbol{q}_k)\ddot{\boldsymbol{q}} - \dot{J}(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k)\dot{\boldsymbol{q}}_k||^2}^{\text{trajectory tracking}} + \overbrace{\omega_r||\ddot{\boldsymbol{q}}_r - \ddot{\boldsymbol{q}}||^2}^{\text{regularisation}} \\ \text{s.t.} \quad & \ddot{\boldsymbol{q}} \in [\ddot{\boldsymbol{q}}_{ub}(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k, \ddot{\boldsymbol{q}}_k),\ \ddot{\boldsymbol{q}}_{lb}(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k, \ddot{\boldsymbol{q}}_k)] \end{aligned} \tag{6.33}$$

where the $\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k \in \mathbb{R}^n$ is robot's current state, $\ddot{\boldsymbol{q}}_k \in \mathbb{R}^n$ is the robot's current acceleration, $J(\boldsymbol{q}_k) \in \mathbb{R}^{m \times n}$ and $\dot{J}(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k) \in \mathbb{R}^{m \times n}$ are the robot's state dependent jacobian matrix and its time derivative. The bounds of each of joint accelerations $\ddot{q}_{i,ub}$, $\ddot{q}_{i,lb}$ are calculated in a way to guarantee that the joint jerk $\dddot{\boldsymbol{q}}$, acceleration $\ddot{\boldsymbol{q}}$, velocity $\dot{\boldsymbol{q}}$ and position $\boldsymbol{q}$ in the horizon δt respect their limits.

$$\ddot{q}_{i,ub} = \min \Big\{ \underbrace{\ddot{q}_{i,k} + \dddot{q}_{i,max}\delta t}_{\text{jerk}},\ \underbrace{\ddot{q}_{i,max}}_{\text{acceleration}},\ \underbrace{\frac{1}{\delta t}(\dot{q}_{i,max} - \dot{q}_{i,k})}_{\text{velocity}},\ \underbrace{\frac{2}{\delta t^2}(q_{i,max} - q_{i,k} - \dot{q}_{i,k}\delta t)}_{\text{position}} \Big\} \tag{6.34}$$

where the horizon δt has to be chosen long enough to ensure constraints compatibility without leading to conservative behaviour [234]. Equation (6.34) shows the upper bound expression, the lower bound calculation is equivalent, and is obtained by substituting *min* for *max* and maximising instead of minimising. The horizon δt chosen for the experiments is 15ms.

The optimisation problem (6.33) consists in two tasks: trajectory tracking task and a secondary (regularisation) task.

Trajectory tracking task The trajectory tracking task is accomplished by following the desired CS acceleration $\ddot{\boldsymbol{x}}^*$. the control law formulated as a PD controller with a feed-forward term through the desired Cartesian acceleration $\ddot{\boldsymbol{x}}^*$

$$\begin{aligned} \ddot{\boldsymbol{x}}^* &= K_p\boldsymbol{e} + K_d(\dot{\boldsymbol{x}}_d - \dot{\boldsymbol{x}}_k) + \ddot{\boldsymbol{x}}_d \\ \boldsymbol{e} &= \mathrm{Ad}(X_k)\log(X_k^{-1}X_d) \end{aligned} \tag{6.35}$$

X_d,$\dot{\boldsymbol{x}}_d$,$\ddot{\boldsymbol{x}}_d$ are the desired CS pose, velocity and acceleration in the next step, while X_k and $\dot{\boldsymbol{x}}_k$ are the measured CS pose and velocity in current step k. $K_p, K_d \in \mathbb{R}^m$ are diagonal matrices containing the proportional and derivative gains. Vector $\boldsymbol{e}$ is the CS pose error expressed in the world frame.

The robot is controlled using the joint velocity commands which are calculated using an Euler backward numerical integration

$$\dot{\boldsymbol{q}}^*_{k+1} = \dot{\boldsymbol{q}}^*_k + \ddot{\boldsymbol{q}}_{opt}\Delta t \tag{6.36}$$

In the experiments, the PD controller gains used are

$$K_p = \mathrm{diag}\big([170.0, 170.0, 170.0, 100.0, 100.0, 100.0]\big)$$

and

$$K_d = \mathrm{diag}\big([50.0, 50.0, 50.0, 30.0, 30.0, 30.0]\big)$$

Regularisation task The robot's redundant degrees of freedom are used to dampen the movement in the trajectory null-space and keep the robot away from its joint limits. The regularisation task is expressed through the joint acceleration $\ddot{\boldsymbol{q}}_r$

$$\ddot{\boldsymbol{q}}_r = k_{rp}(\boldsymbol{q}_r - \boldsymbol{q}) - k_{rd}\dot{\boldsymbol{q}} \tag{6.37}$$

where k_{rp} and k_{rd} are scalar gains and $\boldsymbol{q}_r$ is the initial robot pose at the center of all the joint ranges.

$$\boldsymbol{q}_r = \frac{\boldsymbol{q}_{max} + \boldsymbol{q}_{min}}{2}$$

The secondary task gains used are $k_{rp} = 5s^{-2}$ and $k_{rd} = 2\sqrt{k_{rd}}s^{-1}$, while secondary task weight is $\omega_r = 1e^{-5}$.

6.6.2 Software implementation details

The simulation experiments used for the comparative studies are implemented in Python using open-source implementations of TOPP-RA [211] and Trapezoidal Acceleration Profile (TAP) planning implementation within `ruckig` library [230]. The code used for the simulation experiments is open-source and can be found on GitLab[2].

For the mock-up experiment, the real-time execution of the proposed method is implemented in C++. The TAP planning is implemented using the open-source library `ruckig` [230], while the efficient Linear Program (LP) solver used for real-time Cartesian Space (CS) limit calculation is GLPK [235]. All the software components are integrated using Robot Operating System (ROS).

Additionally, both simulation and real world experiments used robot rigid body dynamics simulation implemented using the open-source library `pinocchio` [166].

6.7 Comparative study

A simulation based comparative study is conducted to evaluate the performance of the proposed method against well known time-optimal planning algorithms. The proposed algorithm is first compared to a state of the art time-optimal Joint Space (JS) planning method called TOPP-RA [211]. Furthermore the proposed algorithm is compared against the Trapezoidal Acceleration Profile (TAP) planning with fixed Cartesian Space (CS) limits provided by the manufacturer.

6.7.1 Benchmarking against time-optimal Joint space planning

In order to evaluate the performance of the trajectories found and executed by the proposed Cartesian Space (CS) planning approach, it is compared against a state-of-the art time-optimal

[2] https://gitlab.inria.fr/auctus-team/people/antunskuric/papers/catp

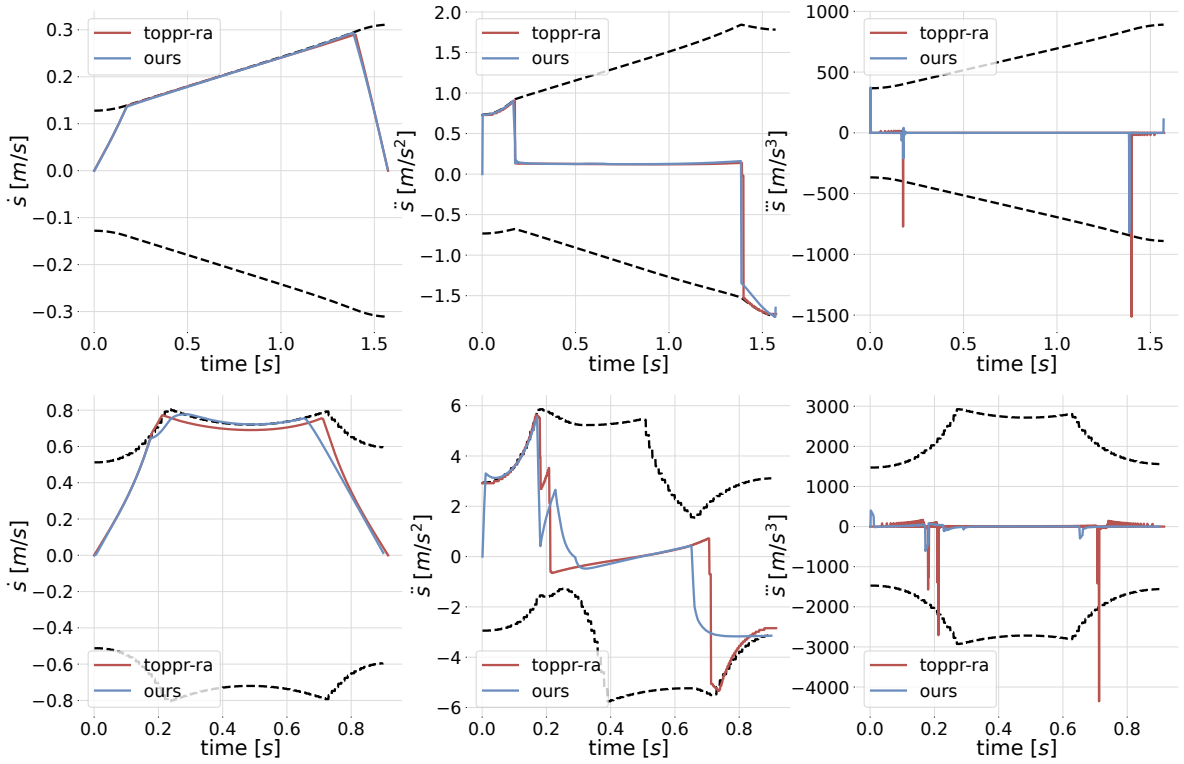

FIGURE 6.14: The plots show the comparison of the path velocity, acceleration and jerk, generated by the proposed approach (blue) and TOPP-RA(red), for two different trajectories (up and down). The dashed lines show the robot's movement limits in the trajectory direction calculated using the proposed method, described in Section 6.3.1.

Joint Space (JS) planning algorithm TOPP-RA [211]. TOPP-RA takes in consideration the robot's JS velocity and acceleration limits and plans for the minimum time JS trajectories. Both algorithms are run on a set of 1000 random straight line trajectories (with random fixed orientation), ranging in length from $d = 10cm$ to $1m$. The goal of this experiment is to compare the generated trajectory profiles of TOPP-RA and the proposed approach as well as their execution times, in order to asses the tome optimally of the proposed real-time planning approach.

TOPP-RA's limitation however is that it cannot deal with CS trajectories directly and it requires an additional step of finding the JS path corresponding to the CS path. For TOPP-RA, the full JS path is determined by sampling the CS path $\mathcal{P}(s)$ into the set of way-points $X_{w,i}$ and finding the inverse kinematic solution $\boldsymbol{q}_{w,i}$ for each one of them. To ensure the JS path continuity, the inverse kinematics solution $\boldsymbol{q}_{w,n+1}$ at the way-point $X_{w,n+1}$ is taken as the one closest to the joint configuration $\boldsymbol{q}_{w,n}$ in the previous way-point $X_{w,n}$. The distance between the way-points used in these experiments is $5cm$.

Both approaches are tested using 50% of the robot's JS velocity, acceleration and jerk capacity $\alpha_v = \alpha_a = \alpha_j = 0.5$.

Results

The comparison of the planned velocity, acceleration and jerk profiles for two trajectories are shown on Figure 6.14. It can be seen that both methods find similar time profiles for all the path variables, indicating the effectiveness of the proposed approach. Moreover, even though TOPP-RA plans entirely in JS its CS trajectory respects the CS velocity and acceleration

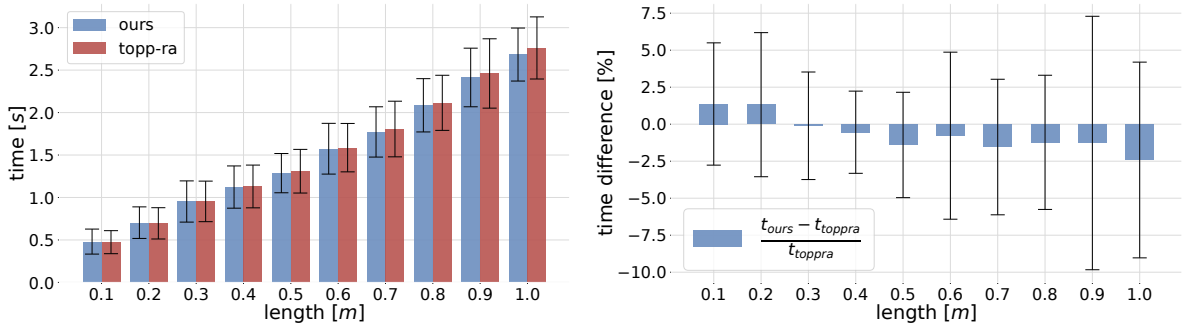

FIGURE 6.15: Plots showing the comparison of the trajectory execution time of the proposed method with TOPP-RA, for trajectory lengths ranging from $d = 0.1$ to $1m$. Left graph shows the comparison of the execution time, while the graph on the right shows the difference in execution time in a form of percentage. Means and variances are calculated over 100 random trajectories.

limits calculated by the proposed method, demonstrating the accuracy of limit calculations. Additionally, TOPP-RA does not limit the jerk which can be seen on the jerk plots in Figure 6.14.

Additionally, the influence of the overshoot compensation strategy can be observed in the second (bottom) trajectory. The proposed approach starts braking earlier and does not use the full braking capacity of the robot. The resulting trajectory does not have an overshoot, even though the robot's braking capacity decreases significantly towards the end of the trajectory. This demonstrates the effectiveness of the proposed strategy in reducing overshoot while not significantly increasing the execution time.

The trajectory execution time comparison is presented in Figure 6.15. The figure shows that the proposed method has a comparable execution time for all tested trajectory lengths. Interestingly, for lengths over $40cm$, our approach is even faster than TOPP-RA. This can be attributed to the fact that our method takes a different JS path than TOPP-RA, which in some cases might be more optimal than the one calculated in advance. This effect is more present for longer trajectories, where taking the JS path the closest to the initial joint configuration might not be the most optimal criteria.

6.7.2 Benchmarking against time-optimal Cartesian space planning

The proposed adaptive approach is further compared to a standard Trapezoidal Acceleration Profile (TAP) trajectory planning using fixed Cartesian Space (CS) kinematic limits. The CS limits for the Franka Emika panda robot are taken from the standard datasheet [225], as given in Table 6.2.

The approaches are compared over 100 random robot straight line trajectories (with random fixed orientation), in the robot's workspace, with a fixed length of $d = 50cm$. The performance of the two approaches is compared for different scaling levels α, starting at 10% (α=0.1) of robot's capacity and going to 90% (α=0.9). The scaling strategy is chosen to be equal for velocity, acceleration and jerk $\alpha = \alpha_v = \alpha_a = \alpha_j$.

The implementation of the TAP trajectory generator for both approaches is done using the open-source library `ruckig` [230]. The robot control strategy implemented for trajectory following is the same for both approaches as detailed in the Section 6.6.1.

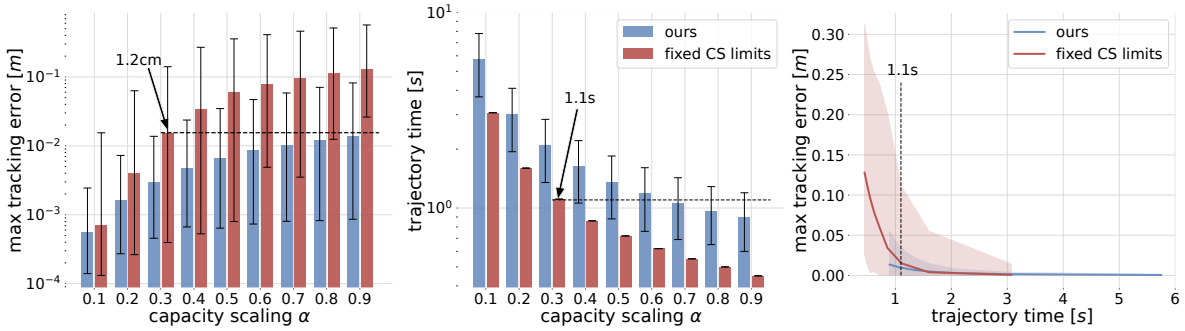

FIGURE 6.16: Result of the benchmarking experiment comparing the fixed CS planning approach (red) to the proposed method (blue). The graph on the left shows comparison of the max tracking error with respect to the ratio of the capacity α used while the graph in the middle shows the trajectory execution time comparison. The graph on the right unites the two graphs on the left in order to show the relation of the trajectory execution time against the max tracking error. Means and variances are calculated on 100 random robot trajectories.

Results

In Figure 6.16, the results of a comparative study between the proposed method and the TAP planning method with fixed CS limits are presented. Both methods show a linear relationship between the tracking error and the ratio of capacity used (α), however the proposed approach has significantly lower mean errors and variances for same values α. This can be attributed, in part, to the overestimated CS limits (Table 6.2) provided by the manufacturer for the Panda robot, see Figure 6.1. Due to this overestimation, the planned TAP trajectories, depending on the path direction in robot's workspace, can become infeasible and induce large tracking errors. This is particularly apparent when using larger percentage of the robot's movement capacity. Figure 6.16 (on the left) shows that the average tracking error for $\alpha \geq 0.7$ is larger than $10cm$ (while the length of the trajectory is $50cm$).

Figure 6.16 further shows that using 30% of the manufacturer's CS limits (i.e., α=0.3) results in a tracking error of around 1.2cm, which is higher than any of the mean errors achieved by the proposed method for all tested values of α. While for α=0.3, the fixed CS planning has the execution time of 1.1 seconds, the proposed method has lower execution time for all $\alpha \geq 0.7$. These results therefore confirm that when it comes to planning for highly dynamical trajectories, taking in consideration robot's true movement capacity results in faster and more precise movements.

In addition to the comparative study, a simple comparative experiment of the two methods is conducted using a Franka Emika Panda robot. In this experiment the operator attempts to find the fastest possible trajectories using the two methods, while having the tracking error under 1cm. The optimal value of the capacity ratio α is determined experimentally, by augmenting it in small increments and finding the maximal value that satisfies the tracking error condition. A short video of the experiment is publicly available[3].

Video

[3]Video: https://youtu.be/KBo0ZHihi3I

6.8 Mock-up experiment: Collaborative waste sorting

Waste recycling is an important tool for addressing the ecological issue of accumulating humanities waste. One of its crucial parts is the waste sorting procedure, where the recyclable materials are extracted from the regular waste and prepared for the recycling process. Robotics and computer vision technologies have a great potential to improve the waste processing efficiency and increase its volume [236]. However, waste sorting is a highly dynamic and unstructured environment, presenting many challenges to potential robotised solutions. To be viable, waste sorting robots need to be able to operate at high speeds, where they use pick-and-place or pick-and-toss [237] techniques to sort the waste in different material groups. Therefore, one of the key challenges for building such systems is producing efficient and reactive movement generation techniques.

Several robotic solutions have been proposed in the literature to address waste sorting tasks [238], such as *Zenrobotics Fast picker*[4] or *SELMA*[5] . However, these solutions are based on expensive industrial robots and are only viable for large scale recycling facilities.

In this work, we propose a mock-up interactive (collaborative) scenario for waste sorting that leverages the proposed real-time trajectory planning method in order to create fast and adaptable robot movement.

In the experiment, a human operator is introducing different waste items at the sorting workstation, where the robot is placed. The operator can introduce the waste items in any time and order, as well as modify their position and orientation on the table. The operator can do the same with the sorting buckets, the bins in which the sorted items are placed. The robot's job is to pick all the waste items and place them in the appropriate sorting buckets as fast as possible.

Waste items (two cans and two cartons), as well as the buckets, are tracked in real-time using a motion capture system *OptiTrack*. In order to efficiently and robustly sort the waste, object manipulation procedure is divided in 6 phases.

1. Position the gripper 5cm above the object with the appropriate orientation
2. Lower the gripper to the object
3. Close the gripper
4. Rise the object 10cm
5. Transport the object to the appropriate bucket
6. Release the object

The robot control architecture, as described in Section 6.6.1, and the proposed real-time planning approach are implemented in programming language C++, using Robot Operating System (ROS) framework, and run in real-time at the frequency of 1kHz. For this experiment, 60% of robot's capacity is used (α=0.6) in order to keep the tracking error under 1cm, as shown on Figure 6.16. The experiment is available in a form of accompanying video[6].

Video

Figure 6.17 shows the time evolution of one run of the experiment. The plot shows the evolution of the translation s_t and orientation s_r in time as well as their respective velocities $\dot{s}_t$, $\dot{s}_r$. The

[4]Zenrobotics Fast picker: https://zenrobotics.com/de/
[5]SELMA: https://www.opteknik.se
[6]Video: https://www.youtube.com/watch?v=BHGipnKNOfA

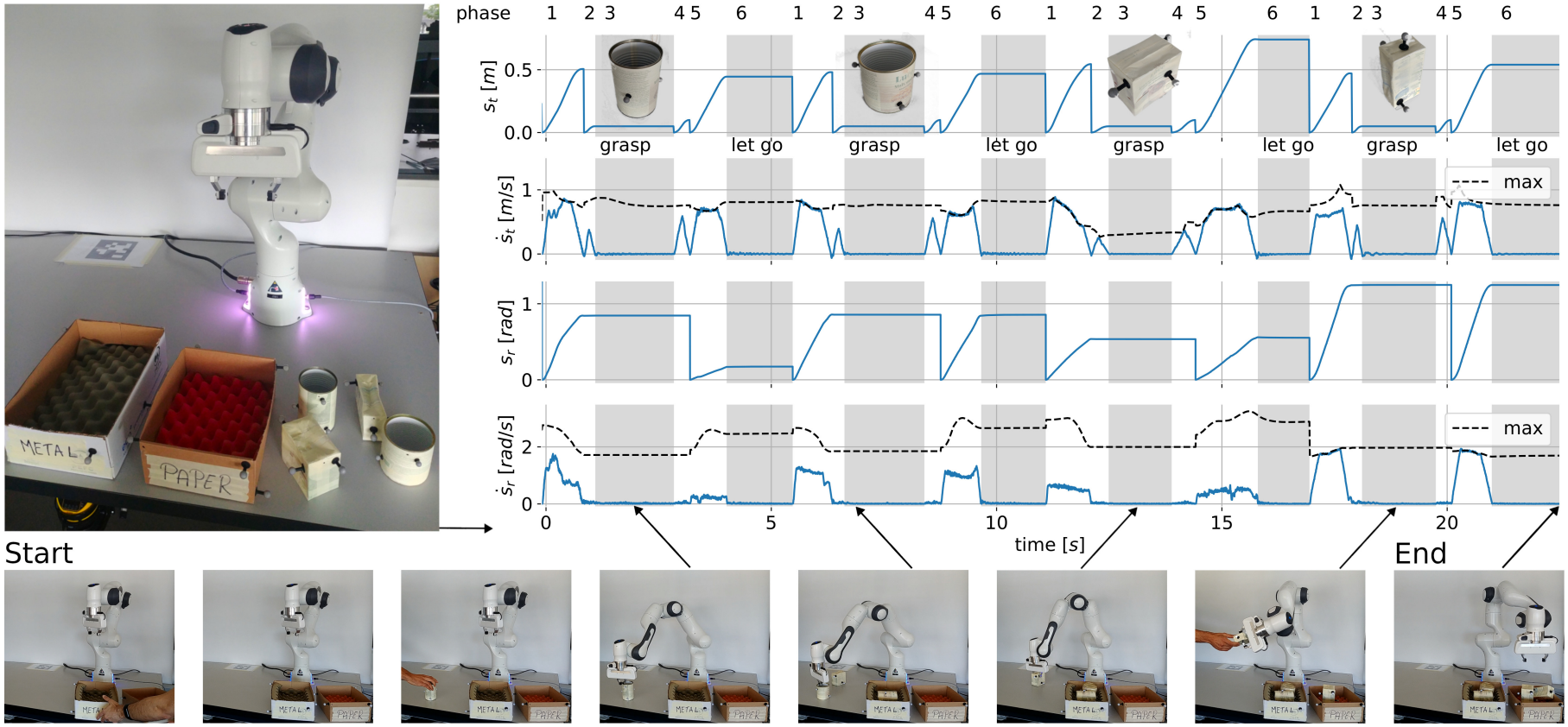

FIGURE 6.17: Figure on the left shows the experimental setup of the mock-up waste sorting experiment. The robot used in the experiments is a Franka Emika Panda robot. The experiment uses six objects: two cans, two cartons and two sorting bins, all tracked in real time using motion capture system Optitrack. The images on the bottom show several moments of the experiment, while the plot (up right) shows the time evolution of the real-time executed trajectories. The plot shows the position s_t and orientation s_r path evolution in time as well as translation $\dot{s}_t$ and rotation $\dot{s}_r$ velocity on the path and their calculated maximal values (dotted lines). In the experiment, the operator first brings the sorting bins to the robot workstation, then introduces the waste objects to the robot with unknown positions and orientations. Robot plans and executes in real-time the trajectories necessary in order to place, as fast as possible, the objects into the appropriate sorting bin.

grey areas show the moments in time where the gripper is closing or opening in order to grasp or let go of an object. Several moments during the experiment run are shown on the images around the plot.

The experiment starts with the human operator bringing the sorting bins (metal and paper) and placing them in the robot's workspace. Then, the operator introduces the objects (sometimes multiple at a time) on the workstation table. The robot plans and executes the necessary trajectories in real-time, in order to place the objects in appropriate sorting bins. Objects' and bins' positions and orientations are not known in advance and can change in real-time.

The velocity $\dot{s}_t$, $\dot{s}_r$ plots show that the proposed approach is able to follow robot's changing capacity and produce motions with maximal possible reachable velocities. It can also be seen that for different trajectories executed, either translation velocity $\dot{s}_t$ or rotation velocity $\dot{s}_r$ is maximised. The reason why they are not both exploited is because the translation and orientation is synchronised in order for the robot to reach both target position and orientation at the same time.

The adaptability of the proposed approach is further demonstrated as the final waste item is handed over to the robot, where the robot grasps the object from the operator's hand.

6.9 Discussion

The proposed online trajectory re-planning approach has several benefits over both classic time-optimal approaches in Joint Space (JS) and reactive approaches in Cartesian Space (CS). It is capable of efficiently exploiting robot's true CS movement capacity, while being reactive at the same time, allowing for planning the efficient trajectories on the fly. In each step of the trajectory execution, the proposed approach efficiently calculates the robot's instantaneous CS movement capacity, using the method described in Section 6.3.1. Then, the robot's CS capacity is considered constant until the end of the trajectory and used to recalculate the time-optimal trajectory for the remaining path using the Trapezoidal Acceleration Profile (TAP) planning. As the approach re-plans in each trajectory execution step, the assumption of constant limits is reasonable, since only the first step of each planned trajectory is ever executed for which the calculated limits are valid.

With this in mind, a parallel between the Model Predictive Control (MPC) [202] and the proposed approach can be made. MPC predicts the future states of the system, given its current states and the available system model and searches to find the optimal action to be executed in the current step given a certain optimality criteria. The proposed approach can then be seen as a simplified special case of the MPC approach, where the robot's model is entirely integrated within the robot's movement capacity limits, which are state dependent and calculated in each time-step. Then the real-time TAP re-planning makes the prediction robot's behaviour until the end of the trajectory and provides the optimal current action to be made in order to execute the desired path in minimum time. The proposed approach lacks the flexibility of the MPC approaches, such are adding additional criteria apart from minimum-time, choosing the prediction time horizon length, where the proposed approach always plans to the end of the trajectory, or considering more complex robot model for in the prediction horizon. However, due to the inherent complexity of the robot's model and the trajectory planning in general, making the proposed simplifications and trading-off the flexibility of the MPC approach, comes with the increase in the computational efficiency and enables the proposed method to run in real-time. The proposed approach has several limitations though.

The main limitation of the proposed approach is its assumption that the CS path can be represented using using only one variable $s(t)$ (or two $s_t(t), s_r(t)$ in case of both translation and

orientation). This assumption enables transforming polytope based robot's capacity metrics (6.5) to the interval ranges of path variables (6.11). As shown in Section 6.3.1, these ranges can be efficiently calculated in real-time and used with the standard trajectory planning algorithms such as TAP. The consequence of such assumption is that the planned trajectory can guarantee respecting the calculated limits only in the path direction, which is reasonable for straight line trajectories. However, if the path is not a straight line, or if it is a straight line but the target position suddenly changes during the trajectory execution, the proposed approach will not be able to guarantee respecting the robot's limits in the directions orthogonal to the path. In order to overcome this effect, trajectory planning algorithms, able to integrate the polytope representation of path constraints (6.5) are required. Therefore, a promising future direction is adapting the planning method for the family of planning techniques based on Quadratic Program (QP) optimisation, such as the MPC approach, which allow for integrating polytope shaped limits, since they can be expressed in a form of linear constrains. The thesis of Nicolas Torres Alberto, a member of AUCTUS team, focuses on this topic. Their work proposes an efficient linear formulation of the MPC in CS, capable of integrating the polytope formulation of robot's state dependent physical abilities, expressed as (6.5) described in Section 6.3.

Different limitation of the proposed approach, that could be resolved using the same set of tools, is the assumption that the translation and orientation paths independent. However, the limits on velocity, acceleration and jerk of both paths are not independent, as both translation and rotation are generated by the same robot actuators with the limits (6.1). The true limits on the path variables s_t and s_r would have a form of a polytope. A way to avoid this issue could be to define the path in Lie space, where the rotation and translation could be represented in one path variable. However, in that case, the robot would no longer move in straight lines in CS. In order to minimise the coupling effect between the rotation and translation, in the context of this work, translation limits are calculated while considering constant rotation velocity, while rotation limits are calculated considering constant translation velocity.

Another limitation of the proposed approach is the assumption of the constant robot's capacity for every CS planning iteration. As described in Section 6.5, this can lead to certain negative effects such as an overshoot or oscillations. In order to overcome this issue, instead of considering only instantaneous robot's capacity for each trajectory generation execution, it would be necessary to predict the robot's kinematic capacity along the remaining trajectory. This is a challenging research topic which results might be applied not just in TAP planning techniques but also to the optimal control methods such as MPC.

Finally, the robot's JS kinematic limits (6.1) are considered constant in time, which is generally not the case. These limits will depend on the robot's actuation limits $\boldsymbol{\tau}$ and different dynamical and gravitational effects acting on the robot, as well of the robot's joint state $\{\boldsymbol{q}, \dot{\boldsymbol{q}}\}$. For highly dynamical robot's movements, the available actuation capacity of the robot might reduce certain of the limits (6.1). Therefore, the integration of the robot's actuation limits $\boldsymbol{\tau}$ could make the proposed approach more robust and it is a promising direction for future research.

6.10 Conclusion

This chapter aims to address one of the key challenges when it comes to trajectory planning in Cartesian Space (CS): accounting for robot's changing movement capacity while executing the trajectory. To tackle this challenge, the chapter proposes an efficient trajectory planning method, capable of adapting to the robot's changing capacities, by evaluating them in real-time and updating the planned trajectory to account for their changes. This chapter showcases the potential of using real-time evaluation of robot's movement capacity (physical ability to generate movement: velocity, acceleration, etc.) for creating time-efficient and reactive trajectory

planning strategies in CS.

The proposed strategy has two main components: an efficient approach to evaluating robot's CS movement ability and an efficient trajectory planning strategy, both capable of running in real-time. Section 6.3 describes the proposed approach for evaluating robot's changing CS movement capacity, leveraging efficient tools from polytope algebra allowing for real-time execution. As described in Section 6.4, the trajectory planning strategy developed in this work is based on time-optimal Trapezoidal Acceleration Profile (TAP) planning, due to its high computational efficiency which enables real-time execution as well. Therefore, in each step of the trajectory execution, the proposed method first efficiently evaluates robot's instantaneous CS movement capacity and then uses it to recalculate the new time-optimal trajectory on the remaining path using the TAP planning. By recalculating the updated time-optimal trajectory in each step of the trajectory execution, the proposed method is able to adapt to the real-time changes in robot's movement capacity, as well as to the changing task constraints and to the potential changes in the trajectory, induced by the events in the environment. The adaptability of the proposed method makes it a robust and flexible tool, particularly interesting for human-robot collaborative scenarios, where responsiveness to real-world events is crucial, as well as the efficient use of robot's capacities, as collaborative robots' often have very limited performance abilities.

To evaluate the performance of the trajectories calculated by the proposed method, it is compared against a state-of-the-art time-optimal Joint Space (JS) planning method called TOPP-RA [211]. The results of the comparison show that the proposed method has a comparable trajectory execution time as TOPP-RA even though it is planning in real-time as opposed in advance optimisation done by TOPP-RA. Furthermore, the proposed method is compared against a standard CS time-optimal TAP planning approach, considering constant CS limits given by the manufacturer. The results show than the proposed method exploits better robot's movement capacity and at the same time has lower tracking error. The comparative study is describe in Section 6.7.

To showcase a potential application of the proposed method, a mock-up experiment is conducted in the context of human-robot collaborative waste sorting. In the experiment, the human operator introduces different waste items on the collaborative workstation with unknown position and orientation in space and at any point in time. The proposed trajectory planning method is then used to generate the time-efficient robot's motions on the fly. The experiment shows that the proposed method enables the robot to execute the pick-and-place motions, placing the waste items in the appropriate sorting bins, as soon as the waste items were introduced and as fast as its movement's abilities allow it. Therefore, the experiment further demonstrates the practical potential of the proposed method in the human-robot collaboration scenarios, showcasing the reactiveness of the proposed approach, as well as its ability to efficiently use robot's movement capacity.

The next chapter, Chapter 7, presents the publicly available open-source software Python package `pycapacity`. The package is developed in the context of this thesis and provides the efficient implementation of algorithms for evaluating polytope and ellipsoid based physical abilities of humans and robots.

Chapter 7

`pycapacity`: An efficient task-space capacity calculation Python package for robotics and biomechanics

Preceding chapters demonstrate the potential of real-time evaluation of robot's and human's physical abilities for creating more flexible human-robot collaboration. Chapter 4 shows that the real-time evaluation of robot's and human's physical ability polytopes can be a valuable tool for creating adaptable robot control strategies for human-robot physical collaboration scenarios, able to adapt to their changing physical abilities. Chapter 5 discusses the potential of polytopes as a visual feedback tools for improving human operators' situational awareness, by providing them with real-time insight into the robot's current state and its changing abilities. Finally, Chapter 6 shows how real-time evaluation of robot's movement capacity enables creating more flexible trajectory planning strategies able to adapt to its changing physical abilities on the fly.

However, all the described approaches heavily rely on the efficient evaluation of different physical ability polytopes, requiring their real-time their execution. As described more in detail in Chapter 3, the computation complexity of polytope evaluation depends highly on its formulation and makes choosing the appropriate algorithm a crucial challenge when it comes to building such applications.

Therefore, this chapter presents the software package called `pycapacity`, which aims to provide a set of efficient tools for evaluating task-space physical ability metrics for humans and robots, based on polytopes and ellipsoids. The package implements several state-of-the-art algorithms for polytope evaluation, including VEPOLI^2 and ICHM developed in the context of this thesis and introduced in Section 3.4 and Section 3.5 respectively. In that way, it bringing many of the physical ability polytopes to the few milliseconds evaluation time, making it possible to use them in online and interactive applications.

Furthermore, `pycapacity` is implemented as a open-source Python package with a goal to be an easy-to-use framework that can be easily integrated with standard robotics and biomechanics libraries. The package can be easily interfaced with standard libraries for robotic manipulator rigid body simulation such as `roboticstoolbox` [201] or `pinocchio` [166], as well as human musculoskeletal model biomechanics software `opensim` [176] and `biorbd` [159]. The package can also be used with the Robot Operating System (ROS) [167].

The package additionally implements a set of visualisation tools for polytopes and ellipsoids intended for fast prototyping and quick and interactive visualisation.

Section 7.1 brings more detailed motivation behind the development of this package. Section 7.2 then brings a brief introduction in the differences between ellipsoids and polytopes and their evaluation within the package. Section 7.3.1 and Section 7.3.2 list the implemented physical

ability ellipsoids and polytopes. Section 7.4 lists the implemented polytope transformation algorithms, followed by Section 7.5 which brings the performance analysis of the implemented polytope algorithms for different physical ability polytopes. Finally, Section 7.6 introduces the brief overview of the software implementation of the package and an example code.

7.1 Motivation

There are many different metrics available in the literature that might be used to characterise different physical abilities of humans and robots: force capacity, velocity capacity, acceleration capacity, accuracy, stiffness etc. Most of these metrics can be represented by two families of geometric shapes: ellipsoids [22] and polytopes [35]. These metrics are traditionally important tools for off-line analysis purposes (workspace design, human motion and ergonomics analysis) and recently, they have shown a great potential to be used for interactive online applications, to be integrated in robot control strategies or as a visual feedback to the operator.

Ellipsoid metrics are often used for evaluating the manipulability of the robot's end-effector. The manipulability ellipsoid is a geometric shape that represents the robot's ability to move within the task-space. Due to their computational efficiency and intuitive visualisation, they have been used in many different applications, such as robot control, workspace design, robot design, etc. Therefore, there are several open-source packages that implement the manipulability ellipsoid evaluation and visualisation, such as `MMC` [239], `manipulability_metrics` [240], `Manipulability` [241][242]. However, most of these packages are limited to the evaluation of the manipulability ellipsoid, representing the velocity capacity, and they do not provide tools for evaluating other ellipsoid metrics, such as force capacity, acceleration capacity, etc. Additionally these software packages are often developed for the use with a specific robotics library, such as `roboticstoolbox` [201] or Robot Operating System (ROS) [167], and they are not trivial to integrate with other libraries.

Even though different efficient tools for evaluating ellipsoids are widely available in the literature and open-source community, the tools for evaluating polytopes are still relatively scarce. The main reason for this is that the polytopes are in general more complex to evaluate and manipulate than ellipsoids. However, the polytopes are much more accurate representation of the true limits. Additionally, polytopes are easy to visualize, as they are essentially triangulated meshes, and they can be easily integrated in the robot control strategies, as they can be expressed as a set of linear constraints.

The evaluation of polytopes is often a computationally expensive task, as their resolution requires using different vertex and facet enumeration algorithms [71]. Therefore, their computation time is often the limiting factor for the use of polytopes in real world applications, especially when it comes to their online use. Furthermore, even though there are several open-source projects that implement polytope evaluation algorithms, such as `pypoman` [243], Multi-Parametric Toolbox 3 (`MPT3`) [244] or `cddlib` [245][246], they are often very generic and not easy to use with standard physical ability polytopes. On the other hand, more specific polytope resolution software solutions, such as `constrained_manipulability` package [33][34] or `pygradientpolytope` [247], are often very specific to their applications, they lack the documentation and flexibility to be extended to new metrics and integrated with other libraries.

Therefore, this chapter presents the `pycapacity` Python package in an effort to provide a set of tools specifically tailored for evaluating task-space physical ability metrics for humans and robots, based on polytopes and ellipsoids. This package groups a set of efficient algorithms for their evaluation in an easy to use framework that can be easily integrated with standard robotics and biomechanics libraries. Furthermore, the package implements several state of

the art algorithms for polytope evaluation that bring many of the polytope metrics to the few milliseconds evaluation time, making it possible to use them in online and interactive applications.

In the context of this thesis, `pycapacity` package has been used for several real-time applications. In Section 4.2 this package is used for real-time control of collaborative carrying using two Franka Emika Panda robots. Section 4.3 uses it to evaluate operator's carrying capacity online and implement a Assist-As-Needed (AAN) control strategy for collaborative carrying task involving a human operator and a robot. The package has also been used to calculate the approximation of the robot's reachable space using convex polytope, described in Section 5.1, allowing for online execution and interactive visualisation.

Finally, in the preliminary work from Laisné *et al.* [182], the package is used in the biomechanics context, with the aim to perform the advanced calibration of human musculoskeletal models to human subjects.

7.2 Ellipsoids and polytopes as physical ability metrics

In robotics, task-space physical ability metrics establish the relationship between different limits of robot's actuators (joint positions, velocities, torques, etc.), their kinematics and dynamics, and the achievable sets of different task related physical quantities, such as achievable positions, velocities, forces and similar. Similar metrics can be established for humans as well, by leveraging their musculoskeletal models. Where the humans in addition to the joint limits (joint positions and velocities) have additional limits due to using their muscles as actuators (contraction forces and velocities).

When it comes to characterizing these achievable sets, the two most common approaches are using ellipsoids and polytopes. Ellipsoids are often used to represent the robot's velocity capacity, so called manipulability, while polytopes are mostly used to represent the robot's force capacity. However, both ellipsoids and polytopes can be used to represent any of the task-space physical ability.

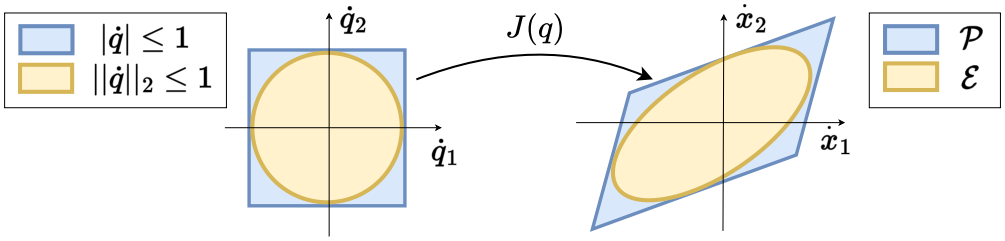

FIGURE 7.1: An example manipulability polytope and ellipsoid geometry for a planar $m = 2$ robot with $n = 2$. The difference between the Joint Space (JS) limits for ellipsoid described with $||\dot{\boldsymbol{q}}||_2 \leq 1$ (orange) and the range limits $-\mathbf{1} \leq \dot{\boldsymbol{q}} \leq \mathbf{1}$ (blue) is shown on the right. The difference in obtained achievable task-space velocity $\dot{x}$ polytope P (blue) and ellipsoid E (orange) is shown on the right plot. The plots show that both in joint and task-space the ellipsoid metric is an underestimation of the true robot's capacity.

To compare the ellipsoid and polytope metrics, the example of the manipulability ellipsoid and manipulability polytope can be used.

The manipulability ellipsoid, proposed by Yoshikawa [22], is defined as the set of all achievable task-space velocities $\dot{\boldsymbol{x}}$ for a given robot configuration q and joint velocity limits $-\mathbf{1} \leq \dot{\boldsymbol{q}} \leq \mathbf{1}$,

and it can be expressed as

$$\mathcal{E} = \{\dot{\boldsymbol{x}} \mid \dot{\boldsymbol{x}} = J(q)\dot{\boldsymbol{q}}, \quad ||\dot{\boldsymbol{q}}||_2 \leq \mathbf{1}\} \tag{7.1}$$

The equivalent polytope representation of the manipulability ellipsoid is the manipulability polytope, which is defined as the set of all achievable task-space velocities $\dot{\boldsymbol{x}}$ for a given robot configuration q and joint velocity limits $-1 \leq \dot{\boldsymbol{q}} \leq 1$, and it can be expressed as

$$\mathcal{P} = \{\dot{\boldsymbol{x}} \mid \dot{\boldsymbol{x}} = J(q)\dot{\boldsymbol{q}}, \quad -\mathbf{1} \leq \dot{\boldsymbol{q}} \leq \mathbf{1}\} \tag{7.2}$$

Figure 7.1 illustrates the difference between the manipulability ellipsoid and polytope for a planar robot with two joints. The manipulability ellipsoid is an underestimation of the true robot's capacity, as it considers that the robot's velocity limits have the shape of a sphere, while in reality the robot's velocity limits $-\mathbf{1} \leq \dot{\boldsymbol{q}} \leq \mathbf{1}$ define a cube. The manipulability polytope is a more accurate representation of the robot's capacity, as it considers the true shape of the robot's velocity limits.

More generally, polytope based representations of different physical abilities present the exact solution both for robots and for human musculoskeletal models, while ellipsoids present an approximation. Figure 7.2 shows the difference between the force ellipsoid and polytope [35] for one configuration of the Franka Emika Panda robot.

Ellipsoids, however, are much more present in the literature, as their computation is much faster than the computation of polytopes.

7.2.1 Evaluating ellipsoids

Evaluating ellipsoids is a computationally efficient task, as it can be done using the Singular Value Decomposition (SVD) [22]. Ellipsoids can be fully defined using their principal axis and principal axis lengths. Once they are known, the ellipsoid can be easily visualised and used for further analysis.

`pycapacity` provides tools for evaluating several common ellipsoid metrics for robots and humans, such as velocity (manipulability), force and acceleration, and it provides a set of tools for their easy visualisation implemented in the module `pycapacity.visual`. All the ellipsoid manipulation is implemented within the generic object `Ellipsoid`, available as a part of `pycapacity.objects` module

7.2.2 Evaluating polytopes

As described more in detail in Chapter 3, evaluating polytopes consists in finding either the minimal set of their vertices, $\mathcal{V}$-representation, or the minimal set of the half-planes defining their faces, $\mathcal{H}$-representation. The $\mathcal{V}$-representation is often used for visualisation purposes, while the $\mathcal{H}$-representation is often integrated in different optimization problems, as it can be represented as a set of linear inequalities.

However, finding the $\mathcal{V}$-representation or the $\mathcal{H}$-representation of a polytope is a computationally expensive task, relying on different vertex and facet enumeration algorithms [71]. The computational complexity of these algorithms depends on the polytope formulation, the dimensionality of the input (number of robot's joints or human muscles) and output spaces (1D, 2D, 3D or 6D Cartesian space) and the complexity of the polytope geometry (number of vertices and faces).

Therefore, polytope evaluation is often a bottleneck in the computation of different physical ability metrics, especially for human musculoskeletal models, which have a large number of degrees of freedom and a large number of muscles. Furthermore, due to the inherent complexity of the polytope evaluation algorithms, finding the appropriate algorithm for a given polytope formulation and dimension ability of the input and output spaces is not a trivial task.

`pycapcity` package aims to provide a selection of algorithms for polytope evaluation, capable of evaluating common physical ability polytopes in an easy to use and efficient way. These algorithms are implemented in Python and can be used as standalone tools as well. Additionally, `pycapacity` implements several common polytope manipulation operations such as:

- Transforming polytope from $\mathcal{H}$ to $\mathcal{V}$-representation
- Transforming polytope from $\mathcal{V}$ to $\mathcal{H}$-representation
- Triangulating the polytope's vertices (sometimes called face or $\mathcal{F}$-representation)
- Creating polytope form the Convex-Hull of a point cloud
- Minkowski sum of polytopes, as described in Appendix A
- Intersection of polytopes, as described in Appendix A
- Chebyshev ball

All these operations are implemented within the generic class `Polytope` and can be used to perform additional operations on any physical ability polytope of humans and robots. Moreover, this class and all the above operations can be used as a standalone library as well for the use-cases outside of the scope of physical ability polytopes. The `Polytope` class and its functionalities can be accessed through the module `pycapacity.objects`. Finally, the package provides tools for easy visualisation the 2D and 3D polytopes implemented in the module `pycapacity.visual`.

7.3 Implemented physical capacity metrics

The package implements different physical ability metrics for robotic manipulators and humans based on musculoskeletal models.

7.3.1 Robotic manipulators metrics

For robotic manipulators the package implement several velocity, force and acceleration capacity calculation functions based on ellipsoids and polytopes. All the metrics are implemented within `pycapacity.robot` module:

Ellipsoids

- Velocity (manipulability) ellipsoid, as described by Yoshikawa [22]

$$\mathcal{E}_v = \{\dot{\boldsymbol{x}} \mid \dot{\boldsymbol{x}} = J\dot{\boldsymbol{q}},\ ||W^{-1}\dot{\boldsymbol{q}}|| \leq 1\}, \qquad W = \text{diag}(\dot{\boldsymbol{q}}_{max}) \tag{7.3}$$

- Acceleration (dynamic manipulability) ellipsoid, as described by Chiacchio [36]

$$\mathcal{E}_a = \{\ddot{\boldsymbol{x}} \mid \ddot{\boldsymbol{x}} = JM^{-1}\boldsymbol{\tau},\ ||W^{-1}\boldsymbol{\tau}|| \leq 1\}, \qquad W = \text{diag}(\boldsymbol{\tau}_{max}) \tag{7.4}$$

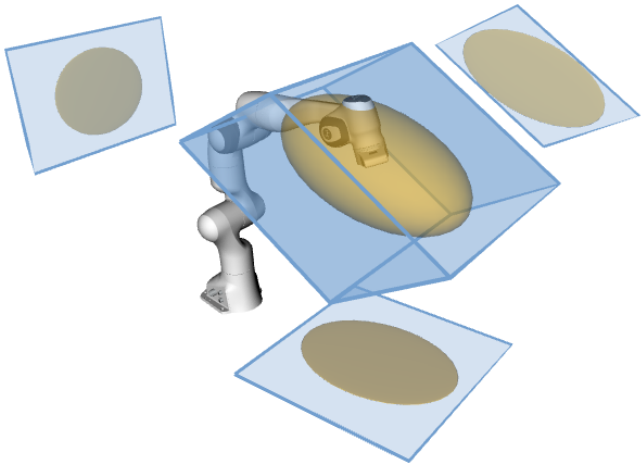

FIGURE 7.2: 2D and 3D force polytopes and their ellipsoid counterparts for a 7 degrees of freedom (DOF) Franka Emika Panda robot. Both polytopes and ellipsoids are calculated separately for the 3D and for each of the 2D reduced task-space cases. Both polytopes and ellipsoids take in consideration the true joint torque limits provided by the manufacturer. The underestimation of the true force capabilities of the robot by ellipsoids appears clearly.

- Force ellipsoid, as described by Chiacchio *et al.* [248]

$$\mathcal{E}_f = \{\boldsymbol{f} \mid J^T\boldsymbol{f} = \boldsymbol{\tau},\ ||W^{-1}\boldsymbol{\tau}|| \leq 1\}, \qquad W = \text{diag}(\boldsymbol{\tau}_{max}) \tag{7.5}$$

In the above definitions, J is the robot Jacobian matrix, M is the inertia matrix, f is the vector of Cartesian Space (CS) forces, $\dot{x}$ and $\ddot{x}$ are vectors for CS velocities and accelerations, q is the vector of Joint Space (JS) positions, $\dot{q}$ is the vector of the JS velocities and τ is the vector of JS torques. Matrix W is a scaling matrix that normalises the JS limits.

Polytopes

- Velocity polytope, described in Section 2.1.2

$$\mathcal{P}_v = \{\dot{\boldsymbol{x}} \mid \dot{\boldsymbol{x}} = J\dot{\boldsymbol{q}},\ \dot{\boldsymbol{q}}_{min} \leq \dot{\boldsymbol{q}} \leq \dot{\boldsymbol{q}}_{max}\} \tag{7.6}$$

- Acceleration polytope, described in Section 2.1.6

$$\mathcal{P}_a = \{\ddot{\boldsymbol{x}} \mid \ddot{\boldsymbol{x}} = JM^{-1}\boldsymbol{\tau},\ \boldsymbol{\tau}_{min} \leq \boldsymbol{\tau} \leq \boldsymbol{\tau}_{max}\} \tag{7.7}$$

- Force polytope, described in Section 2.1.5

$$\mathcal{P}_f = \{\boldsymbol{f} \mid J^T\boldsymbol{f} = \boldsymbol{\tau},\ \boldsymbol{\tau}_{min} \leq \boldsymbol{\tau} \leq \boldsymbol{\tau}_{max}\} \tag{7.8}$$

- Minkowski sum and intersection of force polytopes

$$\mathcal{P}_\cap = P_{f1} \cap P_{f1} \qquad P_\oplus = P_{f1} \oplus P_{f1} \tag{7.9}$$

- Robot's reachable space approximation in the desired horizon of interest Δt_h using the convex polytope formulation, described in Section 5.1

$$\begin{aligned} P_x = \{\Delta \boldsymbol{x} \mid \Delta \boldsymbol{x} = JM^{-1}\boldsymbol{\tau}\frac{\Delta t_h^2}{2}, \\ \boldsymbol{\tau}_{min} \leq \boldsymbol{\tau} \leq \boldsymbol{\tau}_{max}, \\ \dot{\boldsymbol{q}}_{min} \leq M^{-1}\boldsymbol{\tau}\Delta t_h \leq \dot{\boldsymbol{q}}_{max}, \\ q_{min} \leq M^{-1}\boldsymbol{\tau}\frac{\Delta t_h^2}{2} \leq q_{max}\} \end{aligned} \tag{7.10}$$

7.3.2 Human musculoskeletal model metrics

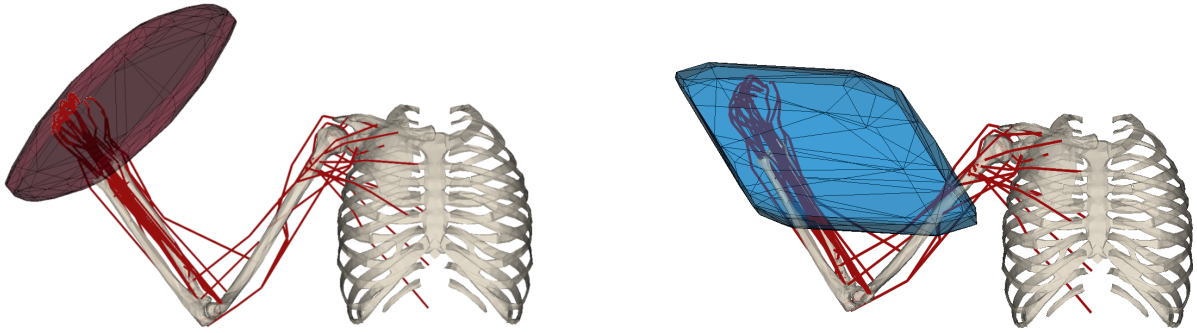

FIGURE 7.3: Cartesian acceleration (red) and force (blue) polytope of a musculoskeletal model of human upper limb with 7DOF and 50 muscles each, visualised with `biorbd`. The polytopes are scaled with a ratio 1m : 5000m/s^2 and 1m : 2000N respectively.

For the human musculoskeletal models this package implements the polytope and ellipsoid evaluation functions for the following metrics. All the metrics are implemented within `pycapacity.human` module.

Ellipsoids

- Velocity (manipulability) ellipsoid, as described by Yoshikawa [22]

$$\mathcal{E}_v = \{\dot{\boldsymbol{x}} \mid J\dot{\boldsymbol{q}} = \dot{\boldsymbol{x}}, \quad ||W^{-1}\dot{\boldsymbol{q}}|| \leq 1\}, \qquad W = \text{diag}(\dot{\boldsymbol{q}}_{max}) \tag{7.11}$$

  As well as its more complete formulation including muscular stretching velocities $\dot{\boldsymbol{l}}$

$$\mathcal{E}_v = \{\dot{\boldsymbol{x}} \mid J\dot{\boldsymbol{q}} = \dot{\boldsymbol{x}}, \; L\dot{\boldsymbol{q}} = \dot{\boldsymbol{l}} \quad ||W^{-1}\dot{\boldsymbol{l}}|| \leq 1\}, \qquad W = \text{diag}(\dot{\boldsymbol{l}}_{max}) \tag{7.12}$$

- Acceleration (dynamic manipulability) ellipsoid, as proposed by Khatib *et al.* [60]

$$\mathcal{E}_a = \{\ddot{\boldsymbol{x}} \mid \ddot{\boldsymbol{x}} = JM^{-1}N\boldsymbol{F}, \quad ||W^{-1}\boldsymbol{F}|| \leq 1\}, \qquad W = \text{diag}(\boldsymbol{F}_{max}) \tag{7.13}$$

- Force ellipsoid, as proposed by Petrič *et al.* [70]

$$\mathcal{E}_f = \{\boldsymbol{f} \mid N\boldsymbol{F} = J^T\boldsymbol{f}, \quad ||W^{-1}\boldsymbol{F}|| \leq 1\}, \qquad W = \text{diag}(\boldsymbol{F}_{max}) \tag{7.14}$$

In the above definitions, J is the robot Jacobian matrix, M is the inertia matrix, L is the muscle length Jacobian matrix and $N = -L^T$ is the moment arm matrix. f is the vector of Cartesian Space (CS) forces, $\dot{x}$ and $\ddot{x}$ are vectors for CS velocities and accelerations, q is the vector of Joint Space (JS) positions, $\dot{q}$ is the vector of the JS velocities and τ is the vector of

JS torques, $\dot{l}$ is the vector of the muscle stretching velocities and F is the vector of muscular forces. Matrix W is a scaling matrix that normalises the JS space limits.

Polytopes

- Velocity polytope, described in Section 2.2.5

$$\mathcal{P}_v = \{\dot{\boldsymbol{x}} \mid \dot{\boldsymbol{l}} = L\dot{\boldsymbol{q}},\ \dot{\boldsymbol{x}} = J\dot{\boldsymbol{q}},\ \dot{\boldsymbol{q}}_{min},\ \leq \dot{\boldsymbol{q}} \leq \dot{\boldsymbol{q}}_{max},\ \dot{\boldsymbol{l}}_{min} \leq \dot{\boldsymbol{l}} \leq \dot{\boldsymbol{l}}_{max}\} \tag{7.15}$$

- Acceleration polytope, described in Section 2.2.4

$$\mathcal{P}_a = \{\ddot{\boldsymbol{x}} \mid \ddot{\boldsymbol{x}} = JM^{-1}N\boldsymbol{F},\ \boldsymbol{F}_{min} \leq \boldsymbol{F} \leq \boldsymbol{F}_{max}\} \tag{7.16}$$

- Force polytope, described in Section 2.2.3

$$\mathcal{P}_f = \{\boldsymbol{f} \mid J^T\boldsymbol{f} = N\boldsymbol{F},\ \boldsymbol{F}_{min} \leq \boldsymbol{F} \leq \boldsymbol{F}_{max}\} \tag{7.17}$$

7.4 Implemented polytope evaluation algorithms

In addition to the efficient tools for generic polytope manipulation, listed in Section 7.2.2, the package implements several algorithms for polytope evaluation of the common formulations of physical ability polytopes of humans and robots

- Hyper-Plane Shifting Method (HPSM)
- Vertex Enumeration Algorithm (VEPOLI^2)
- Iterative Convex Hull Method (ICHM)

These algorithms are all implemented in Python and used to evaluate different polytope based physical ability metrics. Additionally, the algorithms are available to the users to be used standalone as well, and can be accessed through the module `pycapacity.algorithms`.

7.4.1 Hyper-plane shifting method (HPSM)

This is an algorithm based on the article by Gouttefarde and Krut [147] which presents an efficient way of determining the minimal half-space $\mathcal{H}$-representation of polytopes described by the equation

$$\mathcal{P} = \{\boldsymbol{x} \mid \boldsymbol{x} = By, \quad y_{min} \leq y \leq y_{max}\} \tag{7.18}$$

7.4.2 Vertex enumeration algorithm (VEPOLI^2)

This is an algorithm proposed in the context of this thesis, it is introduced in Section 3.4. This algorithm presents an efficient method for finding vertex $\mathcal{V}$-representation of polytopes described by the equation

$$\mathcal{P} = \{\boldsymbol{x} \mid A\boldsymbol{x} = \boldsymbol{y}, \quad \boldsymbol{y}_{min} \leq \boldsymbol{y} \leq \boldsymbol{y}_{max}\} \tag{7.19}$$

7.4.3 Iterative convex-hull method (ICHM)

This is an algorithm proposed in the context of this thesis as well, it is introduced in Section 3.5. This algorihtm implements an efficient method which iteratively approximates polytopes with

a formulation

$$\mathcal{P} = \{\boldsymbol{x} \mid A\boldsymbol{x} = B\boldsymbol{y}, \quad \boldsymbol{y}_{min} \leq \boldsymbol{y} \leq \boldsymbol{y}_{max}\} \tag{7.20}$$

The method finds both vertex $\mathcal{V}$ and half-plane $\mathcal{H}$ representation of the polytope at the same time. The ICHM method implemented within the `pycapacity` package can be additionally extended to the case where there is an additional projection matrix P making a class of problems

$$\mathcal{P} = \{\boldsymbol{x} \mid \boldsymbol{x} = P\boldsymbol{z},\ A\boldsymbol{z} = B\boldsymbol{y}, \quad \boldsymbol{y}_{min} \leq \boldsymbol{y} \leq \boldsymbol{y}_{max}\} \tag{7.21}$$

7.5 Polytope metrics evaluation algorithms and their performance analysis

As describe more in detail in Chapter 3, the applicable methods to evaluate different polytope based physical abilities depend on the family of problems they correspond to. Therefore, this section brings the information about which algorithm is used for which polytope metric and provides a brief performance evaluation of their execution times.

Additionally, to give brief information about the efficiency of the proposed methods, the section provides the execution times of the methods for the example problems. However, as these execution times can vary significantly depending on the complexity of the model used and the hardware it is run on, the users are encouraged to run the benchmark scripts themselves to get the most accurate results. This package provides several benchmarking scripts in the `examples` folder[1].

7.5.1 Robotic manipulators

In case of robotic manipulators the polytope evaluation methods used are given in Table 7.1.

Polytope Metric	Algorithm	Problem type	Execution time [ms] mean ± std. (max)
Velocity	HPSM	$\boldsymbol{x} = B\boldsymbol{y},\ \boldsymbol{y} \in [\boldsymbol{y}_{min},\ \boldsymbol{y}_{max}]$	3.6 ± 0.21 (5.7)
Acceleration	HPSM	$\boldsymbol{x} = B\boldsymbol{y},\ \boldsymbol{y} \in [\boldsymbol{y}_{min},\ \boldsymbol{y}_{max}]$	6.6 ± 1.4 (14.2)
Force	VEPOLI^2	$A\boldsymbol{x} = \boldsymbol{b},\ \boldsymbol{b} \in [\boldsymbol{b}_{min},\ \boldsymbol{b}_{max}]$	6.8 ± 0.88 (16.4)
Force intersection	VEPOLI^2	$A\boldsymbol{x} = \boldsymbol{b},\ \boldsymbol{b} \in [\boldsymbol{b}_{min},\ \boldsymbol{b}_{max}]$	98.2 ± 29.33 (165.8)
Force sum	VEPOLI^2	$A\boldsymbol{x} = \boldsymbol{b},\ \boldsymbol{b} \in [\boldsymbol{b}_{min},\ \boldsymbol{b}_{max}]$	17.1 ± 3.4 (44.9)
Reachable space	ICHM	$\boldsymbol{x} = B\boldsymbol{y},\ \boldsymbol{y} \in P_y$	30.5 ± 6.6 (76.7)

TABLE 7.1: Performance evaluation of polytope metrics for robotic manipulators.

The average execution time is calculated for 1000 random configuration of a 7 DOF Franka Emika panda robot, the model was used with `pinocchio` [166] library. All the experiments are run on a computer equipped with a 1.90GHz Intel i7-8650U processor. The results are obtained using the benchmarking script provided by the repository in the `examples` folder[1]. The reachable space polytope is calculated using the ICHM algorithm with the precision $\varepsilon = 1$mm.

7.5.2 Musculoskeletal models

In case of human musculoskeletal models the methods used are given in Table 7.2.

The average execution time is calculated for 1000 random configuration of a 50 muscle 7 DOF musculoskeletal model described by Holzbaur *et al.* [175], the model was used with `biorbd`

[1]Benchmarking examples: https://github.com/auctus-team/pycapacity/tree/master/examples

Polytope Metric	Algorithm	Problem type	Execution time [ms] mean ± std. (max)
Force	ICHM	$A\boldsymbol{x} = B\boldsymbol{y},\ \boldsymbol{y} \in [\boldsymbol{y}_{min},\ \boldsymbol{y}_{max}]$	186.8 ± 45.6 (281.6)
Acceleration	HPSM or ICHM	$\boldsymbol{x} = B\boldsymbol{y},\ \boldsymbol{y} \in [\boldsymbol{y}_{min},\ \boldsymbol{y}_{max}]$	378.8 ± 62.3 (643.7)
Velocity	ICHM	$\boldsymbol{x} = B\boldsymbol{y},\ \boldsymbol{y} \in P_y$	223.1 ± 60.4 (389.1)

TABLE 7.2: Performance evaluation of polytope metrics for human musculoskeletal models.

[159] biomechanics library. The experiments are run on a computer equipped with a 1.90GHz Intel i7-8650U processor. The results are obtained using the benchmarking script provided by the repository in the `examples` folder[1]. All the polytopes are calculated using the ICHM algorithm with the precision $\varepsilon = 10$ (N, m/s^2 and m/s).

7.6 Package overview

Python package `pycapacity` is publicly available both as a GitHub repository[2] and as a `pip` package[3]. Furthermore, the package has up-to-date and comprehensive documentation[4] including many examples, showing both the package's functionalities and how to integrate it with other libraries. Installing the package can be done using a single line of code, using `pip` package manager

```
> pip install pycapacity
```

pycapacity

Package structure

The package is divided in 6 modules which can be used as standalone tools as well

- `pycapacity.objects` - Module implementing generic `Polytope` and `Ellipsoid` classes
- `pycapacity.algorihtms` - Polytope evaluation algorithms described in Section 7.4
- `pycapacity.human` - Physical ability metrics for humans described in Section 7.3.2
- `pycapacity.robot` - Physical ability metrics for robots described in Section 7.3.1
- `pycapacity.visual` - 2D and 3D visualisation of polytopes and ellipsoids
- `pycapacity.examples` - Module implementing different toy models for fast prototyping

Supported libraries

The package is modular and can be easily integrated with different robotics and biomechanics libraries. The documentation provides tutorials helping to jump start practical applications[5].

- `OpenSim`[176] - https://github.com/opensim-org
- `biorbd`[159] - https://github.com/pyomeca/biorbd
- `pinocchio`[166] - https://github.com/stack-of-tasks/pinocchio
- `roboticstoolbox`[201] - https://github.com/petercorke/robotics-toolbox-python

[2] GitHub repo: https://github.com/auctus-team/pycapacity
[3] `pip` package: https://pypi.org/project/pycapacity/
[4] Documentation: https://auctus-team.github.io/pycapacity/
[5] Tutorials: https://auctus-team.github.io/pycapacity/examples/index.html

Example program

To demonstrate the user-friendly nature of `pycpacity`, this section provides an illustrative example involving the computation of a Cartesian Space (CS) force for a $m = 3$ polytope $\mathcal{P}_f$. This polytope is associated with a randomised robot manipulator featuring $n = 6$ degrees of freedom. Furthermore, the obtained polytope $\mathcal{P}_f$ is intersected with a generic cube $\mathcal{C}_f$ and visualised. The resulting program output is displayed in Figure 7.4 for reference.

```
# robot capacity module
from pycapacity import robot, visual, objects
import matplotlib.pyplot as plt
import numpy as np

# radomised robot data
m, n = 3, 6 # 3d forces, 6 dof
# joint torque limits max and min
t_min, t_max = -np.ones(n), np.ones(n)
# random jacobian matrix
J = np.array(np.random.rand(m,n))*2-1

# calculate the force polytope
P_f = robot.force_polytope(J, t_min, t_max)

# define a cube with H-representation
I = np.eye(3)
C_f = objects.Polytope(H = np.vstack((I,-I)),
                       d = np.ones(6))
# calculate the intersection
P_int = P_f & C_f

# plotting the polytope
visual.plot_polytope(plot=plt,
                     polytope=P_f,
                     label='$P_f$',
                     edge_color='black',
                     alpha=0.2)
# plotting the cube
visual.plot_polytope(plot=plt,
                     polytope=C_f,
                     label='$C_f$',
                     color='red',
                     alpha=0.1)
# plotting the intersection
visual.plot_polytope(plot=plt,
                     polytope=P_int,
                     label='$P_f\cap C_f$',
                     color='yellow',
                     alpha=0.5)
plt.legend()
plt.show()
```

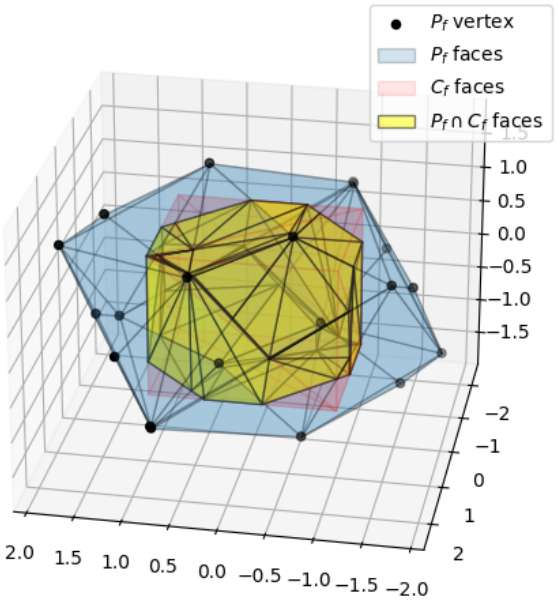

FIGURE 7.4: Output of the example program.

7.7 Conclusion

This chapter presents the `pycapacity` Python package, a toolkit designed to evaluate task-space physical ability metrics for both humans and robots, based on polytopes and ellipsoids. The aim of this package is to provide efficient tools for evaluating these metrics within an easily accessible framework, which can seamlessly integrate with standard robotics and biomechanics libraries. By implementing state-of-the-art algorithms for polytope evaluation, `pycapacity`

enables the evaluation of these metrics in an efficient manner, making them applicable for interactive online applications.

Moreover, `pycapacity` has several additional features facilitating its use in practical applications. It implements many common polytope manipulation operations (such as Minkowski sum, intersection, Chebyshev ball stc.) allowing users to seamlessly work with these mathematical structures and make efficient operations over the physical ability polytopes. The package implements tools for 2D and 3D visualisation of both polytopes and ellipsoids, aiding in the interpretation and communication of results. Furthermore, `pycapacity` has a relatively complete documentation, providing users with practical examples demonstrating both the package features and its integration with other software packages. The package provides the tools for the performance evaluation of the polytope evaluation algorithms, enabling users to assess and optimise the efficiency of their applications.

In summary, `pycapacity` aims to provide a versatile set of tools for robotics and biomechanics communities, enabling them to efficiently evaluate, manipulate and visualise different physical ability metrics based on polytopes and ellipsoids. The package is implemented in an user-friendly fashion, enabling an easy setup of user applications and having the potential to bring these metrics to the wider community.

Article **A. Skuric** *et al.* [A6], introducing the `pycapacity` package, has been recently published in Journal of Open Source Software (JOSS)[6].

[6] Journal of Open Source Software (JOSS): https://joss.theoj.org/

Chapter 8

Conclusion

This thesis is built on a vision of robotics and the industry of the future centred around humans. In this future, robots are no longer just tools for replacing human labour; instead, they become their active assistants: coexisting in their close proximity and interacting physically when executing tasks. Such collaborative systems benefit from both humans' and robots' individual set of abilities, potentially improving their efficiency and enhancing human safety and well-being through individualised robotic assistance.

Creating such flexible collaborative systems in practice requires having a set of tools for characterising differnet abilities required to accomplish different tasks (RQ1), as well as the individual abilities of humans and robots in a unified manner (RQ2). Furthermore, in order to decide if a task is better suited for humans or robots, or if it might require their collaboration, their common abilities when collaborating (as one single system) have to be quantified as well (RQ3). Moreover, providing operators with personalised assistance requires quantifying different notions of the extent of assistance the operator needs, as well as their safety and well-being (RQ4). Finally, as both their abilities and different safety concerns can evolve in time and can change significantly during the task execution, tools able to capture these changes online are needed (RQ5).

This thesis argues in favour of physical ability metrics as promising tools that can potentially answer to all these requirements (RQ1-RQ5). More specifically, this thesis concentrates on polytope characterisations of human's and robot's physical abilities, as one of their most accurate characterisations. Different physical ability polytopes are well known tools for characterising robot's and human's physical abilities (RQ2), and as shown in Chapter 2, they have a great potential to be used to characterise their common abilities when collaborating (RQ3). Moreover, polytopes are local metrics, being calculated for any given robot's or human's state. Therefore, they enable capturing the state dependent and changing nature of their physical abilities (RQ5). Finally, polytopes enable evaluating the need of assistance of the operators as their lacking physical ability to execute a certain task. This need of assistance can be quantified by comparing the physical abilities required to execute certain task (RQ1), with the changing physical abilities of the operator. In that way, polytopes enable creating assistive robot control strategies that guarantee that the operator's physical abilities are never surpassed which has a direct impact on their safety and well-being (RQ4).

8.1 Thesis contributions

Due to the relatively high computational complexity of polytope evaluation, real-time (interactive) applications of physical ability polytope are still relatively rare in practice. Therefore, the focus of the first two chapters of this thesis (Chapter 2 and Chapter 3) is put on developing

efficient tools for evaluating physical ability polytopes, aiming to enable interactive and online applications.

Chapter 2 brings an overview of common physical ability polytopes formulations applicable for both humans and robots. The chapter then proposes the use of efficient polytope algebra operations to characterise their common physical abilities when collaborating in the polytope form as well. Therefore, the chapter shows that the polytopes give a unified view on their individual abilities, as well as their abilities when collaborating physically. To go a step further, the chapter proposes a synthesis of the described polytope formulations in a form of a single generic polytope formulation unifying all the common physical ability polytopes of humans and robots.

Building on this generic formulation, Chapter 3 then concentrates on efficient methods for transforming these polytopes to their standard representations that can be used with practical applications. First, the chapter proposes a structured overview of different families of polytope formulations, derived from the generic formulation, with respect to their transformation strategy. Then the chapter brings an overview of standard polytope transformation strategies applicable to the proposed families of formulations. Following the literature overview of the standard strategies, two new polytope evaluation algorithms VEPOLI^2 and ICHM are introduced. The algorithms' complexity is experimentally evaluated and compared against the state of the arm methods. The results show that the algorithms substantially reduce the complexity and the computation time of the standard methods and have a potential to be used in online applications.

The following three chapters (Chapter 4, Chapter 5 and Chapter 6) bring the applications of real-time polytope evaluation in the resolution on different robotics problems in the context of human-robot collaboration.

Chapter 4 focuses on a challenging problem of developing robot control strategies for an efficient human-robot physical interaction. The chapter is set in the context of collaborative carrying of a heavy object, inspired by the LiChIE project. The chapter shows two collaboration experiments: dual robot arm collaborative carrying and human-robot collaborative carrying. In these experiments the real-time polytope evaluation is used to capture the changing physical abilities of the human and the robot online. This real-time information is then used for creating robot control strategies that adapt to their changes and distribute the weight accordingly. In the dual robot experiment, two Franka Emika Panda robots carry a 12kg object, largely above their rated capacity (6kg). The results show that by having real-time information about both robots' carrying capacities, the proposed collaborative control strategy adapted to their changes in real-time and successfully distributes the weight of the object during the whole duration of the experiment. In the human-robot collaborative carrying experiment, human operator and a Franka robot carry 7kg object. The experiment shows that, having online information about their changing capacity enabled to use the robot's and human's physical potential without compromising their safety. Furthermore, even though neither the robot nor the human would have been able to carry the entire object's weight on their own, by collaborating they were able to accomplish the task.

Chapter 5 focuses on developing tools to provide the operator with the interactive insight into the robot's current state and its current abilities in the context of the human-robot interaction. The chapter argues that the polytope representation of robot's physical abilities could be used as a valuable visual communication tool. Especially since polytopes can be transformed into triangulated meshes that can be easily visualised with standard visualisation tools. Moreover, the chapter introduces a new polytope formulation developed particularly with the visualisation in mind, representing the robot's reachable space within a time horizon. Finally, the chapter

brings the preliminary work on the development of the testing platform, based on Augmented Reality (AR) tools, with the long-term goal to validate the efficiency of polytopes in sharing information and their different visualisation modalities.

Chapter 6 concentrates on developing the trajectory planning strategies suitable for dynamical environments, particularly present when it comes to the human-robot collaboration. Such trajectory planning strategies require being reactive to the changes in the environment, while at the same efficiently exploiting robot's movement capacity. The chapter brings a new Cartesian Space (CS) trajectory planning strategy that calculates the robot's movement capacity in real-time, using efficient polytope algebra tools, and re-plans the updated trajectory at each step to account for their changes. Moreover, by re-planning in real-time the proposed method is able to be reactive to the environmental changes as well. The method's time-efficiency is confirmed experimentally, by benchmarking it against the state of the art offline methods. The results shows that the proposed online re-planning method has similar trajectory execution times as the offline time-optimal methods, without requiring any in-advance computation. The method's practical utility is demonstrated using a mock-up experiment in the context of human-robot collaborative waste sorting. In this experiment the proposed method is used to plan for time-efficient trajectories on the fly and pick-and-place the waste items into the appropriate sorting bins. The waste items are introduced by the operator, while the robot does not have any a proiri knowledge about their position, orientation or the time they will arrive.

Finally, Chapter 7 presents the `pycapacity` package. An efficient framework for calculating different physical ability metrics for both humans and robots, based on polytopes and ellipsoids. The aim of the package is to provide an efficient set of tools for evaluation of different physical ability metrics in an easy to use framework. The package implements several state of the art algorithms for polytope evaluation and manipulation, including VEPOLI^2 and ICHM developed in the context of this thesis, bringing many of them to the interactive (online) capable execution times. At the same time, `pycapacity` is open-source, easy to install and interface with other standard robotics and biomechanics libraries, and has relatively extensive documentation. Finally, the package is written in an user-friendly way with the aim to facilitate building applications and potentially bring these metrics to the wider community.

In conclusion, this thesis focuses on the vision of the future where the industry and the robotics are more human-centred and have a high degree of human-robot collaboration. Such close collaboration has a potential to benefit from both of their individual strengths and enhance the collaboration efficiency, as well as human's safety and well-being. This thesis argues that the polytope characterisation of their physical abilities is a powerful tool that can help to make a step towards such future, providing an unified view on physical abilities of humans and robots. Therefore, in this thesis, a set of fundamental tools for polytope evaluation is developed enabling the efficient polytope evaluation and setting the foundation for their use in online and interactive applications. The thesis then demonstrates that the real-time physical ability polytopes evaluation can provide solutions to several challenging robotics questions in the context of human-robot interaction. Finally, this thesis presents the `pycapacity` package, aiming to facilitate the use of the polytope based tools and potentially bring them to the wider community.

8.2 Perspectives

This section discusses the directions for further investigation in order to enhance the research findings of this thesis, as well as to apply them to different problems and domains.

8.2.1 Capacity aware Cartesian Space motion planning

As described in Chapter 6, real-time polytope evaluation enables creating reactive and on the fly trajectory planning strategies capable of exploiting robot's full movement potential. As discussed in Section 6.9, the method proposed in this chapter has several limitations. The method makes the assumption of point-to-point straight-line paths, allowing it to consider only robot's movement capacity in the trajectory direction. Furthermore, the method uses Trapezoidal Acceleration Profile (TAP) as its time-optimal trajectory planning strategy, which does not allow for considering polytope shaped movement capacity limits. A natural extension of this method is towards Model Predictive Control (MPC) strategies that do not require any a priori on the path and can consider polytope shaped limits. Such MPC approaches would allow to generate highly reactive and adaptable robot's trajectories on the fly, without the need to specify the path, while at the same time exploiting robot's full motion capacity.

A step in this direction is described in the thesis of Nicolas Torres Alberto, a member of the AUCTUS team. Their work proposes an efficient linear formulation of the MPC in Cartesian Space (CS), capable of integrating the polytope formulation of robot's state dependent physical abilities and allowing for sub-millisecond execution times.

Such approaches are not only limited to the robot's movement capacity constraints. As described in Section 6.3, they can additionally integrate different task and environment related constraints as well. This property has a great potential to be used for human-robot comanipulaiton, where the robot's motion planning would consider human's movement capacity in addition to the robot's ones. For example, such human aware strategy could enable the robot to plan for trajectories that are safe for the operator while at the same time allowing the operator to adapt the trajectory on the fly.

8.2.2 Safety applications of reachable space approximation

Chapter 5 introduces two new characterisations of robot's reachable space within a horizon time, in the context of providing the operator with informative and timely visual feedback about robot's current state and its changing physical abilities. These metrics represent the Cartesian Space (CS) space the robot can reach, from its current position, within a certain time horizon. This metric could be potentially very useful for safety considerations. For example by evaluating if the operator entered the space reachable by the robot and adapting the robot's behaviour accordingly. Similar approaches are proposed by Pereira and Althoff [196] and Schepp *et al.* [197], where the robot's reachable space is approximated using ellipsoids and cylinders, while the horizon time of interest is robot's maximal stopping time.

However, as discussed in Chapter 5, both introduced metrics are approximations of the robot's real reachable space which is highly nonlinear and hard to characterise. Moreover, the methods do not provide any formal guarantees on their approximation accuracy. Therefore, in order to enable their use in such safety applications, further work in necessary to improve and guarantee the accuracy of proposed metrics.

8.2.3 Human-robot physical collaboration

Chapter 4 showcased an example application which requires a high degree of physical collaboration: collaborative carrying of a heavy object. In this example, the chapter shows that the real-time evaluation of human's and robot's physical abilities enables creating collaborative robot control strategies capable of efficiently distributing the weight between the robot and human, and at the same time guarantee their safety. In this example scenario, as the collaborative task is relatively simple, applying force in the vertical direction, both human's and robot's

physical ability to execute this task is obtained from their wrench capacity polytopes. The perspective of using their wrench polytopes to create more advanced human-centred assistance strategies is discussed in Section 4.4.1.

However, polytope based collaborative strategies potentially allow for more flexible scenarios, where multiple polytope metrics of multiple physical abilities of humans and robots can be considered. For example if the task requires applying a force and executing a movement at the same time. In that case, both wrench and movement capacity polytopes could be used within the collaborative robot control strategy, making sure to exploit both their wrench and movement abilities, while at the same time making sure to respect their safety.

Another promising application of physical ability polytopes in the context of human-robot interaction is in task distribution. As discussed in Chapter 2, both human's and robot's physical abilities can be accurately represented in polytope form, as well as their joint abilities when interacting physically. Therefore polytopes could be used as tools to evaluate if different tasks suite better human's or robot's set of skills or do they potentially require their collaboration.

Furthermore, as their physical abilities evolve in time, especially human ones, more dynamic task allocation strategies could be envisaged. Such dynamic task allocation strategies could benefit from the real-time evaluation of their abilities and take actions in real-time to avoid potential safety risks. Recently, a similar approach was proposed by Messeri *et al.* [249], where the dynamic task distribution strategy is coupled with the real-time evaluation of operators fatigue. The strategy proposed in their work distributes the tasks on the fly, by ensuring that the operator's fatigue level is within the safety limits.

8.2.4 Efficient physical ability polytope evaluation for biomechanics

The proposed polytope evaluation algorithm ICHM (described in Section 3.5), enables efficiently calculating many different physical ability polytopes for human musculoskeletal models. This algorithm opens many possibilities for wider use of human physical ability polytopes in the area of the human-robot collaboration, by both reducing the computation time and enabling the use of more detailed human models, better describing human subjects [177]. Section 4.4.2 discusses more in detail the potential benefits of using such detailed models in the context of human-robot collaboration.

However, the potential applications of the ICHM algorithm extend beyond robotics. Efficient evaluation of physical abilities of musculoskeletal models is a promising tool for biomechanics community as well. The method allows for analysing the physical abilities of very detailed musculoskeletal models, for which the standard methods would have been intractable.

One such use-case is developed in the context of the thesis of Gautier Laisne, AUCTUS team member, which aims to calibrate detailed musculoskeletal models to the human subjects. The thesis approaches this calibration by first measuring human subject's force capacity and then employs genetic algorithm based approach to find the parameters of the musculoskeletal model that produce the force polytope that match the measured data [182]. Their approach leverages the efficiency of the the ICHM algorithm, which is used to efficiently calculate the force polytope of the simulated musculoskeletal models.

Additionally, the flexibility of the method allows extending common polytope formulations with additional human body limitations and in that way increases the estimation accuracy. Recently, Rezzoug *et al.* [250] used the ICHM algorithm, to characterise the human arm wrench capacity polytope while taking in consideration the stability constrains of to glenohumeral (shoulder) joint.

8.2.5 Information sharing - visualisation

Chapter 5 discussed using polytope representation of robot's physical abilities for providing the real-time visual feedback to the operator about the robot's current state. Such interactive visualisation aims to improve the operators situational awareness which has a potential to improve the overall collaboration efficiency and operator's safety [186].

However, choosing the appropriate modality of polytope visualisation is a challenging scientific question. Zolotas *et al.* [79] showed that the direct visualisation of polytopes is not always the most intuitive for the operators. There are several possible explanations of these observations. Polytopes are relatively complex geometrical structures, with many faces and vertices, making it challenging to the operator to extract the useful information. Furthermore, physical ability polytopes often represent abstract physical quantities that might not always be easy to interpret to the operators. Therefore, in order to use polytope based visualisations as a source of useful information to the operators, it is important to determine the suitable visualisation modality, as well as the appropriate physical ability polytope with respect to their tasks.

In the context of this thesis, a preliminary work has been carried out on the testing platform for interactive visualisation, based on Augmented Reality (AR) tools. This platform has a long-term goal to enable studying and validating different ways of visualising polytope based metrics and quantify their information sharing effectiveness. Furthermore, the platform potentially enables testing different physical ability polytopes, while executing different tasks, and to study their influence on the collaborative performance and safety.

Appendix A

Performing operations over polytopes

Once the polytopes are transformed in standard forms such as $\mathcal{H}$ or $\mathcal{V}$-representations, they can be exploited to efficiently calculate different operations over multiple polytopes, such as Minkowski sums and intersections.

A.1 Minkowski sum of polytopes

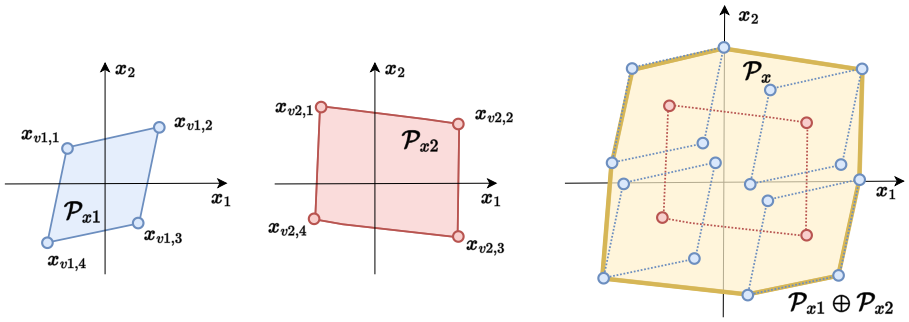

FIGURE A.1: A 2d ($m=2$) example of the construction of the intersection of two polytopes using their $\mathcal{V}$-representation. The points representing the vertices of the polytope $\mathcal{P}_{x1}$ are shown in blue, and for polytope $\mathcal{P}_{x2}$ in red. The Minkowski sum polytope $\mathcal{P}_x$ is formed by calculating all the combinations of all the vertices of both polytopes, and then calculating their convex hull, the hull is shown in yellow color.

In order to calculate Minkowski sum of multiple polytopes, for example two polytopes $\mathcal{P}_{x1}$ and $\mathcal{P}_{x2}$

$$\mathcal{P}_x = \mathcal{P}_{x1} \oplus \mathcal{P}_{x2} \tag{A.1}$$

the most straight-forward approach is to express both polytopes in their $\mathcal{V}$-representation form

$$\mathcal{P}_{x1} = \text{Conv}\left(\boldsymbol{x}_{v1,1},\ \boldsymbol{x}_{v1,2},\ \dots\ ,\boldsymbol{x}_{v1,N_1}\right) \tag{A.2}$$

$$\mathcal{P}_{x2} = \text{Conv}\left(\boldsymbol{x}_{v2,1},\ \boldsymbol{x}_{v2,2},\ \dots\ ,\boldsymbol{x}_{v2,N_2}\right) \tag{A.3}$$

where polytope $\mathcal{P}_{x1}$ has N_1 vertices $\boldsymbol{x}_{v1,i} \in \mathbb{R}^m$, and polytope $\mathcal{P}_{x2}$ has N_2 vertices $\boldsymbol{x}_{v2,i} \in \mathbb{R}^m$.

Then the Minkowski sum of the polytopes can be found as the convex-hull of all the $N_1 \cdot N_2$ combinations of all the vertices of the two polytopes

$$\begin{aligned} \mathcal{P}_x = conv\big\{&\boldsymbol{x}_{v1,1} + \boldsymbol{x}_{v2,1},\ \dots\ ,\boldsymbol{x}_{v1,N_1} + \boldsymbol{x}_{v2,1},\\ &\boldsymbol{x}_{v1,1} + \boldsymbol{x}_{v2,2},\ \dots\ ,\boldsymbol{x}_{v1,N_1} + \boldsymbol{x}_{v2,2},\\ &\dots\\ &\boldsymbol{x}_{v1,1} + \boldsymbol{x}_{v2,N_2},\ \dots\ ,\boldsymbol{x}_{v1,N_1} + \boldsymbol{x}_{v2,N_2}\big\} \end{aligned} \tag{A.4}$$

These $N_1 \cdot N_2$ are not all vertices of the polytope $\mathcal{P}_x$, some of them are inside of the polytope, as shown on a graphical example of the Minkowski sum of two 2d ($m = 2$) polytopes on Figure A.1. In order to find the minimal set of vertices of the polytope $\mathcal{P}_x$, defining its $\mathcal{V}$-representation, different convex-hull algorithms [74] can be used.

Once the $\mathcal{V}$-representation is determined, different standard representation conversion algorithms (Section 3.3.1), can be used to efficiently find its $\mathcal{H}$-representation.

A.1.1 Special case - projection formulation

A special case of simplified Minkowski sum calculation arises when polytopes $\mathcal{P}_{x1}$ and $\mathcal{P}_{x2}$ both have projection formulation

$$\mathcal{P}_{x1} = \{\boldsymbol{x}_1 \in \mathbb{R}^m | \ \boldsymbol{x}_1 = B_1 \boldsymbol{y}_1, \ \boldsymbol{y}_1 \in \mathcal{I}_1\} \tag{A.5}$$

$$\mathcal{P}_{x2} = \{\boldsymbol{x}_2 \in \mathbb{R}^m | \ \boldsymbol{x}_2 = B_2 \boldsymbol{y}_2, \ \boldsymbol{y}_2 \in \mathcal{I}_2\} \tag{A.6}$$

where both polytopes are defined in the common m dimensional output space $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^m$, while their input spaces $\boldsymbol{y}_1 \in \mathbb{R}^{n_1}$,$\boldsymbol{y}_2 \in \mathbb{R}^{n_2}$ might have different dimensions $n_1 \neq n_2$ and are limited, in generic case, by polytope input sets $\mathcal{I}_1$ and $\mathcal{I}_2$.

$$\mathcal{I}_1 = \{\boldsymbol{y}_1 \in \mathbb{R}^{n_1} \mid H_1 \boldsymbol{y}_1 \leq \boldsymbol{d}_1\}, \qquad \mathcal{I}_2 = \{\boldsymbol{y}_2 \in \mathbb{R}^{n_2} \mid H_2 \boldsymbol{y}_2 \leq \boldsymbol{d}_2\} \tag{A.7}$$

As the Minkowski sum of two polytopes can then be expressed as the achievable set of output variable $\boldsymbol{x} \in \mathbb{R}^m$ corresponding the sum of the variables $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$

$$\mathcal{P}_x = \mathcal{P}_{x1} \oplus \mathcal{P}_{x2} = \{\boldsymbol{x} \in \mathbb{R}^m \mid \boldsymbol{x} = \boldsymbol{x}_1 + \boldsymbol{x}_2, \quad \boldsymbol{x}_1 \in \mathcal{P}_{x1}, \ \boldsymbol{x}_2 \in \mathcal{P}_{x2}, \} \tag{A.8}$$

the projection formulation of their Minkowski sum can be expressed directly

$$\mathcal{P}_x = \left\{ \boldsymbol{x} \in \mathbb{R}^m \ \middle| \ \boldsymbol{x} = \begin{bmatrix} B_1 & 0 \\ 0 & B_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \end{bmatrix}, \ \begin{bmatrix} H_1 \\ H_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \end{bmatrix} \leq \begin{bmatrix} \boldsymbol{d}_1 \\ \boldsymbol{d}_2 \end{bmatrix} \right\} \tag{A.9}$$

Finding the $\mathcal{V}$ and $\mathcal{H}$-representation of this polytope can then be done using the approaches for polytope with projection formulation, described in Section 3.3.3.

The same logic can be used if the Minkowski sum is calculated for more than two polytopes

$$\mathcal{P}_x = \mathcal{P}_{x1} \oplus \mathcal{P}_{x2} \oplus \ \cdots \tag{A.10}$$

A.2 Polytope intersection

In order to calculate an intersection of multiple polytopes, for example two polytopes $\mathcal{P}_{x1}$ and $\mathcal{P}_{x2}$

$$\mathcal{P}_x = \mathcal{P}_{x1} \cap \mathcal{P}_{x2} \tag{A.11}$$

the simplest approach is to transform both polytopes to their $\mathcal{H}$-representation

$$\mathcal{P}_{x1} = \{\boldsymbol{x} \in \mathbb{R}^m \mid H_1 \boldsymbol{x} \leq \boldsymbol{d}_1\} \tag{A.12}$$

$$\mathcal{P}_{x2} = \{\boldsymbol{x} \in \mathbb{R}^m \mid H_2 \boldsymbol{x} \leq \boldsymbol{d}_2\} \tag{A.13}$$

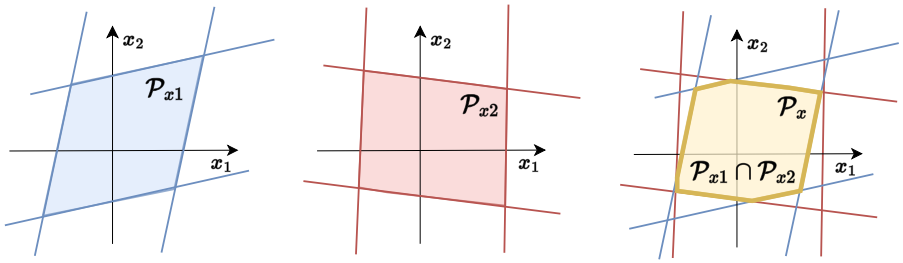

FIGURE A.2: A 2d ($m=2$) example of the construction of the intersection of two polytopes using their $\mathcal{H}$-representation. The lines representing half-planes of the polytope $\mathcal{P}_{x1}$ are shown in blue, and for polytope $\mathcal{P}_{x2}$ in red. The intersection polytope $\mathcal{P}_x$ is formed by including all of their half-planes, where the space representing their intersection is shown in yellow.

where polytopes $\mathcal{P}_{x1}$ and $\mathcal{P}_{x2}$ represent limits of the same output variable $\boldsymbol{x} \in \mathbb{R}^m$. Additionally matrices $H_i \in \mathbb{R}^{m \times N_i}$ and vectors $\boldsymbol{d}_i \in \mathbb{R}^{N_i}$ define the half-plane representation of the polytopes $\mathcal{P}_{xi}$ with N_i faces.

Given their $\mathcal{H}$-representations the intersections of these polytopes can be expressed directly as

$$\mathcal{P}_x = \left\{ \boldsymbol{x} \in \mathbb{R}^m \;\middle|\; \begin{bmatrix} H_1 \\ H_2 \end{bmatrix} \boldsymbol{x} \leq \begin{bmatrix} \boldsymbol{d}_1 \\ \boldsymbol{d}_2 \end{bmatrix} \right\} \tag{A.14}$$

by stacking their matrices H_i and vectors $\boldsymbol{d}_i$ forming the $\mathcal{H}$-representation of the polytope $\mathcal{P}_x$. The obtained $\mathcal{H}$-representation might not be minimal though, it might have some redundant inequalities. If the application requires it, they can be removed using standard algorithms [138]. A visual example of constructing the intersection of $m = 2$ dimensional polytopes using their $\mathcal{H}$-representation is shown on Figure A.2.

Once the $\mathcal{H}$-representation is determined, different standard representation conversion algorithms (Section 3.3.1), can be used to efficiently find its $\mathcal{V}$-representation.

A.2.1 Special case - intersection formulation

A special case of simplified polytope intersection calculation arises when polytopes $\mathcal{P}_{x1}$ and $\mathcal{P}_{x2}$ both have intersection formulation

$$\mathcal{P}_{x1} = \{\boldsymbol{x} \in \mathbb{R}^m | \; A_1 \boldsymbol{x} = \boldsymbol{y}_1, \; \boldsymbol{y}_1 \in \mathcal{I}_1\} \tag{A.15}$$

$$\mathcal{P}_{x2} = \{\boldsymbol{x} \in \mathbb{R}^m | \; A_2 \boldsymbol{x} = \boldsymbol{y}_2, \; \boldsymbol{y}_2 \in \mathcal{I}_2\} \tag{A.16}$$

where both polytopes are defined in the common m dimensional output space $\boldsymbol{x} \in \mathbb{R}^m$, while their input spaces $\boldsymbol{y}_1 \in \mathbb{R}^{n_1}$,$\boldsymbol{y}_2 \in \mathbb{R}^{n_2}$ might have different dimensions $n_1 \neq n_2$ and are limited, in generic case, by polytope input sets $\mathcal{I}_1$ and $\mathcal{I}_2$.

$$\mathcal{I}_1 = \{\boldsymbol{y}_1 \in \mathbb{R}^{n_1} \mid H_1 \boldsymbol{y}_1 \leq \boldsymbol{d}_1\}, \qquad \mathcal{I}_2 = \{\boldsymbol{y}_2 \in \mathbb{R}^{n_2} \mid H_2 \boldsymbol{y}_2 \leq \boldsymbol{d}_2\} \tag{A.17}$$

Using these polytope formulations their intersection polytope can be expressed in the intersection formulation by stacking the equations

$$\mathcal{P}_x = \left\{ \boldsymbol{x} \in \mathbb{R}^m \;\middle|\; \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} \boldsymbol{x} = \begin{bmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \end{bmatrix}, \; \begin{bmatrix} H_1 \\ H_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \end{bmatrix} \leq \begin{bmatrix} \boldsymbol{d}_1 \\ \boldsymbol{d}_2 \end{bmatrix} \right\} \tag{A.18}$$

Finding the $\mathcal{V}$ and $\mathcal{H}$-representation of this polytope can then be done using the approaches for polytope with intersection formulation, described in Section 3.3.2.

The same logic can be used if the intersection is calculated for more than two polytopes

$$\mathcal{P}_x = \mathcal{P}_{x1} \cap \mathcal{P}_{x2} \cap \ \cdots \tag{A.19}$$

Bibliography

[1] P. K. R. Maddikunta *et al.*, "Industry 5.0: A survey on enabling technologies and potential applications," *Journal of Industrial Information Integration*, vol. 26, p. 100 257, 2022, ISSN: 2452-414X. DOI: https://doi.org/10.1016/j.jii.2021.100257. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2452414X21000558
MENTIONED ON PAGE 1

[2] S. Huang, B. Wang, X. Li, P. Zheng, D. Mourtzis, and L. Wang, "Industry 5.0 and society 5.0—comparison, complementation and co-evolution," *Journal of Manufacturing Systems*, vol. 64, pp. 424–428, 2022, ISSN: 0278-6125. DOI: https://doi.org/10.1016/j.jmsy.2022.07.010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0278612522001224
MENTIONED ON PAGE 1

[3] X. Xu, Y. Lu, B. Vogel-Heuser, and L. Wang, "Industry 4.0 and industry 5.0—inception, conception and perception," *Journal of Manufacturing Systems*, vol. 61, pp. 530–535, 2021, ISSN: 0278-6125. DOI: https://doi.org/10.1016/j.jmsy.2021.10.006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0278612521002119
MENTIONED ON PAGE 1

[4] J. Leng *et al.*, "Industry 5.0: Prospect and retrospect," *Journal of Manufacturing Systems*, vol. 65, pp. 279–295, 2022, ISSN: 0278-6125. DOI: https://doi.org/10.1016/j.jmsy.2022.09.017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0278612522001662
MENTIONED ON PAGE 1

[5] A. C. Simões, A. Pinto, J. Santos, S. Pinheiro, and D. Romero, "Designing human-robot collaboration (hrc) workspaces in industrial settings: A systematic literature review," *Journal of Manufacturing Systems*, vol. 62, pp. 28–43, 2022, ISSN: 0278-6125. DOI: https://doi.org/10.1016/j.jmsy.2021.11.007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0278612521002296
MENTIONED ON PAGES 1, 77

[6] E. Coronado, T. Kiyokawa, G. A. G. Ricardez, I. G. Ramirez-Alpizar, G. Venture, and N. Yamanobe, "Evaluating quality in human-robot interaction: A systematic search and classification of performance and human-centered factors, measures and metrics towards an industry 5.0," *Journal of Manufacturing Systems*, vol. 63, pp. 392–410, 2022, ISSN: 0278-6125. DOI: https://doi.org/10.1016/j.jmsy.2022.04.007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0278612522000577
MENTIONED ON PAGE 2

[7] S. Patel and T. Sobh, "Manipulator performance measures-a comprehensive literature survey," *Journal of Intelligent & Robotic Systems*, vol. 77, pp. 547–570, 2015
MENTIONED ON PAGE 2

[8] C. Pholsiri, "Task-based decision making and control of robotic manipulators," Theses, The University of Texas at Austin, 2004. [Online]. Available: http://hdl.handle.net/2152/1388
Mentioned on page 2

[9] A. Golabchi, S. Han, J. Seo, S. Han, S. Lee, and M. Al-Hussein, "An automated biomechanical simulation approach to ergonomic job analysis for workplace design," *Journal of Construction Engineering and Management*, vol. 141, no. 8, p. 04 015 020, 2015. DOI: 10.1061/(ASCE)CO.1943-7862.0000998
Mentioned on page 2

[10] P. Chiacchio, S. Chiaverini, L. Sciavicco, and B. Siciliano, "Global task space manipulability ellipsoids for multiple-arm systems," en, *IEEE Transactions on Robotics and Automation*, vol. 7, no. 5, pp. 678–685, Oct. 1991, ISSN: 1042296X. DOI: 10.1109/70.97880. [Online]. Available: http://ieeexplore.ieee.org/document/97880/ (visited on 10/14/2020)
Mentioned on pages 2, 5, 6, 8, 17

[11] J. Lee, "Velocity workspace analysis for multiple arm robot systems," *Robotica*, vol. 19, no. 5, pp. 581–591, 2001
Mentioned on pages 2, 9, 28

[12] M. Russo, "Measuring performance: Metrics for manipulator design, control, and optimization," *Robotics*, vol. 12, no. 1, p. 4, 2022
Mentioned on page 4

[13] C. M. Gosselin and M. Guillot, "The Synthesis of Manipulators with Prescribed Workspace," *Journal of Mechanical Design*, vol. 113, no. 4, pp. 451–455, Dec. 1991, ISSN: 1050-0472. DOI: 10.1115/1.2912804. eprint: https://asmedigitalcollection.asme.org/mechanicaldesign/article-pdf/113/4/451/5920004/451_1.pdf. [Online]. Available: https://doi.org/10.1115/1.2912804
Mentioned on pages 4, 102, 117

[14] N. Vahrenkamp *et al.*, "Workspace analysis for planning human-robot interaction tasks," in *IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, 2016, pp. 1298–1303. DOI: 10.1109/HUMANOIDS.2016.7803437
Mentioned on pages 4, 102, 117

[15] S. Kucuk and Z. Bingul, "Robot workspace optimization based on a novel local and global performance indices," in *Proceedings of the IEEE International Symposium on Industrial Electronics, 2005. ISIE 2005.*, vol. 4, 2005, pp. 1593–1598
Mentioned on pages 4, 5, 102, 117

[16] J. .-. Merlet, "Determination of 6d workspaces of gough-type parallel manipulator and comparison between different geometries," *The International Journal of Robotics Research*, vol. 18, no. 9, pp. 902–916, 1999. DOI: 10.1177/02783649922066646. eprint: https://doi.org/10.1177/02783649922066646. [Online]. Available: https://doi.org/10.1177/02783649922066646
Mentioned on page 4

[17] Q. Jiang and C. M. Gosselin, "The Maximal Singularity-Free Workspace of the Gough–Stewart Platform for a Given Orientation," *Journal of Mechanical Design*, vol. 130, no. 11, p. 112 304, Sep. 2008, ISSN: 1050-0472. DOI: 10.1115/1.2976452. eprint: https://asmedigitalcollection.asme.org/mechanicaldesign/article-pdf/130/11/112304/5923427/112304_1.pdf. [Online]. Available: https://doi.org/10.1115/1.2976452
Mentioned on page 4

[18] M. Gouttefarde, J.-P. Merlet, and D. Daney, "Determination of the wrench-closure workspace of 6-dof parallel cable-driven mechanisms," in *Advances in Robot Kinematics: Mechanisms and Motion*, Springer, 2006, pp. 315–322
MENTIONED ON PAGES 4, 82

[19] D. Lau, D. Oetomo, and S. K. Halgamuge, "Wrench-Closure Workspace Generation for Cable Driven Parallel Manipulators Using a Hybrid Analytical-Numerical Approach," *Journal of Mechanical Design*, vol. 133, no. 7, Jul. 2011, ISSN: 1050-0472. DOI: 10.1115/1.4004222. eprint: https://asmedigitalcollection.asme.org/mechanicaldesign/article-pdf/133/7/071004/5925839/071004_1.pdf. [Online]. Available: https://doi.org/10.1115/1.4004222
MENTIONED ON PAGES 4, 82

[20] F. Zacharias, C. Borst, and G. Hirzinger, "Capturing robot workspace structure: Representing robot capabilities," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007, pp. 3229–3236. DOI: 10.1109/IROS.2007.4399105
MENTIONED ON PAGES 4, 102

[21] S. Patel and T. Sobh, "Manipulator performance measures - a comprehensive literature survey," *Journal of Intelligent & Robotic Systems*, vol. 77, no. 3, pp. 547–570, Mar. 2015, ISSN: 1573-0409. DOI: 10.1007/s10846-014-0024-y. [Online]. Available: https://doi.org/10.1007/s10846-014-0024-y
MENTIONED ON PAGE 4

[22] T. Yoshikawa, "Manipulability of Robotic Mechanisms," en, *The International Journal of Robotics Research*, vol. 4, no. 2, pp. 3–9, Jun. 1985, ISSN: 0278-3649, 1741-3176. DOI: 10.1177/027836498500400201. [Online]. Available: http://journals.sagepub.com/doi/10.1177/027836498500400201 (visited on 10/22/2020)
MENTIONED ON PAGES 4, 5, 17, 154–157, 159

[23] C. Gosselin and J. Angeles, "A Global Performance Index for the Kinematic Optimization of Robotic Manipulators," *Journal of Mechanical Design*, vol. 113, no. 3, pp. 220–226, Sep. 1991, ISSN: 1050-0472. DOI: 10.1115/1.2912772. eprint: https://asmedigitalcollection.asme.org/mechanicaldesign/article-pdf/113/3/220/5796880/220_1.pdf. [Online]. Available: https://doi.org/10.1115/1.2912772
MENTIONED ON PAGE 4

[24] J. P. Merlet, "Jacobian, Manipulability, Condition Number, and Accuracy of Parallel Robots," en, *Journal of Mechanical Design*, vol. 128, no. 1, pp. 199–206, Jan. 2006, ISSN: 1050-0472. DOI: 10.1115/1.2121740. [Online]. Available: https://asmedigitalcollection.asme.org/mechanicaldesign/article/128/1/199/470479/Jacobian-Manipulability-Condition-Number-and (visited on 10/25/2020)
MENTIONED ON PAGES 4, 17

[25] T. Yoshikawa, "Dynamic manipulability of robot manipulators," *Transactions of the Society of Instrument and Control Engineers*, vol. 21, no. 9, pp. 970–975, 1985
MENTIONED ON PAGES 4, 5

[26] A. Pashkevich, A. Klimchik, and D. Chablat, "Enhanced stiffness modeling of manipulators with passive joints," *Mechanism and Machine Theory*, vol. 46, no. 5, pp. 662–679, 2011, ISSN: 0094-114X. DOI: https://doi.org/10.1016/j.mechmachtheory.2010.12.008. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0094114X10002338
MENTIONED ON PAGE 4

[27] L. Guilamo, J. Kuffner, K. Nishiwaki, and S. Kagami, "Manipulability optimization for trajectory generation," in *Proceedings IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, 2006, pp. 2017–2022. DOI: 10.1109/ROBOT.2006.1642001
MENTIONED ON PAGE 5

[28] H. Asada, "Dynamic analysis and design of robot manipulators using inertia ellipsoids," in *Proceedings. IEEE International Conference on Robotics and Automation*, vol. 1, 1984, pp. 94–102. DOI: 10.1109/ROBOT.1984.1087211
MENTIONED ON PAGE 5

[29] A. Ajoudani, N. G. Tsagarakis, and A. Bicchi, "On the role of robot configuration in cartesian stiffness control," in *IEEE international conference on robotics and automation (ICRA)*, 2015, pp. 1010–1016
MENTIONED ON PAGES 5, 6, 21

[30] J. Lee, "A study on the manipulability measures for robot manipulators," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robot and Systems. Innovative Robotics for Real-World Applications. IROS '97*, vol. 3, 1997, 1458–1465 vol.3. DOI: 10.1109/IROS.1997.656551
MENTIONED ON PAGES 6, 17

[31] R. Finotello, T. Grasso, G. Rossi, and A. Terribile, "Computation of kinetostatic performances of robot manipulators with polytopes," in *Proceedings. IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, vol. 4, 1998, 3241–3246 vol.4. DOI: 10.1109/ROBOT.1998.680928
MENTIONED ON PAGES 6, 15, 18, 101

[32] G. Boschetti and R. Minto, "Kinematic directional index for the performance of redundant manipulators," *Robotica*, pp. 1–21, 2023. DOI: 10.1017/S0263574723000796
MENTIONED ON PAGE 6

[33] P. Long and T. Padir, "Evaluating robot manipulability in constrained environments by velocity polytope reduction," in *IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*, 2018, pp. 1–9. DOI: 10.1109/HUMANOIDS.2018.8624962
MENTIONED ON PAGES 6, 104, 154

[34] P. Long and T. Padır, "Constrained Manipulability for Humanoid Robots Using Velocity Polytopes," en, *International Journal of Humanoid Robotics*, vol. 17, no. 01, p. 1 950 037, Feb. 2020, ISSN: 0219-8436, 1793-6942. DOI: 10.1142/S0219843619500373. [Online]. Available: https://www.worldscientific.com/doi/abs/10.1142/S0219843619500373 (visited on 09/27/2020)
MENTIONED ON PAGES 6, 17, 101, 154

[35] P. Chiacchio, Y. Bouffard-Vercelli, and F. Pierrot, "Evaluation of force capabilities for redundant manipulators," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 4, Minneapolis, MN, USA, 1996, pp. 3520–3525, ISBN: 978-0-7803-2988-1. DOI: 10.1109/ROBOT.1996.509249. [Online]. Available: http://ieeexplore.ieee.org/document/509249/ (visited on 09/29/2020)
MENTIONED ON PAGES 6, 18, 49, 56, 58, 59, 63, 64, 154, 156

[36] P. Chiacchio, "A new dynamic manipulability ellipsoid for redundant manipulators," en, *Robotica*, vol. 18, no. 4, pp. 381–387, Jul. 2000, ISSN: 1469-8668, 0263-5747. DOI: 10.1017/S0263574799002106. (visited on 10/14/2020)
MENTIONED ON PAGES 6, 20, 157

[37] C. Pholsiri, C. Kapoor, and D. Tesar, "Real-time robot capability analysis," in *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, vol. 47446, 2005, pp. 973–982
MENTIONED ON PAGES 6, 15, 18

[38] J. N. Hodder, N. J. La Delfa, and J. R. Potvin, "Testing the assumption in ergonomics software that overall shoulder strength can be accurately calculated by treating orthopedic axes as independent," *Journal of Electromyography and Kinesiology*, vol. 29, pp. 50–54, 2016, ISSN: 1050-6411. DOI: https://doi.org/10.1016/j.jelekin.2015.05.002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1050641115001108
MENTIONED ON PAGE 6

[39] K. R. Holzbaur, S. L. Delp, G. E. Gold, and W. M. Murray, "Moment-generating capacity of upper limb muscles in healthy adults," *Journal of Biomechanics*, vol. 40, no. 11, pp. 2442–2449, 2007, ISSN: 0021-9290. DOI: https://doi.org/10.1016/j.jbiomech.2006.11.013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021929006004507
MENTIONED ON PAGES 6, 23

[40] M. N. Castro, J. Rasmussen, S. Bai, and M. S. Andersen, "The reachable 3-d workspace volume is a measure of payload and body-mass-index: A quasi-static kinetic assessment," *Applied Ergonomics*, vol. 75, pp. 108–119, 2019, ISSN: 0003-6870. DOI: https://doi.org/10.1016/j.apergo.2018.09.010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0003687018304071
MENTIONED ON PAGE 6

[41] D. M. Jessop and M. T. G. Pain, "Maximum velocities in flexion and extension actions for sport," en, *J Hum Kinet*, vol. 50, pp. 37–44, Apr. 2016
MENTIONED ON PAGE 6

[42] T. Tsuji, P. G. Morasso, K. Goto, and K. Ito, "Human hand impedance characteristics during maintained posture," *Biological Cybernetics*, vol. 72, no. 6, pp. 475–485, May 1995, ISSN: 1432-0770. DOI: 10.1007/BF00199890. [Online]. Available: https://doi.org/10.1007/BF00199890
MENTIONED ON PAGE 6

[43] P. K. Artemiadis, P. T. Katsiaris, M. V. Liarokapis, and K. J. Kyriakopoulos, "Human arm impedance: Characterization and modeling in 3d space," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 3103–3108. DOI: 10.1109/IROS.2010.5652025
MENTIONED ON PAGES 6, 7

[44] J. A. Haisma *et al.*, "Changes in physical capacity during and after inpatient rehabilitation in subjects with a spinal cord injury," *Archives of Physical Medicine and Rehabilitation*, vol. 87, no. 6, pp. 741–748, 2006, ISSN: 0003-9993. DOI: https://doi.org/10.1016/j.apmr.2006.02.032. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0003999306002012
MENTIONED ON PAGE 6

[45] S. Hignett and L. McAtamney, "Rapid entire body assessment (reba)," *Applied Ergonomics*, vol. 31, no. 2, pp. 201–205, 2000, ISSN: 0003-6870. DOI: https://doi.org/10.1016/S0003-6870(99)00039-3. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0003687099000393
MENTIONED ON PAGES 7, 86

[46] L. McAtamney and E. Nigel Corlett, "Rula: A survey method for the investigation of work-related upper limb disorders," *Applied Ergonomics*, vol. 24, no. 2, pp. 91–99, 1993, ISSN: 0003-6870. DOI: https://doi.org/10.1016/0003-6870(93)90080-S. [Online]. Available: https://www.sciencedirect.com/science/article/pii/000368709390080S
MENTIONED ON PAGE 7

[47] A. Yazdani, R. S. Novin, A. Merryweather, and T. Hermans, "Dula and deba: Differentiable ergonomic risk models for postural assessment and optimization in ergonomically intelligent phri," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 9124–9131. DOI: 10.1109/IROS47612.2022.9981528
MENTIONED ON PAGE 7

[48] Health and Safety Commission, *Manual Handling: Manual Handling Operations Regulations 1992, Guidance on Regulation L23*. HSE, 2016, ISBN: 978-0-717-66653-9
MENTIONED ON PAGES 7, 87

[49] NASA. "Man-systems integration standards." (), (visited on 09/10/2021)
MENTIONED ON PAGES 7, 87

[50] B. Busch, G. Maeda, Y. Mollard, M. Demangeat, and M. Lopes, "Postural optimization for an ergonomic human-robot interaction," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 2778–2785. DOI: 10.1109/IROS.2017.8206107
MENTIONED ON PAGE 7

[51] F. Ore, B. R. Vemula, L. Hanson, and M. Wiktorsson, "Human – industrial robot collaboration: Application of simulation software for workstation optimisation," *Procedia CIRP*, vol. 44, pp. 181–186, 2016, ISSN: 2212-8271. DOI: https://doi.org/10.1016/j.procir.2016.02.002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2212827116002195
MENTIONED ON PAGE 7

[52] P. Lietaert, N. Billen, and S. Burggraeve, "Model-based multi-attribute collaborative production cell layout optimization," in *2019 20th International Conference on Research and Education in Mechatronics (REM)*, 2019, pp. 1–7. DOI: 10.1109/REM.2019.8744136
MENTIONED ON PAGE 7

[53] P. Maurice, "Virtual ergonomics for the design of collaborative robots," Theses, Université Pierre et Marie Curie - Paris VI, Jun. 2015. [Online]. Available: https://hal.science/tel-01171482
MENTIONED ON PAGE 7

[54] J. Lenarcic and A. Umek, "Simple model of human arm reachable workspace," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 24, no. 8, pp. 1239–1246, 1994. DOI: 10.1109/21.299704
MENTIONED ON PAGE 7

[55] G. Kurillo, A. Chen, R. Bajcsy, and J. J. Han, "Evaluation of upper extremity reachable workspace using kinect camera," *Technology and Health Care*, vol. 21, pp. 641–656, 2013, ISSN: 1878-7401. DOI: 10.3233/THC-130764. [Online]. Available: https://doi.org/10.3233/THC-130764
MENTIONED ON PAGE 7

[56] L. F. C. Figueredo, R. C. Aguiar, L. Chen, S. Chakrabarty, M. R. Dogar, and A. G. Cohn, "Human comfortability: Integrating ergonomics and muscular-informed metrics for manipulability analysis during human-robot collaboration," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 351–358, 2021. DOI: 10.1109/LRA.2020.3043173
MENTIONED ON PAGE 7

[57] J. Jacquier-Bret, P. Gorce, and N. Rezzoug, "The manipulability: A new index for quantifying movement capacities of upper extremity," *Ergonomics*, vol. 55, no. 1, pp. 69–77, 2012. DOI: 10.1080/00140139.2011.633176. eprint: https://doi.org/10.1080/00140139.2011.633176. [Online]. Available: https://doi.org/10.1080/00140139.2011.633176
MENTIONED ON PAGE 7

[58] N. Rezzoug, J. Jacquier-Bret, V. Hernandez, and P. Gorce, "Application of robotic indices to evaluate human upper-limb force capacities," in *IECON 38th Annual Conference on IEEE Industrial Electronics Society*, 2012, pp. 5568–5573
MENTIONED ON PAGES 7, 24

[59] M. Sasaki, T. Iwami, K. Miyawaki, I. Sato, G. Obinata, and A. Dutta, "Higher dimensional spatial expression of upper limb manipulation ability based on human joint torque characteristics," in *Robot Manipulators New Achievements*, A. Lazinica and H. Kawai, Eds., Rijeka: IntechOpen, 2010, ch. 36. DOI: 10.5772/9344. [Online]. Available: https://doi.org/10.5772/9344
MENTIONED ON PAGE 7

[60] O. Khatib, E. Demircan, V. De Sapio, L. Sentis, T. Besier, and S. Delp, "Robotics-based synthesis of human motion," *Journal of physiology-Paris*, vol. 103, no. 3-5, pp. 211–219, 2009
MENTIONED ON PAGES 7, 26, 159

[61] M. G. Carmichael and D. Liu, "Estimating physical assistance need using a musculoskeletal model," *IEEE Transactions on Biomedical Engineering*, vol. 60, no. 7, pp. 1912–1919, Jul. 2013, ISSN: 1558-2531. DOI: 10.1109/TBME.2013.2244889
MENTIONED ON PAGES 7, 25, 55–57

[62] E. Demircan, T. F. Besier, and O. Khatib, "Muscle force transmission to operational space accelerations during elite golf swings," in *IEEE International Conference on Robotics and Automation*, 2012, pp. 1464–1469
MENTIONED ON PAGES 7, 26

[63] N. Rezzoug, V. Hernandez, and P. Gorce, "Upper-limb isometric force feasible set: Evaluation of joint torque-based models," *Biomechanics*, vol. 1, no. 1, pp. 102–117, 2021, ISSN: 2673-7078. DOI: 10.3390/biomechanics1010008. [Online]. Available: https://www.mdpi.com/2673-7078/1/1/8
MENTIONED ON PAGE 7

[64] V. Hernandez, N. Rezzoug, and P. Gorce, "Toward isometric force capabilities evaluation by using a musculoskeletal model: Comparison with direct force measurement," *Journal of Biomechanics*, vol. 48, no. 12, pp. 3178–3184, 2015, ISSN: 0021-9290. DOI: https://doi.org/10.1016/j.jbiomech.2015.07.003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021929015003802
MENTIONED ON PAGE 7

[65] M. E. Lund, M. S. Andersen, M. de Zee, and J. Rasmussen, "Scaling of musculoskeletal models from static and dynamic trials," *International Biomechanics*, vol. 2, no. 1, pp. 1–11, 2015. DOI: 10.1080/23335432.2014.993706. eprint: https://doi.org/10.1080/

23335432.2014.993706. [Online]. Available: https://doi.org/10.1080/23335432.2014.993706
Mentioned on page 7

[66] Z. Ding, C. K. Tsang, D. Nolte, A. E. Kedgley, and A. M. J. Bull, "Improving musculoskeletal model scaling using an anatomical atlas: The importance of gender and anthropometric similarity to quantify joint reaction forces," *IEEE Transactions on Biomedical Engineering*, vol. 66, no. 12, pp. 3444–3456, 2019. DOI: 10.1109/TBME.2019.2905956
Mentioned on page 7

[67] W. Kim, L. Peternel, M. Lorenzini, J. Babič, and A. Ajoudani, "A human-robot collaboration framework for improving ergonomics during dexterous operation of power tools," *Robotics and Computer-Integrated Manufacturing*, vol. 68, p. 102 084, 2021, ISSN: 0736-5845. DOI: https://doi.org/10.1016/j.rcim.2020.102084. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0736584520302945
Mentioned on page 8

[68] M. G. Carmichael and D. Liu, "Admittance control scheme for implementing model-based assistance-as-needed on a robot," in *2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2013, pp. 870–873
Mentioned on pages 8, 10, 87, 89, 94, 99

[69] M. G. Carmichael and D. Liu, "Towards using musculoskeletal models for intelligent control of physically assistive robots," en, in *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Boston, MA, Aug. 2011, pp. 8162–8165, ISBN: 978-1-4577-1589-1 978-1-4244-4121-1 978-1-4244-4122-8. DOI: 10.1109/IEMBS.2011.6092013. [Online]. Available: http://ieeexplore.ieee.org/document/6092013/ (visited on 10/14/2020)
Mentioned on pages 8, 10, 70, 71

[70] T. Petrič, L. Peternel, J. Morimoto, and J. Babič, "Assistive arm-exoskeleton control based on human muscular manipulability," *Frontiers in Neurorobotics*, vol. 13, p. 30, 2019, ISSN: 1662-5218. DOI: 10.3389/fnbot.2019.00030. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnbot.2019.00030
Mentioned on pages 8, 27, 94, 95, 159

[71] K. Fukuda *et al.*, "Frequently asked questions in polyhedral computation," *ETH, Zurich, Switzerland*, vol. 85, 2004. [Online]. Available: https://www.cs.mcgill.ca/~fukuda/download/paper/polyfaq980826.ps.gz
Mentioned on pages 10, 35, 36, 154, 156

[72] C. J. Ong and E. Gilbert, "The gilbert-johnson-keerthi distance algorithm: A fast version for incremental motions," in *Proceedings of International Conference on Robotics and Automation*, vol. 2, 1997, 1183–1189 vol.2. DOI: 10.1109/ROBOT.1997.614298
Mentioned on page 10

[73] J. Lawrence, "Polytope volume computation," *Mathematics of Computation*, vol. 57, no. 195, pp. 259–271, Jul. 1991. DOI: 10.1090/S0025-5718-1991-1079024-2
Mentioned on page 10

[74] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, "The quickhull algorithm for convex hulls," *ACM Trans. Math. Softw.*, vol. 22, no. 4, pp. 469–483, Dec. 1996, ISSN: 0098-3500. DOI: 10.1145/235815.235821. [Online]. Available: https://doi.org/10.1145/235815.235821
Mentioned on pages 10, 52, 54, 120, 172

[75] H. Barki, F. Denis, and F. Dupont, "Contributing vertices-based minkowski sum computation of convex polyhedra," *Computer-Aided Design*, vol. 41, no. 7, pp. 525–538, 2009, ISSN: 0010-4485. DOI: https://doi.org/10.1016/j.cad.2009.03.008. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010448509000724
MENTIONED ON PAGE 10

[76] H. R. Tiwary, "On the hardness of computing intersection, union and minkowski sum of polytopes," *Discrete & Computational Geometry*, vol. 40, no. 3, pp. 469–479, Oct. 2008, ISSN: 1432-0444. DOI: 10.1007/s00454-008-9097-3. [Online]. Available: https://doi.org/10.1007/s00454-008-9097-3
MENTIONED ON PAGE 10

[77] N. D. Botkin and V. L. Turova-Botkina, "An algorithm for finding the chebyshev center of a convex polyhedron," *Applied Mathematics and Optimization*, vol. 29, no. 2, pp. 211–222, Mar. 1994, ISSN: 1432-0606. DOI: 10.1007/BF01204183. [Online]. Available: https://doi.org/10.1007/BF01204183
MENTIONED ON PAGE 10

[78] A. Bemporad, C. Filippi, and F. D. Torrisi, "Inner and outer approximations of polytopes using boxes," *Computational Geometry*, vol. 27, no. 2, pp. 151–178, 2004, ISSN: 0925-7721. DOI: https://doi.org/10.1016/S0925-7721(03)00048-8. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925772103000488
MENTIONED ON PAGE 10

[79] M. Zolotas, M. Wonsick, P. Long, and T. Padır, "Motion polytopes in virtual reality for shared control in remote manipulation applications," *Frontiers in Robotics and AI*, vol. 8, 2021, ISSN: 2296-9144. DOI: 10.3389/frobt.2021.730433. [Online]. Available: https://www.frontiersin.org/articles/10.3389/frobt.2021.730433
MENTIONED ON PAGES 10, 17, 101, 122, 170

[80] V. Weistroffer, F. Keith, A. Bisiaux, C. Andriot, and A. Lasnier, "Using physics-based digital twins and extended reality for the safety and ergonomics evaluation of cobotic workstations," *Frontiers in Virtual Reality*, vol. 3, 2022, ISSN: 2673-4192. DOI: 10.3389/frvir.2022.781830. [Online]. Available: https://www.frontiersin.org/articles/10.3389/frvir.2022.781830
MENTIONED ON PAGES 10, 101

[81] P. T. Boggs and J. W. Tolle, "Sequential Quadratic Programming," *Acta Numerica*, vol. 4, pp. 1–51, 1995. DOI: 10.1017/S0962492900002518
MENTIONED ON PAGE 10

[82] D. Goldfarb and M. J. Todd, "Chapter II: Linear programming," in *Optimization*, ser. Handbooks in Operations Research and Management Science, vol. 1, Elsevier, 1989, pp. 73–170. DOI: https://doi.org/10.1016/S0927-0507(89)01003-0. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0927050789010030
MENTIONED ON PAGE 10

[83] G. Marani, J. Kim, J. Yuh, and W. K. Chung, "A real-time approach for singularity avoidance in resolved motion rate control of robotic manipulators," in *Proceedings IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, vol. 2, 2002, 1973–1978 vol.2. DOI: 10.1109/ROBOT.2002.1014830
MENTIONED ON PAGE 17

[84] J. Sverdrup-Thygeson, S. Moe, K. Y. Pettersen, and J. T. Gravdahl, "Kinematic singularity avoidance for robot manipulators using set-based manipulability tasks," in *IEEE*

Conference on Control Technology and Applications (CCTA), 2017, pp. 142–149. DOI: 10.1109/CCTA.2017.8062454
MENTIONED ON PAGE 17

[85] T. Pardi, V. Ortenzi, C. Fairbairn, T. Pipe, A. M. G. Esfahani, and R. Stolkin, "Planning maximum-manipulability cutting paths," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1999–2006, 2020. DOI: 10.1109/LRA.2020.2970949
MENTIONED ON PAGE 17

[86] K. Nagatani, T. Hirayama, A. Gofuku, and Y. Tanaka, "Motion planning for mobile manipulator with keeping manipulability," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 2, 2002, 1663–1668 vol.2. DOI: 10.1109/IRDS.2002.1043994
MENTIONED ON PAGE 17

[87] F. Chen, M. Selvaggio, and D. G. Caldwell, "Dexterous grasping by manipulability selection for mobile manipulator with visual guidance," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 2, pp. 1202–1210, 2019. DOI: 10.1109/TII.2018.2879426
MENTIONED ON PAGE 17

[88] R. Xu, J. Luo, and M. Wang, "Optimal grasping pose for dual-arm space robot cooperative manipulation based on global manipulability," *Acta Astronautica*, vol. 183, pp. 300–309, 2021, ISSN: 0094-5765. DOI: https://doi.org/10.1016/j.actaastro.2021.03.021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S009457652100134X
MENTIONED ON PAGE 17

[89] L. Grandguillaume, S. Lavernhe, and C. Tournier, "Optimal tool orientation in 3 + 2-axis machining considering machine kinematics," *The International Journal of Advanced Manufacturing Technology*, vol. 115, no. 9, pp. 2765–2783, Aug. 2021, ISSN: 1433-3015. DOI: 10.1007/s00170-021-07036-z. [Online]. Available: https://doi.org/10.1007/s00170-021-07036-z
MENTIONED ON PAGE 18

[90] Z. Zou, X. Pang, and J. Chen, "Comprehensive theoretical digging performance analysis for hydraulic excavator using convex polytope method," *Multibody System Dynamics*, vol. 47, no. 2, pp. 137–164, Oct. 2019, ISSN: 1573-272X. DOI: 10.1007/s11044-019-09686-0. [Online]. Available: https://doi.org/10.1007/s11044-019-09686-0
MENTIONED ON PAGE 19

[91] H. Ferrolho, W. Merkt, C. Tiseo, and S. Vijayakumar, "Comparing metrics for robustness against external perturbations in dynamic trajectory optimization," *arXiv preprint arXiv:1908.05380*, 2019
MENTIONED ON PAGE 19

[92] R. Orsolino, M. Focchi, C. Mastalli, H. Dai, D. G. Caldwell, and C. Semini, "Application of wrench-based feasibility analysis to the online trajectory optimization of legged robots," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3363–3370, 2018. DOI: 10.1109/LRA.2018.2836441
MENTIONED ON PAGE 19

[93] M. T. Rosenstein and R. A. Grupen, "Velocity-dependent dynamic manipulability," in *Proceedings IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, vol. 3, 2002, pp. 2424–2429
MENTIONED ON PAGE 20

[94] M. E. Jang and J. Lee, "Grasp stability analysis based on acceleration convex polytopes for multi-fingered robot hands," en, *International Journal of Control, Automation and Systems*, vol. 7, no. 2, pp. 253–265, Apr. 2009, ISSN: 1598-6446. DOI: 10.1007/s12555-009-0211-y. [Online]. Available: https://doi.org/10.1007/s12555-009-0211-y (visited on 10/22/2020)
MENTIONED ON PAGE 20

[95] S.-H. Jeong and T. Takahashi, "Optimal braking for impact force reduction using the dynamics of redundant manipulators," in *Proceedings IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, 2006, pp. 1898–1903
MENTIONED ON PAGE 20

[96] J. K. Salisbury, "Active stiffness control of a manipulator in cartesian coordinates," in *1980 19th IEEE Conference on Decision and Control including the Symposium on Adaptive Processes*, 1980, pp. 95–100. DOI: 10.1109/CDC.1980.272026
MENTIONED ON PAGES 20, 27

[97] A. Ajoudani, N. G. Tsagarakis, and A. Bicchi, "Choosing poses for force and stiffness control," *IEEE Transactions on Robotics*, vol. 33, no. 6, pp. 1483–1490, 2017
MENTIONED ON PAGES 21, 27

[98] A. V. Hill, "The heat of shortening and the dynamic constants of muscle," *Proceedings of the Royal Society of London. Series B-Biological Sciences*, vol. 126, no. 843, pp. 136–195, 1938
MENTIONED ON PAGE 22

[99] F. Zajac, "Muscle and tendon: Properties, models, scaling, and application to biomechanics and motor control," *Critical reviews in biomedical engineering*, vol. 17, no. 4, pp. 359–411, 1989, ISSN: 0278-940X. [Online]. Available: http://europepmc.org/abstract/MED/2676342
MENTIONED ON PAGE 22

[100] M. G. Pandy, "Moment arm of a muscle force," *Exercise and sport sciences reviews*, vol. 27, no. 1, pp. 79–118, 1999
MENTIONED ON PAGE 22

[101] J. M. SOUCIE *et al.*, "Range of motion measurements: Reference values and a database for comparison studies," *Haemophilia*, vol. 17, no. 3, pp. 500–507, 2011. DOI: https://doi.org/10.1111/j.1365-2516.2010.02399.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1365-2516.2010.02399.x. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1365-2516.2010.02399.x
MENTIONED ON PAGE 23

[102] P. M. Kane, B. G. Vopat, C. Got, K. Mansuripur, and E. Akelman, "The effect of supination and pronation on wrist range of motion," en, *J Wrist Surg*, vol. 3, no. 3, pp. 187–191, Aug. 2014
MENTIONED ON PAGE 23

[103] M. Grimmer, A. A. Elshamanhory, and P. Beckerle, "Human lower limb joint biomechanics in daily life activities: A literature based requirement analysis for anthropomorphic robot design," *Frontiers in Robotics and AI*, vol. 7, 2020, ISSN: 2296-9144. DOI: 10.3389/frobt.2020.00013. [Online]. Available: https://www.frontiersin.org/articles/10.3389/frobt.2020.00013
MENTIONED ON PAGE 23

[104] Y. Wang, C. Song, T. Zheng, D. Lau, K. Yang, and G. Yang, "Cable routing design and performance evaluation for multi-link cable-driven robots with minimal number of actuating cables," *IEEE Access*, vol. 7, pp. 135 790–135 800, 2019. DOI: 10.1109/ACCESS.2019.2924982
MENTIONED ON PAGE 23

[105] D. Lau, D. Oetomo, and S. K. Halgamuge, "Generalized modeling of multilink cable-driven manipulators with arbitrary routing using the cable-routing matrix," *IEEE Transactions on Robotics*, vol. 29, no. 5, pp. 1102–1113, 2013. DOI: 10.1109/TRO.2013.2264866
MENTIONED ON PAGE 23

[106] M. Sasaki, "Vertex Search Algorithm of Convex Polyhedron Representing Upper Limb Manipulation Ability," en, *Search Algorithms and Applications*, p. 13, 2011
MENTIONED ON PAGES 24, 49, 56, 58, 59, 63–65, 74

[107] V. Hernandez, N. Rezzoug, and P. Gorce, "Toward isometric force capabilities evaluation by using a musculoskeletal model: Comparison with direct force measurement," en, *Journal of Biomechanics*, vol. 48, no. 12, pp. 3178–3184, Sep. 2015, ISSN: 0021-9290. DOI: 10.1016/j.jbiomech.2015.07.003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0021929015003802 (visited on 10/14/2020)
MENTIONED ON PAGE 25

[108] M. G. Carmichael and D. Liu, "Towards using musculoskeletal models for intelligent control of physically assistive robots," in *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 2011, pp. 8162–8165. DOI: 10.1109/IEMBS.2011.6092013
MENTIONED ON PAGES 25, 55, 72

[109] X. Sheng, L. Tang, X. Huang, L. Zhu, X. Zhu, and G. Gu, "Operational-space wrench and acceleration capability analysis for multi-link cable-driven robots," *Science China Technological Sciences*, vol. 63, no. 10, pp. 2063–2072, 2020
MENTIONED ON PAGES 25, 26

[110] V. Muralidharan, P. Wenger, and C. Chevallereau, "Kinematic and Static Analysis of a Cable-Driven 2-X Tensegrity Manipulator for Two Actuation Strategies," in *Advances in Robot Kinematics 2022*, O. Altuzarra and A. Kecskeméthy, Eds., Cham: Springer International Publishing, 2022, pp. 149–159, ISBN: 978-3-031-08140-8
MENTIONED ON PAGES 25, 27, 101

[111] S. L. Chiu, "Task compatibility of manipulator postures," *The International Journal of Robotics Research*, vol. 7, no. 5, pp. 13–21, 1988. DOI: 10.1177/027836498800700502. eprint: https://doi.org/10.1177/027836498800700502. [Online]. Available: https://doi.org/10.1177/027836498800700502
MENTIONED ON PAGE 27

[112] Y. Shen, B. P.-Y. Hsiao, J. Ma, and J. Rosen, "Upper limb redundancy resolution under gravitational loading conditions: Arm postural stability index based on dynamic manipulability analysis," in *IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, 2017, pp. 332–338. DOI: 10.1109/HUMANOIDS.2017.8246894
MENTIONED ON PAGE 27

[113] M. L. Latash and V. M. Zatsiorsky, "Joint stiffness: Myth or reality?" *Human Movement Science*, vol. 12, no. 6, pp. 653–692, 1993, ISSN: 0167-9457. DOI: https://doi.org/10.1016/0167-9457(93)90010-M. [Online]. Available: https://www.sciencedirect.com/science/article/pii/016794579390010M
MENTIONED ON PAGE 27

[114] A. Ajoudani, C. Fang, N. Tsagarakis, and A. Bicchi, "Reduced-complexity representation of the human arm active endpoint stiffness for supervisory control of remote manipulation," *The International Journal of Robotics Research*, vol. 37, no. 1, pp. 155–167, 2018. DOI: 10.1177/0278364917744035. eprint: https://doi.org/10.1177/0278364917744035. [Online]. Available: https://doi.org/10.1177/0278364917744035

MENTIONED ON PAGE 27

[115] J. M. Inouye and F. J. Valero-Cuevas, "Muscle synergies heavily influence the neural control of arm endpoint stiffness and energy consumption," *PLOS Computational Biology*, vol. 12, no. 2, pp. 1–24, Feb. 2016. DOI: 10.1371/journal.pcbi.1004737. [Online]. Available: https://doi.org/10.1371/journal.pcbi.1004737
MENTIONED ON PAGES 27, 28

[116] V. Ramadoss, K. Sagar, M. S. Ikbal, D. Zlatanov, and M. Zoppi, "Modeling and stiffness evaluation of tendon-driven robot for collaborative human-robot interaction," in *IEEE International Conference on Intelligence and Safety for Robotics (ISR)*, 2021, pp. 233–238. DOI: 10.1109/ISR50024.2021.9419387
MENTIONED ON PAGE 28

[117] E. Lamon, A. De Franco, L. Peternel, and A. Ajoudani, "A capability-aware role allocation approach to industrial assembly tasks," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3378–3385, 2019. DOI: 10.1109/LRA.2019.2926963
MENTIONED ON PAGE 28

[118] D. Prattichizzo and J. C. Trinkle, "Grasping," en, in *Springer Handbook of Robotics*, ser. Springer Handbooks, B. Siciliano and O. Khatib, Eds., Cham: Springer International Publishing, 2016, pp. 955–988, ISBN: 978-3-319-32552-1. DOI: 10.1007/978-3-319-32552-1_16. [Online]. Available: https://doi.org/10.1007/978-3-319-32552-1_38 (visited on 10/24/2020)
MENTIONED ON PAGE 29

[119] M. Henk, J. Richter-Gebert, and G. M. Ziegler, "Basic properties of convex polytopes," in *Handbook of discrete and computational geometry*, Chapman and Hall/CRC, 2017, pp. 383–413
MENTIONED ON PAGE 36

[120] N. Kochdumper and M. Althoff, "Representation of polytopes as polynomial zonotopes," *arXiv: Combinatorics*, 2019
MENTIONED ON PAGE 36

[121] S. Sigl and M. Althoff, "M-Representation of Polytopes," *arXiv: Combinatorics*, 2023
MENTIONED ON PAGE 36

[122] D. Bremner, K. Fukuda, and A. Marzetta, "Primal—dual methods for vertex and facet enumeration," *Discrete & Computational Geometry*, vol. 20, no. 3, pp. 333–357, 1998. DOI: 10.1007/pl00009389
MENTIONED ON PAGES 37, 44, 45, 71

[123] T. Burger, P. Gritzmann, and V. Klee, "Polytope projection and projection polytopes," *The American Mathematical Monthly*, vol. 103, no. 9, pp. 742–755, 1996. DOI: 10.1080/00029890.1996.12004814. eprint: https://doi.org/10.1080/00029890.1996.12004814. [Online]. Available: https://doi.org/10.1080/00029890.1996.12004814
MENTIONED ON PAGE 38

[124] P. McMullen, "On zonotopes," *Transactions of the American Mathematical Society*, vol. 159, pp. 91–109, 1971, ISSN: 00029947. [Online]. Available: http://www.jstor.org/stable/1996000 (visited on 07/15/2023)
MENTIONED ON PAGE 39

[125] G. C. Shephard, "Combinatorial properties of associated zonotopes," *Canadian Journal of Mathematics*, vol. 26, no. 2, pp. 302–321, 1974. DOI: 10.4153/CJM-1974-032-5
MENTIONED ON PAGE 39

[126] C. Larson and M. Zandieh, "Three interpretations of the matrix equation Ax=b," *For the Learning of Mathematics*, vol. 33, no. 2, pp. 11–17, 2013, ISSN: 02280671. [Online]. Available: http://www.jstor.org/stable/43894844 (visited on 06/25/2023)
MENTIONED ON PAGES 39, 40

[127] C. F. Van Loan and G. Golub, "Matrix computations (johns hopkins studies in mathematical sciences)," *Matrix Computations*, vol. 5, 1996
MENTIONED ON PAGE 40

[128] G. Wang, Y. Wei, and S. Qiao, "Generalized inverses: Theory and computations," in (Developments in Mathematics), Developments in Mathematics. Springer Nature Singapore, 2018, ch. Chapter 1: Equation Solving Generalized Inverses, pp. 1–65, ISBN: 9789811301469. [Online]. Available: https://books.google.fr/books?id=P5ZaDwAAQBAJ

MENTIONED ON PAGES 40, 46

[129] K. Fukuda and A. Prodon, "Double description method revisited," in *Combinatorics and Computer Science*, M. Deza, R. Euler, and I. Manoussakis, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 91–111, ISBN: 978-3-540-70627-4
MENTIONED ON PAGES 44, 45, 56

[130] D. Avis and K. Fukuda, "A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra," *Discrete & Computational Geometry*, vol. 8, no. 3, pp. 295–313, Sep. 1992, ISSN: 1432-0444. DOI: 10.1007/BF02293050. [Online]. Available: https://doi.org/10.1007/BF02293050
MENTIONED ON PAGES 44, 45

[131] D. Avis and C. Jordan, "Comparative computational results for some vertex and facet enumeration codes," en, *arXiv:1510.02545 [cs]*, Jun. 2016. [Online]. Available: http://arxiv.org/abs/1510.02545 (visited on 09/29/2020)
MENTIONED ON PAGE 45

[132] M. E. Dyer, "The Complexity of Vertex Enumeration Methods," *Mathematics of Operations Research*, vol. 8, no. 3, pp. 381–402, 1983. DOI: 10.1287/moor.8.3.381. eprint: https://doi.org/10.1287/moor.8.3.381. [Online]. Available: https://doi.org/10.1287/moor.8.3.381
MENTIONED ON PAGES 45, 54

[133] K. Fukuda, "Lecture: Polyhedral computation, spring 2016," *Institute for Operations Research and Institute of Theoretical Computer Science, ETH Zurich. https://inf. ethz. ch/personal/fukudak/lect/pclect/notes2015/PolyComp2015. pdf*, 2016
MENTIONED ON PAGES 45, 47

[134] D. Avis, D. Bremner, and R. Seidel, "How good are convex hull algorithms?" *Computational Geometry*, vol. 7, no. 5, pp. 265–301, 1997, ISSN: 0925-7721. DOI: https://doi.org/10.1016/S0925-7721(96)00023-5. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925772196000235
MENTIONED ON PAGE 45

[135] T. S. Motzkin, H. Raiffa, G. L. Thompson, and R. M. Thrall, "3. the double description method," in *Contributions to the Theory of Games (AM-28), Volume II*, H. W. Kuhn and A. W. Tucker, Eds. Princeton: Princeton University Press, 1953, pp. 51–74, ISBN: 9781400881970. DOI: doi:10.1515/9781400881970-004. [Online]. Available: https://doi.org/10.1515/9781400881970-004
MENTIONED ON PAGE 45

[136] R. Seidel, "A convex hull algorithm optimal for point sets in even dimensions," Ph.D. dissertation, University of British Columbia, 1981. DOI: http://dx.doi.org/10.14288/1.0051821. [Online]. Available: https://open.library.ubc.ca/collections/ubctheses/831/items/1.0051821
MENTIONED ON PAGE 45

[137] R. Seidel, "Output-size sensitive algorithms for constructive problems in computational geometry," Ph.D. dissertation, Cornell University, USA, 1987. [Online]. Available: https://dl.acm.org/doi/10.5555/39974
MENTIONED ON PAGE 45

[138] S. Paulraj, C. Chellappan, and T. R. Natesan, "A heuristic approach for identification of redundant constraints in linear programming models," *International Journal of Computer Mathematics*, vol. 83, no. 8-9, pp. 675–683, 2006. DOI: 10.1080/00207160601014148. eprint: https://doi.org/10.1080/00207160601014148. [Online]. Available: https://doi.org/10.1080/00207160601014148
MENTIONED ON PAGES 47, 173

[139] J. Telgen, "Identifying redundant constraints and implicit equalities in systems of linear constraints," *Management Science*, vol. 29, no. 10, pp. 1209–1222, 1983. DOI: 10.1287/mnsc.29.10.1209. eprint: https://doi.org/10.1287/mnsc.29.10.1209. [Online]. Available: https://doi.org/10.1287/mnsc.29.10.1209
MENTIONED ON PAGE 47

[140] M. Gouttefarde, J. Lamaury, C. Reichert, and T. Bruckmann, "A Versatile Tension Distribution Algorithm for n-DOF Parallel Robots Driven by $n+2$ Cables," en, *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1444–1457, Dec. 2015, ISSN: 1552-3098, 1941-0468. DOI: 10.1109/TRO.2015.2495005. [Online]. Available: http://ieeexplore.ieee.org/document/7331330/ (visited on 10/22/2020)
MENTIONED ON PAGE 49

[141] G. B. Dantzig and B. C. Eaves, "Fourier-motzkin elimination and its dual," *Journal of Combinatorial Theory, Series A*, vol. 14, no. 3, pp. 288–297, 1973
MENTIONED ON PAGES 51, 56, 109

[142] R.-J. Jing, M. Moreno-Maza, and D. Talaashrafi, "Complexity estimates for fourier-motzkin elimination," in *Computer Algebra in Scientific Computing*, F. Boulier, M. England, T. M. Sadykov, and E. V. Vorozhtsov, Eds., Cham: Springer International Publishing, 2020, pp. 282–306, ISBN: 978-3-030-60026-6
MENTIONED ON PAGE 51

[143] D. Monniaux, "Quantifier elimination by lazy model enumeration," in *Computer Aided Verification*, T. Touili, B. Cook, and P. Jackson, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 585–599, ISBN: 978-3-642-14295-6
MENTIONED ON PAGE 51

[144] C. Jones, E. C. Kerrigan, and J. Maciejowski, "Equality set projection: A new algorithm for the projection of polytopes in halfspace representation," Cambridge University Engineering Dept, Tech. Rep., 2004
MENTIONED ON PAGES 51, 56, 109

[145] T. Glassle, D. Gross, and R. Chaves, "Computational tools for solving a marginal problem with applications in bell non-locality and causal modeling," *Journal of Physics A: Mathematical and Theoretical*, vol. 51, no. 48, p. 484 002, Nov. 2018. DOI: 10.1088/1751-8121/aae754. [Online]. Available: https://doi.org/10.1088%5C%2F1751-8121%5C%2Faae754
MENTIONED ON PAGES 51, 54

[146] S. Bouchard, C. Gosselin, and B. Moore, "On the Ability of a Cable-Driven Robot to Generate a Prescribed Set of Wrenches," *Journal of Mechanisms and Robotics*, vol. 2, no. 1, Dec. 2009, ISSN: 1942-4302. DOI: 10.1115/1.4000558. [Online]. Available: https://doi.org/10.1115/1.4000558
MENTIONED ON PAGES 53, 87

[147] M. Gouttefarde and S. Krut, "Characterization of parallel manipulator available wrench set facets," in *Advances in Robot Kinematics: Motion in Man and Machine*, J. Lenarcic and M. M. Stanisic, Eds., Dordrecht: Springer Netherlands, 2010, pp. 475–482, ISBN: 978-90-481-9262-5. DOI: 10.1007/978-90-481-9262-5_51
MENTIONED ON PAGES 53, 56, 71, 160

[148] P. K. Agarwal and J. Matoušek, "Ray shooting and parametric search," *SIAM Journal on Computing*, vol. 22, no. 4, pp. 794–806, 1993
MENTIONED ON PAGES 54, 67

[149] C. Lassez, "Quantifier elimination for conjunctions of linear constraints via a convex hull algorithm," *Symbolic and Numerical Computation for Artificial Intelligence*, pp. 103–122, 1992
MENTIONED ON PAGES 54, 66–68, 74

[150] T. Huynh, C. Lassez, and J. Lassez, "Practical issues on the projection of polyhedral sets," *Annals of Mathematics and Artificial Intelligence*, vol. 6, pp. 295–315, 2005
MENTIONED ON PAGE 54

[151] T. Bretl and S. Lall, "Testing static equilibrium for legged robots," *IEEE Transactions on Robotics*, vol. 24, no. 4, pp. 794–807, 2008. DOI: 10.1109/TRO.2008.2001360
MENTIONED ON PAGES 55, 70

[152] A. Del Prete, S. Tonneau, and N. Mansard, "Fast algorithms to test robust static equilibrium for legged robots," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1601–1607. DOI: 10.1109/ICRA.2016.7487299
MENTIONED ON PAGES 55, 68

[153] H. Audren and A. Kheddar, "3-d robust stability polyhedron in multicontact," *IEEE Transactions on Robotics*, vol. 34, no. 2, pp. 388–403, 2018. DOI: 10.1109/TRO.2017.2786683
MENTIONED ON PAGE 55

[154] J. Ponce and B. Faverjon, "On computing three-finger force-closure grasps of polygonal objects," *IEEE Transactions on Robotics and Automation*, vol. 11, no. 6, pp. 868–881, 1995. DOI: 10.1109/70.478433
MENTIONED ON PAGE 55

[155] W. Xu, J. Wang, and J. Sun, "A projection method for derivation of non-shannon-type information inequalities," in *IEEE International Symposium on Information Theory*, 2008, pp. 2116–2120. DOI: 10.1109/ISIT.2008.4595363
MENTIONED ON PAGE 55

[156] V. Klema and A. Laub, "The singular value decomposition: Its computation and some applications," *IEEE Transactions on Automatic Control*, vol. 25, no. 2, pp. 164–176, Apr. 1980, ISSN: 1558-2523. DOI: 10.1109/TAC.1980.1102314
MENTIONED ON PAGES 61, 68

[157] S. Vajda and S. I. Gass, "Linear programming. methods and applications," *Journal of the Operational Research Society*, vol. 15, no. 4, pp. 412–413, 1964. DOI: 10.2307/3007132
MENTIONED ON PAGES 67, 132

[158] K. R. Saul *et al.*, "Benchmarking of dynamic simulation predictions in two software platforms using an upper limb musculoskeletal model," *Computer methods in biomechanics and biomedical engineering*, vol. 18, no. 13, pp. 1445–1458, 2015
MENTIONED ON PAGES 72, 87, 92

[159] B. Michaud and M. Begon, "Biorbd: A c++, python and matlab library to analyze and simulate the human body biomechanics," *Journal of Open Source Software*, vol. 6, no. 57, p. 2562, 2021. DOI: 10.21105/joss.02562. [Online]. Available: https://doi.org/10.21105/joss.02562
MENTIONED ON PAGES 72, 92, 153, 162

[160] R. Bejarano, B. R. Ferrer, W. M. Mohammed, and J. L. Martinez Lastra, "Implementing a human-robot collaborative assembly workstation," in *IEEE 17th International Conference on Industrial Informatics (INDIN)*, vol. 1, 2019, pp. 557–564. DOI: 10.1109/INDIN41052.2019.8972158
MENTIONED ON PAGE 77

[161] H. Giberti, T. Abbattista, M. Carnevale, L. Giagu, and F. Cristini, "A methodology for flexible implementation of collaborative robots in smart manufacturing systems," *Robotics*, vol. 11, no. 1, 2022, ISSN: 2218-6581. DOI: 10.3390/robotics11010009. [Online]. Available: https://www.mdpi.com/2218-6581/11/1/9
MENTIONED ON PAGE 77

[162] L. Wang, "A futuristic perspective on human-centric assembly," *Journal of Manufacturing Systems*, vol. 62, pp. 199–201, 2022, ISSN: 0278-6125. DOI: https://doi.org/10.1016/j.jmsy.2021.11.001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0278612521002235
MENTIONED ON PAGE 77

[163] H. Arai, T. Takubo, Y. Hayashibara, and K. Tanie, "Human-robot cooperative manipulation using a virtual nonholonomic constraint," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 4, 2000, 4063–4069 vol.4. DOI: 10.1109/ROBOT.2000.845365
MENTIONED ON PAGE 78

[164] K. Kosuge and N. Kazamura, "Control of a robot handling an object in cooperation with a human," in *Proceedings 6th IEEE International Workshop on Robot and Human Communication. RO-MAN'97 SENDAI*, 1997, pp. 142–147. DOI: 10.1109/ROMAN.1997.646971
MENTIONED ON PAGE 78

[165] T. Tsumugiwa, R. Yokogawa, and K. Hara, "Variable impedance control with virtual stiffness for human-robot cooperative peg-in-hole task," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 2, 2002, 1075–1081 vol.2. DOI: 10.1109/IRDS.2002.1043874
MENTIONED ON PAGE 78

[166] J. Carpentier, N. Mansard, F. Valenza, J. Mirabel, G. Saurel, and R. Budhiraja, *Pinocchio - Efficient and versatile Rigid Body Dynamics algorithms*, version 2.6.3, Jul. 2021. [Online]. Available: https://github.com/stack-of-tasks/pinocchio
Mentioned on pages 83, 91, 120, 122, 144, 153, 161, 162

[167] M. Quigley *et al.*, "Ros: An open-source robot operating system," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Robotics*, Kobe, Japan, May 2009. [Online]. Available: https://www.ros.org
Mentioned on pages 83, 91, 113, 153, 154

[168] F. Ferraguti, R. Villa, C. Talignani Landi, A. Maria Zanchettin, P. Rocco, and C. Secchi, "A unified architecture for physical and ergonomic human–robot collaboration," *Robotica*, vol. 38, no. 4, pp. 669–683, 2020. DOI: 10.1017/S026357471900095X
Mentioned on page 86

[169] N. Mansfeld, B. Djellab, J. R. Veuthey, F. Beck, C. Ott, and S. Haddadin, "Improving the performance of biomechanically safe velocity control for redundant robots through reflected mass minimization," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 5390–5397. DOI: 10.1109/IROS.2017.8206435
Mentioned on pages 86, 126, 134

[170] P. Rani and N. Sarkar, "Operator engagement detection for robot behavior adaptation," *International Journal of Advanced Robotic Systems*, vol. 4, no. 1, p. 1, 2007. DOI: 10.5772/5716. eprint: https://doi.org/10.5772/5716. [Online]. Available: https://doi.org/10.5772/5716
Mentioned on page 86

[171] E. A. Byrne and R. Parasuraman, "Psychophysiology and adaptive automation," *Biological Psychology*, vol. 42, no. 3, pp. 249–268, 1996, ISSN: 0301-0511. DOI: https://doi.org/10.1016/0301-0511(95)05161-9. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0301051195051619
Mentioned on page 86

[172] C. Shoaf, A. Genaidy, W. Karwowski, T. Waters, and D. Christensen, "Comprehensive manual handling limits for lowering, pushing, pulling and carrying activities," *Ergonomics*, vol. 40, no. 11, pp. 1183–1200, 1997
Mentioned on page 87

[173] W. Kim, J. Lee, L. Peternel, N. Tsagarakis, and A. Ajoudani, "Anticipatory robot assistance for the prevention of human static joint overloading in human–robot collaboration," *IEEE Robotics and Automation Letters*, vol. 3, no. 1, pp. 68–75, 2018. DOI: 10.1109/LRA.2017.2729666
Mentioned on page 87

[174] M. G. Carmichael and D. Liu, "Experimental evaluation of a model-based assistance-as-needed paradigm using an assistive robot," in *2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2013, pp. 866–869. DOI: 10.1109/EMBC.2013.6609638
Mentioned on pages 87, 94

[175] K. R. Holzbaur, W. M. Murray, and S. L. Delp, "A model of the upper extremity for simulating musculoskeletal surgery and analyzing neuromuscular control," *Annals of biomedical engineering*, vol. 33, no. 6, pp. 829–840, 2005
Mentioned on pages 87, 92, 97, 98, 161

[176] S. L. Delp *et al.*, "Opensim: Open-source software to create and analyze dynamic simulations of movement," *IEEE Transactions on Biomedical Engineering*, vol. 54, no. 11, pp. 1940–1950, 2007. DOI: 10.1109/TBME.2007.901024
MENTIONED ON PAGES 87, 92, 153, 162

[177] M. H. Sohn, D. M. Smith, and L. H. Ting, "Effects of kinematic complexity and number of muscles on musculoskeletal model robustness to muscle dysfunction," *PLOS ONE*, vol. 14, no. 7, pp. 1–16, Jul. 2019. DOI: 10.1371/journal.pone.0219779. [Online]. Available: https://doi.org/10.1371/journal.pone.0219779
MENTIONED ON PAGES 87, 97, 169

[178] NaturalPoint-Inc., *Motion capture systems - optitrack*, 2023. [Online]. Available: https://optitrack.com/
MENTIONED ON PAGE 92

[179] C. C. Gordon, T. Churchill, C. E. Clauser, B. Bradtmiller, and J. T. McConville, "Anthropometric survey of us army personnel: Methods and summary statistics 1988," Anthropology Research Project Inc Yellow Springs OH, Tech. Rep., 1989
MENTIONED ON PAGE 92

[180] T. A. Correa and M. G. Pandy, "A mass–length scaling law for modeling muscle strength in the lower limb," *Journal of Biomechanics*, vol. 44, no. 16, pp. 2782–2789, 2011, ISSN: 0021-9290. DOI: https://doi.org/10.1016/j.jbiomech.2011.08.024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021929011005847
MENTIONED ON PAGES 92, 97

[181] S. Bolghanabadi, h. Dehghan, and a. Mehdi Pour, "The relationship between musculoskeletal disorders, stress and fatigue in the food industry employees," *Journal of Ergonomics*, vol. 2, no. 1, 2014. [Online]. Available: http://journal.iehfs.ir/article-1-55-en.html
MENTIONED ON PAGE 97

[182] G. Laisné, J.-M. Salotti, and N. Rezzoug, "Genetic algorithms for force polytopes prediction," in *48ème Congrès de la Société de Biomécanique*, Société de Biomécanique, Grenoble, France, Oct. 2023. [Online]. Available: https://inria.hal.science/hal-04151258
MENTIONED ON PAGES 97, 155, 169

[183] G. El-Ghazaly, M. Gouttefarde, V. Creuze, and F. Pierrot, "Maximum wrench feasible payload in cable-driven parallel robots equipped with a serial robot," in *IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, 2016, pp. 1572–1578. DOI: 10.1109/AIM.2016.7576994
MENTIONED ON PAGE 101

[184] T. Rasheed, P. Long, D. Marquez-Gamez, and S. Caro, "Available wrench set for planar mobile cable-driven parallel robots," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 962–967. DOI: 10.1109/ICRA.2018.8461199
MENTIONED ON PAGE 101

[185] M. R. Endsley, "Toward a theory of situation awareness in dynamic systems," *Human Factors*, vol. 37, no. 1, pp. 32–64, 1995. DOI: 10.1518/001872095779049543. eprint: https://doi.org/10.1518/001872095779049543. [Online]. Available: https://doi.org/10.1518/001872095779049543
MENTIONED ON PAGE 101

[186] B. Camblor, N. Benhabib, D. Daney, V. Padois, and J.-M. Salotti, "Task-consistent signaling motions for improved understanding in human-robot interaction and workspace sharing," in *2022 17th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2022, pp. 275–283. DOI: 10.1109/HRI53351.2022.9889575
MENTIONED ON PAGES 101, 170

[187] B. Camblor, J.-M. Salotti, C. Fage, and D. Daney, "Degraded situation awareness in a robotic workspace: Accident report analysis," *Theoretical Issues in Ergonomics Science*, vol. 23, no. 1, pp. 60–79, 2022. DOI: 10.1080/1463922X.2021.1879308. eprint: https://doi.org/10.1080/1463922X.2021.1879308. [Online]. Available: https://doi.org/10.1080/1463922X.2021.1879308
MENTIONED ON PAGE 101

[188] R. Suzuki, A. Karim, T. Xia, H. Hedayati, and N. Marquardt, "Augmented reality and robotics: A survey and taxonomy for ar-enhanced human-robot interaction and robotic interfaces," in *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, ser. CHI '22, New Orleans, LA, USA: Association for Computing Machinery, 2022, ISBN: 9781450391573. DOI: 10.1145/3491102.3517719. [Online]. Available: https://doi.org/10.1145/3491102.3517719
MENTIONED ON PAGE 101

[189] D. B. Van de Merwe, L. Van Maanen, F. B. Ter Haar, R. J. E. Van Dijk, N. Hoeba, and N. Van der Stap, "Human-robot interaction during virtual reality mediated teleoperation: How environment information affects spatial task performance and operator situation awareness," in *Virtual, Augmented and Mixed Reality. Applications and Case Studies*, J. Y. Chen and G. Fragomeni, Eds., Cham: Springer International Publishing, 2019, pp. 163–177, ISBN: 978-3-030-21565-1
MENTIONED ON PAGE 101

[190] *Universal robots e-series user manual*, English, version 5.7, Universal Robots A/S, 2020, 254 pp. [Online]. Available: https://www.universal-robots.com/download/, pre-published
MENTIONED ON PAGES 102, 129

[191] *Product manual franka emika robot: Instruction handbook*, English, Franka Emika GmbH, Munich, Germany, Oct. 2021, 187 pp. [Online]. Available: https://www.franka.de/documents/, pre-published
MENTIONED ON PAGES 102, 103, 113, 121

[192] M. Gouttefarde, D. Daney, and J.-P. Merlet, "Interval-analysis-based determination of the wrench-feasible workspace of parallel cable-driven robots," *IEEE Transactions on Robotics*, vol. 27, no. 1, pp. 1–13, 2011. DOI: 10.1109/TRO.2010.2090064
MENTIONED ON PAGES 102, 117

[193] M. Althoff, "Reachability analysis and its application to the safety assessment of autonomous cars," Ph.D. dissertation, Technische Universität München, 2010
MENTIONED ON PAGE 103

[194] M. Althoff and B. H. Krogh, "Reachability analysis of nonlinear differential-algebraic systems," *IEEE Transactions on Automatic Control*, vol. 59, no. 2, pp. 371–383, 2014. DOI: 10.1109/TAC.2013.2285751
MENTIONED ON PAGE 103

[195] M. Althoff, G. Frehse, and A. Girard, "Set propagation techniques for reachability analysis," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, no. 1,

pp. 369–395, 2021. DOI: 10.1146/annurev-control-071420-081941. [Online]. Available: https://doi.org/10.1146/annurev-control-071420-081941
MENTIONED ON PAGE 103

[196] A. Pereira and M. Althoff, "Calculating human reachable occupancy for guaranteed collision-free planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 4473–4480. DOI: 10.1109/IROS.2017.8206314
MENTIONED ON PAGES 103, 168

[197] S. R. Schepp, J. Thumm, S. B. Liu, and M. Althoff, "Sara: A tool for safe human-robot coexistence and collaboration through reachability analysis," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 4312–4317. DOI: 10.1109/ICRA46639.2022.9811952
MENTIONED ON PAGES 103, 168

[198] M. Hamad, N. Mansfeld, S. Abdolshah, and S. Haddadin, "The role of robot payload in the safety map framework," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 195–200. DOI: 10.1109/IROS40897.2019.8968022
MENTIONED ON PAGE 106

[199] *Cobots: Robots for Collaboration With Human Operators*, vol. Dynamic Systems and Control, ASME International Mechanical Engineering Congress and Exposition, Nov. 1996, pp. 433–439. DOI: 10.1115/IMECE1996-0367. [Online]. Available: https://doi.org/10.1115/IMECE1996-0367
MENTIONED ON PAGE 107

[200] A. Bowling and O. Khatib, "The dynamic capability equations: A new tool for analyzing robotic manipulator performance," *IEEE Transactions on Robotics*, vol. 21, no. 1, pp. 115–123, 2005. DOI: 10.1109/TRO.2004.837243
MENTIONED ON PAGE 112

[201] P. Corke and J. Haviland, "Not your grandmother's toolbox–the robotics toolbox reinvented for python," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 11 357–11 363. [Online]. Available: https://github.com/petercorke/robotics-toolbox-python
MENTIONED ON PAGES 113, 153, 154, 162

[202] B. Kouvaritakis and M. Cannon, "Model predictive control," in 1st ed. Springer International Publishing, 2016, ch. Chapter 2: MPC with No Model Uncertainty. DOI: 10.1007/978-3-319-24853-0. [Online]. Available: https://doi.org/10.1007/978-3-319-24853-0
MENTIONED ON PAGES 117, 127, 150

[203] J.-P. Merlet, C. M. Gosselin, and N. Mouly, "Workspaces of planar parallel manipulators," *Mechanism and Machine Theory*, vol. 33, no. 1, pp. 7–20, 1998, ISSN: 0094-114X. DOI: https://doi.org/10.1016/S0094-114X(97)00025-6. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0094114X97000256
MENTIONED ON PAGE 119

[204] S. H. Lo, "Delaunay triangulation of non-convex planar domains," *International Journal for Numerical Methods in Engineering*, vol. 28, no. 11, pp. 2695–2707, 1989. DOI: 10.1002/nme.1620281113. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.1620281113
MENTIONED ON PAGE 120

[205] H. Edelsbrunner and E. P. Mücke, "Three-dimensional alpha shapes," *ACM Trans. Graph.*, vol. 13, no. 1, pp. 43–72, Jan. 1994, ISSN: 0730-0301. DOI: 10.1145/174462.156635. [Online]. Available: https://doi.org/10.1145/174462.156635
MENTIONED ON PAGE 120

[206] T. K. F. Da, S. Loriot, and M. Yvinec, "3D alpha shapes," in *CGAL User and Reference Manual*, 5.6, CGAL Editorial Board, 2023. [Online]. Available: https://doc.cgal.org/5.6/Manual/packages.html%5C#PkgAlphaShapes3
MENTIONED ON PAGE 120

[207] A. Ajoudani, A. M. Zanchettin, S. Ivaldi, A. Albu-Schäffer, K. Kosuge, and O. Khatib, "Progress and prospects of the human–robot collaboration," *Autonomous Robots*, vol. 42, no. 5, pp. 957–975, 2018
MENTIONED ON PAGE 126

[208] G. Pardo-Castellote and R. Cannon, "Proximate time-optimal algorithm for on-line path parameterization and modification," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 2, 1996, 1539–1546 vol.2. DOI: 10.1109/ROBOT.1996.506923
MENTIONED ON PAGE 126

[209] A. Gasparetto, P. Boscariol, A. Lanzutti, and R. Vidoni, "Trajectory planning in robotics," *Mathematics in Computer Science*, vol. 6, no. 3, pp. 269–279, Sep. 2012, ISSN: 1661-8289. DOI: 10.1007/s11786-012-0123-8. [Online]. Available: https://doi.org/10.1007/s11786-012-0123-8
MENTIONED ON PAGES 126, 135

[210] J. E. Bobrow, S. Dubowsky, and J. S. Gibson, "Time-optimal control of robotic manipulators along specified paths," *The international journal of robotics research*, vol. 4, no. 3, pp. 3–17, 1985
MENTIONED ON PAGE 126

[211] H. Pham and Q.-C. Pham, "A new approach to time-optimal path parameterization based on reachability analysis," *IEEE Transactions on Robotics*, vol. 34, no. 3, pp. 645–659, 2018. DOI: 10.1109/TRO.2018.2819195
MENTIONED ON PAGES 126, 128, 130, 144, 145, 152

[212] S. Macfarlane and E. Croft, "Jerk-bounded manipulator trajectory planning: Design for real-time applications," *IEEE Transactions on Robotics and Automation*, vol. 19, no. 1, pp. 42–52, 2003. DOI: 10.1109/TRA.2002.807548
MENTIONED ON PAGE 126

[213] R. Haschke, E. Weitnauer, and H. Ritter, "On-line planning of time-optimal, jerk-limited trajectories," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008, pp. 3248–3253
MENTIONED ON PAGE 126

[214] P. Svarny, M. Hamad, A. Kurdas, M. Hoffmann, S. Abdolshah, and S. Haddadin, "Functional mode switching for safe and efficient human-robot interaction," in *IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*, 2022, pp. 888–894. DOI: 10.1109/Humanoids53995.2022.10000070
MENTIONED ON PAGE 126

[215] Y. Fang, J. Hu, W. Liu, Q. Shao, J. Qi, and Y. Peng, "Smooth and time-optimal s-curve trajectory planning for automated robots and machines," *Mechanism and Machine Theory*, vol. 137, pp. 127–153, 2019, ISSN: 0094-114X. DOI: https://doi.org/10.1016/

j.mechmachtheory.2019.03.019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0094114X18317890
Mentioned on page 126

[216] K. M. Lynch and F. C. Park, *Modern Robotics: Mechanics, Planning, and Control*, 1st. USA: Cambridge University Press, 2017, ISBN: 1107156300
Mentioned on pages 126, 127, 135, 136

[217] A. Palleschi, M. Hamad, S. Abdolshah, M. Garabini, S. Haddadin, and L. Pallottino, "Fast and safe trajectory planning: Solving the cobot performance/safety trade-off in human-robot shared environments," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5445–5452, 2021. DOI: 10.1109/LRA.2021.3076968
Mentioned on page 126

[218] K. Springer, H. Gattringer, and C. Stöger, "A real-time nearly time-optimal point-to-point trajectory planning method using dynamic movement primitives," in *2014 23rd International Conference on Robotics in Alpe-Adria-Danube Region (RAAD)*, 2014, pp. 1–6. DOI: 10.1109/RAAD.2014.7002244
Mentioned on page 127

[219] S. Zhang, A. M. Zanchettin, R. Villa, and S. Dai, "Real-time trajectory planning based on joint-decoupled optimization in human-robot interaction," *Mechanism and Machine Theory*, vol. 144, p. 103 664, 2020, ISSN: 0094-114X. DOI: https://doi.org/10.1016/j.mechmachtheory.2019.103664. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0094114X19313849
Mentioned on page 127

[220] N. Torres Alberto, A. Skuric, L. Joseph, V. Padois, and D. Daney, "Linear Model Predictive Control in SE(3) for online trajectory planning in dynamic workspaces," Sep. 2022. [Online]. Available: https://hal.science/hal-03790059
Mentioned on page 127

[221] M. Eckhoff, R. J. Kirschner, E. Kern, S. Abdolshah, and S. Haddadin, "An mpc framework for planning safe & trustworthy robot motions," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 4737–4742. DOI: 10.1109/ICRA46639.2022.9812160
Mentioned on page 127

[222] S. Kleff, A. Meduri, R. Budhiraja, N. Mansard, and L. Righetti, "High-frequency nonlinear model predictive control of a manipulator," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 7330–7336. DOI: 10.1109/ICRA48506.2021.9560990
Mentioned on page 127

[223] M. Massaro, S. Lovato, M. Bottin, and G. Rosati, "An optimal control approach to the minimum-time trajectory planning of robotic manipulators," *Robotics*, vol. 12, no. 3, 2023, ISSN: 2218-6581. DOI: 10.3390/robotics12030064. [Online]. Available: https://www.mdpi.com/2218-6581/12/3/64
Mentioned on page 127

[224] A. D. Dragan, K. C. Lee, and S. S. Srinivasa, "Legibility and predictability of robot motion," in *2013 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2013, pp. 301–308. DOI: 10.1109/HRI.2013.6483603
Mentioned on page 128

[225] Franka Emika, *Panda robot datasheet - franka control interface*, 2022. [Online]. Available: https://frankaemika.github.io/docs/control%5C_parameters.html
Mentioned on pages 129, 142, 146

[226] C. D. Meyer, *Matrix Analysis and Applied Linear Algebra*. SIAM, 2001. [Online]. Available: http://www.matrixanalysis.com/
Mentioned on page 132

[227] L. Joseph, J. K. Pickard, V. Padois, and D. Daney, "Online velocity constraint adaptation for safe and efficient human-robot workspace sharing," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 11 045–11 051. DOI: 10.1109/IROS45743.2020.9340961
Mentioned on page 134

[228] L. Moriello, L. Biagiotti, C. Melchiorri, and A. Paoli, "Manipulating liquids with robots: A sloshing-free solution," *Control Engineering Practice*, vol. 78, pp. 129–141, 2018, ISSN: 0967-0661. DOI: https://doi.org/10.1016/j.conengprac.2018.06.018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0967066118302053
Mentioned on page 134

[229] K. D. Nguyen, T.-C. Ng, and I.-M. Chen, "On algorithms for planning s-curve motion profiles," *International Journal of Advanced Robotic Systems*, vol. 5, no. 1, p. 11, 2008. DOI: 10.5772/5652. eprint: https://doi.org/10.5772/5652. [Online]. Available: https://doi.org/10.5772/5652
Mentioned on page 135

[230] L. Berscheid and T. Kröger, "Jerk-limited real-time trajectory generation with arbitrary target states," *Robotics: Science and Systems XVII*, 2021
Mentioned on pages 135, 144, 146

[231] D. Constantinescu and E. A. Croft, "Smooth and time-optimal trajectory planning for industrial manipulators along specified paths," *Journal of Robotic Systems*, vol. 17, no. 5, pp. 233–249, 2000. DOI: https://doi.org/10.1002/(SICI)1097-4563(200005)17:5<233::AID-ROB1>3.0.CO;2-Y
Mentioned on page 135

[232] F. Pfeiffer and R. Johanni, "A concept for manipulator trajectory planning," *IEEE Journal on Robotics and Automation*, vol. 3, no. 2, pp. 115–123, 1987. DOI: 10.1109/JRA.1987.1087090
Mentioned on page 135

[233] J. Bobrow, S. Dubowsky, and J. Gibson, "Time-optimal control of robotic manipulators along specified paths," *The International Journal of Robotics Research*, vol. 4, no. 3, pp. 3–17, 1985. DOI: 10.1177/027836498500400301. eprint: https://doi.org/10.1177/027836498500400301. [Online]. Available: https://doi.org/10.1177/027836498500400301
Mentioned on page 136

[234] A. D. Prete, "Joint position and velocity bounds in discrete-time acceleration/torque control of robot manipulators," *IEEE Robotics and Automation Letters*, vol. 3, no. 1, pp. 281–288, 2018. DOI: 10.1109/LRA.2017.2738321
Mentioned on page 143

[235] A. O. Makhorin, *Glpk - gnu linear programming kit, version 4.0*, 2022. [Online]. Available: http://www.gnu.org/software/glpk/ (visited on 09/10/2022)
Mentioned on page 144

[236] M. Koskinopoulou, F. Raptopoulos, G. Papadopoulos, N. Mavrakis, and M. Maniadakis, "Robotic waste sorting technology: Toward a vision-based categorization system for the industrial robotic separation of recyclable waste," *IEEE Robotics & Automation Magazine*, vol. 28, no. 2, pp. 50–60, 2021. DOI: 10.1109/MRA.2021.3066040
MENTIONED ON PAGE 148

[237] G. Hassan *et al.*, "Time-optimal pick-and-throw s-curve trajectories for fast parallel robots," *IEEE/ASME Transactions on Mechatronics*, vol. 27, no. 6, pp. 4707–4717, 2022. DOI: 10.1109/TMECH.2022.3164247
MENTIONED ON PAGE 148

[238] R. Sarc, A. Curtis, L. Kandlbauer, K. Khodier, K. Lorber, and R. Pomberger, "Digitalisation and intelligent robotics in value chain of circular economy oriented waste management – a review," *Waste Management*, vol. 95, pp. 476–492, 2019, ISSN: 0956-053X. DOI: https://doi.org/10.1016/j.wasman.2019.06.035. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0956053X19304234
MENTIONED ON PAGE 148

[239] J. Haviland and P. Corke, "Maximising manipulability during resolved-rate motion control," *CoRR*, vol. abs/2002.11901, 2020. arXiv: 2002.11901. [Online]. Available: https://arxiv.org/abs/2002.11901
MENTIONED ON PAGE 154

[240] M. Prada, *Manipulability metrics: Ros-based library for computing manipulability metrics*, https://github.com/tecnalia-medical-robotics/manipulability_metrics, 2023
MENTIONED ON PAGE 154

[241] N. Jaquier, *Manipulability: Repository containing the codes for manipulability learning, tracking and transfer*. https://github.com/NoemieJaquier/Manipulability, 2023
MENTIONED ON PAGE 154

[242] N. Jaquier, L. Rozo, D. G. Caldwell, and S. Calinon, "Geometry-aware manipulability learning, tracking and transfer," *Intl. Journal of Robotics Research"*, vol. 20, no. 2-3, pp. 624–650, 2021
MENTIONED ON PAGE 154

[243] S. Caron, *Pypoman: Polyhedron manipulation in python*, https://github.com/stephane-caron/pypoman, 2023
MENTIONED ON PAGE 154

[244] M. Herceg, M. Kvasnica, C. N. Jones, and M. Morari, "Multi-parametric toolbox 3.0," in *2013 European Control Conference (ECC)*, 2013, pp. 502–510. DOI: 10.23919/ECC.2013.6669862. [Online]. Available: https://www.mpt3.org/
MENTIONED ON PAGE 154

[245] E. Zurich, *Cddlib: An efficient implementation of the double description method*, https://github.com/cddlib, 2023
MENTIONED ON PAGE 154

[246] K. Fukuda, "Cdd/cdd+ reference manual," *Institute for Operations Research, ETH-Zentrum*, pp. 91–111, 1997
MENTIONED ON PAGE 154

[247] K. Sagar, *Pygradientpolytope: A set of ros nodes for generating and visualizing cartesian velocity, force and desired polytopes*, https://gitlab.com/KeerthiSagarSN/rospygradientpolytope, 2023
MENTIONED ON PAGE 154

[248] P. Chiacchio, Y. Bouffard-Vercelli, and F. Pierrot, "Force polytope and force ellipsoid for redundant manipulators," *Journal of Robotic Systems*, vol. 14, no. 8, pp. 613–620, 1997. DOI: https://doi.org/10.1002/(SICI)1097-4563(199708)14:8<613::AID-ROB3>3.0.CO;2-P
MENTIONED ON PAGE 158

[249] C. Messeri, A. Bicchi, A. M. Zanchettin, and P. Rocco, "A dynamic task allocation strategy to mitigate the human physical fatigue in collaborative robotics," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2178–2185, 2022. DOI: 10.1109/LRA.2022.3143520
MENTIONED ON PAGE 169

[250] N. Rezzoug, A. Skuric, V. Padois, and D. Daney, *Simulation study of the upper-limb wrench feasible set with glenohumeral joint constraints*, 2023. [Online]. Available: https://arxiv.org/abs/2309.07487, *arXiv preprint*
MENTIONED ON PAGE 169