Whole-body teleoperation of the Talos humanoid robot: preliminary results

Eloïse Dalin, Ivan Bergonzani, Timothée Anne, Serena Ivaldi, Jean-Baptiste Mouret

Whole-body teleoperation of the Talos humanoid robot: preliminary results

Eloïse Dalin[1], Ivan Bergonzani[1], Timothée Anne[1], Serena Ivaldi[1], Jean-Baptiste Mouret[1]

***Abstract*— We propose a task-space, whole-body teleoperation framework for the Talos humanoid robot. Our main focus is to ensure the safety of the robot by preventing falls and collision: the whole-body controller tracks the hands and the head, while a repulsor-based task prevents self-collisions and a force-based stabilizer corrects the posture to avoid falls.**

I. INTRODUCTION

Our objective is to allow an operator to perform whole-body teleoperation with a motion capture suit and a virtual reality headset. We envision applications in which a humanoid robot is sent to hazardous places and the operator acts remotely in a fluid way, almost as they were "there".

We focus on retargeting the Cartesian position of the head and of the end-effectors (the hands), by contrast with previous work in which the motion of the hands was relative to the starting position and the trajectories were scaled to match the size of the robot [1]–[3]. At each time-step, a task-based whole-body controller [4], [5] computes the joint positions while taking into account the constraints (not falling, joint limits, etc.). In addition to track the end-effectors, our teleoperation controller ensures the stability and the safety of the robot:

- it leverages the force-torque sensors of the feet to reject external perturbations and stabilize the robot in case of highly dynamic motions;
- it prevents self-collisions.

In this short paper, we describe our whole-body teleoperation framework applied to Talos, a 100kg, 32-DOF and 1.75m humanoid robot from PAL Robotics.

II. WHOLE-BODY CONTROL FRAMEWORK : INRIA_WBC

We formulate the whole-body control as multi-objective optimisation problem: given several goals in a task space (usually Cartesian space), an optimiser finds the control inputs that both achieve the goal and respect the constraints. Those control inputs can be joint velocities, joint accelerations or joint torques.

Our whole-body control framework (inria_wbc[2]) is based on TSID[3] [6] to solve the whole-body control optimisation problem. The formulation is the following:

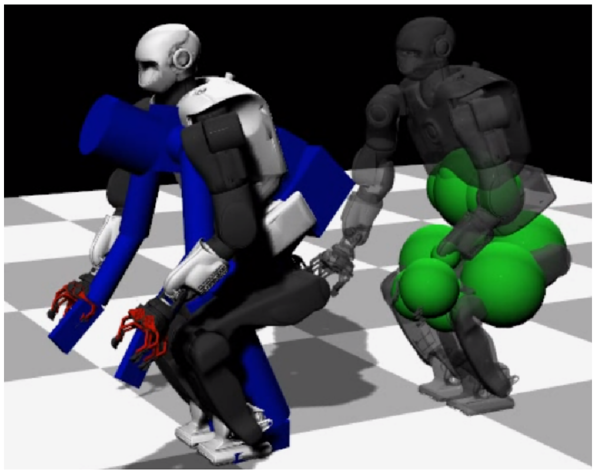

Fig. 1. **Talos teleoperation behavior simulated in robot_dart.** The 66 degrees of freedom human model is displayed as a graphical object in blue. It is overlaid to the simulated Talos. The grey Talos is also a pure graphical object (no forces, gravity, physics...) showing the direct whole-body control output solution. The green spheres are the display of the repulsors of the left gripper.

$$(\tau^*, \ddot{q}^*) = \underset{\tau, \ddot{q}}{\operatorname{argmin}} \sum_{k=0}^{n\_tasks} w_k \| A_k(q,\dot{q})\ddot{q} - b_k(q,\dot{q}) \|^2 \quad (1)$$

$$\text{s.t.} \begin{cases} A_{ineq}(q,\dot{q})\ddot{q} <= b_{ineq}(q,\dot{q}) \\ A_{eq}(q,\dot{q})\ddot{q} = b_{eq}(q,\dot{q}) \\ \tau = M(q)\ddot{q} + h(q,\dot{q}) - J(q)^{\intercal} f \end{cases}$$

where τ^* and $\ddot{q}^*$ are the optimal joint torques and accelerations. w_k is the weight of the task k described by $A_k(q,\dot{q})$ and $b_k(q,\dot{q})$. $M(q)$ is the joint-space mass matrix, $h(q,\dot{q})$ contains all the non-linear terms (Coriolis, centrifugal and gravity), J(q) is the contact jacobian and f the contact forces. Equality constraints are described by $A_{eq}(q,\dot{q})$ and $b_{eq}(q,\dot{q})$. Inequality constraints are described by $A_{ineq}(q,\dot{q})$ and $b_{ineq}(q,\dot{q})$. Tasks are weighted on the same level and their priority directly depends on w_k [5]. We get the optimal joint accelerations and integrate them twice to control the Talos in position, although we could use the same values and control the robot directly in torque.

TSID uses the Eiquadprog[4] QP solver and Pinocchio[5] [7], [8] to compute the dynamics and the rigid body quantities. Inria_wbc uses TSID and defines a set of functions and classes to design whole-body control experiments that

* This work was supported by the DGA (project "humanoide résilient"), the Inria ADT "wbTorque", the CPER projects "CyberEntreprises" and "SCIARAT", the European Union Horizon 2020 Research and Innovation Program under Grant Agreement No. 731540 (project AnDy), and a scholarship from ENS for Timothée Anne.

[1]Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France, jean-baptiste.mouret@inria.fr.

[2]https://github.com/resibots/inria_wbc

[3]https://github.com/stack-of-tasks/tsid

[4]https://github.com/stack-of-tasks/eiquadprog

[5]https://github.com/stack-of-tasks/pinocchio

combine controllers (a specific set of tasks), estimators (e.g., Center of pressure), and stabilization strategies.

Inria_wbc is optionally linked to robot_dart[6], a wrapper around DART[7] [9] for easy robot simulation. Robot_dart optionally uses Magnum[8] for the graphics. It makes it possible to test the behaviors and controllers in a simulated environment. DART is a fast synchronous physics simulator with no communication overhead (by contrast with Gazebo or ROS-based solutions).

III. Stabilization

We implemented several strategies from the literature [10], [11] to balance the robot using the force/torque sensors of the feet. These strategies are combined because they are often complementary [10].

Center of Mass (CoM) admittance control [10]: We modify the x and y com Cartesian position references x_{com} and its derivatives $\dot{x}_{com}$ and $\ddot{x}_{com}$ thanks to the difference between the measured center of pressure x_{mcop} and its value in the Pinocchio model x_{cop}. We compute the new references x^*_{com}, $\dot{x}^*_{com}$, $\ddot{x}^*_{com}$ to give to the whole-body controller with:

$$\begin{cases} x^*_{com} = x_{com} - k_d(x_{cop} - x_{mcop}) \\ \dot{x}^*_{com} = \dot{x}_{com} - k_v(x_{cop} - x_{mcop})/d_t \\ \ddot{x}^*_{com} = \ddot{x}_{com} - k_a(x_{cop} - x_{mcop})/d_t^2 \end{cases} \tag{2}$$

Where d_t is the controller time step and k_d, k_v, k_a are stabilization gains.

Ankle admittance control [10]: We follow the same principle as the CoM admittance control but we modify the roll and pitch reference of the ankle R_{ankle} according to the measured foot Center of Pressure (CoP) position x_{mfcop}. If x_{mfcop} is far from the middle of the foot position x_{foot}, then a roll and pitch angle modification will be used to keep the foot CoP in the middle of the foot:

$$\begin{cases} R^*_{ankle} = R_{ankle} - k_d(x_{foot} - x_{mfcop}) \\ \dot{R}^*_{ankle} = \dot{R}_{ankle} - k_v(x_{foot} - x_{mfcop})/d_t \\ \ddot{R}^*_{ankle} = \ddot{R}_{ankle} - k_a(x_{foot} - x_{mfcop})/d_t^2 \end{cases} \tag{3}$$

Please note that x_{foot} can be replaced by the desired foot CoP but that we have here considered the middle of the foot x_{foot} to be the most stable CoP position.

Foot force difference admittance control [11]: We follow the same principle as for the CoM and ankle admittance control. This time we modify the torso roll reference R_{torso} according to the force ratio difference from the sensors and from the model. As in [11] we take the following difference:

$$D = |f_{Lz} - f_{Rz}| - |f^m_{Lz} - f^m_{Rz}| \tag{4}$$

where f^m_{Lz} and f^m_{Rz} are the measured left and right foot z forces components. f_{Lz} and f_{Rz} are the contact model left and right foot z forces components. We then uses this difference here:

$$\begin{cases} R^*_{torso} = R_{torso} - k_d \cdot D \\ \dot{R}^*_{torso} = \dot{R}_{torso} - k_v \cdot D/d_t \\ \ddot{R}^*_{torso} = \ddot{R}_{torso} - k_a \cdot D/d_t^2 \end{cases} \tag{5}$$

ZMP distributor admittance control [11]: We take from [11] the following ZMP (Zero-Moment Point) distributor which converts the ZMP position p^d_{zmp} to target forces f^d_R, f^d_L and torques τ^d_i:

$$\begin{cases} f^d_R = -\alpha M g \\ f^d_L = -(1-\alpha) M g \\ \tau^d_i = -(p_i - p^d_{zmp}) \cdot f_i (i = R, L) \end{cases} \tag{6}$$

$\alpha \subset [0,1]$ indicates if p^d_{zmp} is closer to the right or left foot. We concatenate f^d_R, f^d_L and τ^d_i inside F^d and we use admittance control:

$$F^* = F - k_p \cdot (F - F^d) - k_d \cdot (F - F^d)/d_t \tag{7}$$

Where F is the initial forces and torques reference and F^* is the newly computed references to give to the contact force regularization task (Table. I)

In double a support scenario, the Talos is the most “stable” when the CoM is in its central xy position $(0,0)$. Our stabilizer helps to achieve this (Fig. 2). The video displays a case where the stabilizer prevents the robot from falling.

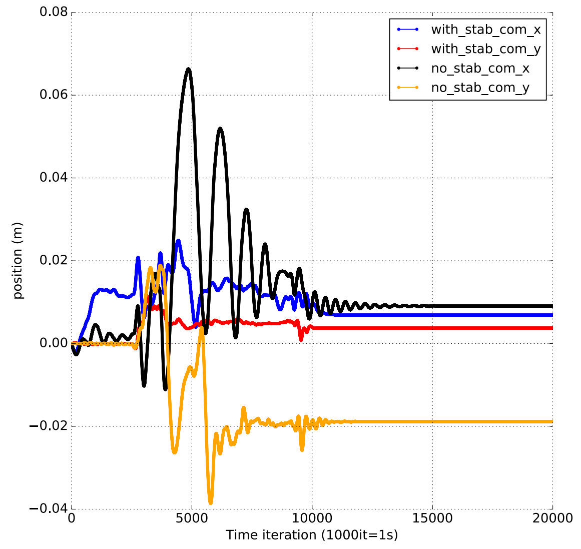

Fig. 2. **COM planar xy trajectory during a double support teleoperation test** Teleoperation references are given to a simulated Talos in robot_dart (Fig. 3). The references are given twice faster than in Fig. 4 and Fig. 5 which leads to COM instability. The blue and red plots are the output trajectory with our activated stabilizer. The black and orange plots are without stabilization.

IV. Self-collisions

It is critical in teleoperation to filter the commands that could damage the robot [1]. As an example, the Talos has a wide body and therefore the references for the hands are inside Talos' hips when the operator's arms are along their body. We prevent these self-collisions by adding repulsors

[6]https://github.com/resibots/robot_dart
[7]https://github.com/dartsim/dart
[8]https://github.com/mosra/magnum

(Fig. 1) between the grippers and the body. They are added as a task in the whole-body control problem (Table I).

To avoid those repulsors, we create an attractor function $C(q)$ where q is the joint position vector. $C(q)$ will attract a robot frame outside of the repulsion zones. Thus it should converge toward zero when we are far from repulsors:

$$\ddot{C} + K_d\dot{C} + K_pC = 0 \tag{8}$$

where K_p and K_d are convergence gains. Now we express $C(q)$ with respect to the Cartesian coordinates of the robot frame that needs to avoid the repulsion points:

$$\begin{aligned} &C : q \mapsto C(f(q)) = C(x) \\ &\text{with } \dot{x} = J(q)\cdot\dot{q} \end{aligned} \tag{9}$$

We develop the terms of the convergence equation (Eq. 8):

$$\begin{aligned} &\dot{C}(x) = \nabla C(x)^T \cdot J(q)\cdot\dot{q} \\ &\ddot{C}(x) = (\nabla^2 C(x)\cdot J(q)\cdot\dot{q})^T\cdot J(q)\cdot\dot{q} \\ &+\nabla C(x)^T\cdot(J(q)\cdot\ddot{q} + \dot{J}(q)\cdot\dot{q}) \end{aligned} \tag{10}$$

where $\nabla^2 C(x)$ is the Hessian and $\nabla C(x)$ is the gradient of $C(x)$. To have a whole-body control task formulation for the repulsors, we need to find $A_{rep}(q,\dot{q})$ and $b_{rep}(q,\dot{q})$ such that:

$$A_{rep}(q,\dot{q})\ddot{q} - b_{rep}(q,\dot{q}) = 0 \tag{11}$$

We can identify $A_{rep}(q,\dot{q})$ and $b_{rep}(q,\dot{q})$ by using the developed terms of Eq. 10 in Eq. 8:

$$\begin{aligned} &A(q,\dot{q}) = \nabla C(x)^T\cdot J(q) \\ &b(q,\dot{q}) = -((\nabla^2 C(x)\cdot J(q)\cdot\dot{q})^T\cdot J(q)\cdot\dot{q} \\ &+\nabla C(x)^T\cdot(\dot{J}(q)\cdot\dot{q} + K_dJ(q)\cdot\dot{q}) + K_pC(x)) \end{aligned} \tag{12}$$

We have chosen the following attractor function:

$$C(q) = \exp\left(-\frac{\|x-p_0\|^p}{a}\right) \tag{13}$$

Where $a = r_{rep} + r_{frame}$, with r_{rep} a radius around a repulsion point and r_{frame} a radius around the frame that needs to avoid the repulsors. p_0 is the repulsion point Cartesian coordinates. p is a user defined exponent. This function has been chosen because it is differentiable, smooth, and easy to tune. The exponential decay makes a fast convergence toward zero when we are far from p_0, which means that repulsors do not influence the behaviors when the two bodies are not close.

V. Teleoperation

We are using a Xsens motion tracking suit based on inertial sensors [12] to teleoperate the Talos robot. At each time step the Xsens software estimates the joint angle positions, the joint Cartesian positions, and the joint orientations (defined in a Xsens world frame) of the operator. Inria_wbc_teleop extends inria_wbc and creates a teleoperation behavior in which Cartesian references for the Talos' grippers and head are computed from Xsens data.

Here, we chose to use absolute Cartesian teleoperation. This means that we want Talos' grippers to follow the human hands as if the Talos and the human were overlaid.

TABLE I
Tasks, contacts and constraints used for teleoperation

tasks-contacts-constraints	mask (xyzrpy)	w_k	k_p
task: head	111000	100	8000
task: head_pitch	000010	100	30
task: head_yaw	000001	100	30
task: left_hand	111111	100	8000
task: right_hand	111111	100	8000
task: torso	000110	2000	30
task: left_foot	111111	1000	30
task: right_foot	111111	1000	30
task: com	110	2000000	3000
task: posture	111111	100.75	30
task: self_collision-left	111111	700	500
task: self_collision-right	111111	700	500
contact: left_foot	none	none	30.0
contact: right_foot	none	none	30.0
constraint: pos-vel-acc	none	none	none

Masks are given following the x, y, z, roll, pitch, yaw order. There is a 1 for a controlled dimension and 0 otherwise. The pos-vel-acc constraint is putting position, velocity and acceleration limits to each joint. The limits are by default taken in the urdf. k_p is the internal convergence gain of the task.

It is not possible to directly use the Cartesian positions and orientations given by the Xsens because Talos and Xsens data are not expressed in the same reference frame.

To align the frames, Inria_wbc_teleop is using a 66-DOFs URDF human model (Fig. 1) that corresponds to the Xsens joint angle data. A script adapts the human model size to each new user thanks to the input Xsens data. Then the module is initialized as follows:

- load the human model with Pinocchio;
- apply joint angles from the Xsens data to the model;
- compute the human model floating base position as if its feet frames were equal to Talos feet frame;
- use point cloud matching to compute the transform between Xsens and Pinocchio world frames; the points are the joint positions of the human model and of the Xsens data; Umeyama's method [13] is used as it is available in the Eigen library.
- apply this transform to Xsens joint Cartesian position data and compute the initial rotation between the human model joint frames and the Xsens Cartesian joint orientation.

The same transformation can be used at each time-step: it is computationally efficient and it makes it possible to take the desired human joint Cartesian positions and orientations as an input for the whole-body controller.

VI. Results

We tested the teleoperation in simulation with online and offline Xsens data. We disabled the stabilization so that the robot is in open-loop, which makes it easier to interpret the tracking errors. The position commands from the whole-body control are converted to torques thanks to Stable Proportional Derivative (SPD) control [15]. Those torques are then sent to the simulated Talos. Hence, the teleoperation tracking error might be affected by the gains of the SPD control.

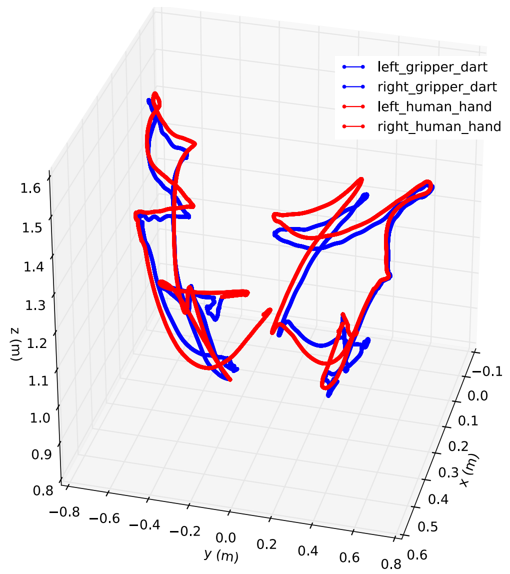

Fig. 3. **Grippers and human hands trajectories during teleoperation** The human hand trajectories are representated in red and are given as a reference to the whole-body control framework. The simulation is run thanks to robot_dart. Talos' grippers trajectory in the simulator are represented in blue. The tasks, contacts and constraints used are in Table I.

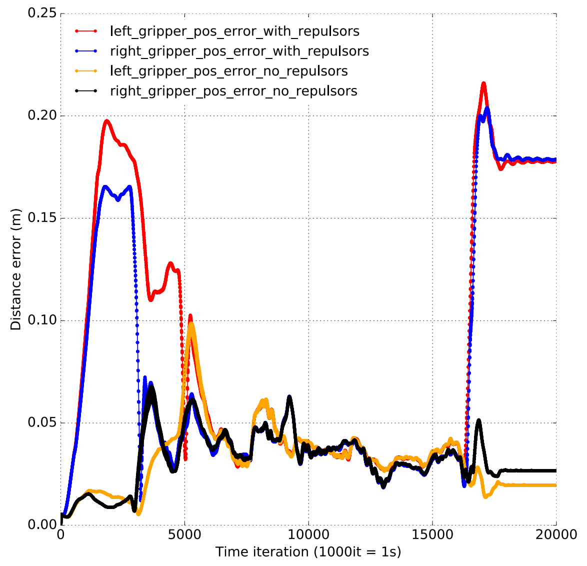

Fig. 4. **Grippers Position error during teleoperation** This corresponds to the Fig. 3 trajectory. At the beginning and end of the teleoperation experiment, the human hands are overlaid to Talos hips and the repulsors prevent any collision.

With our teleoperation framework, Talos' grippers are following the human wrists position and orientation (Fig. 3). The tasks, contacts and constraints are listed in Table I. At the initialisation, the operator's hands are inside the hips of the robot: the self collisions tasks prevents the collision and thus Talos' grippers do not follow the reference trajectory during the first time-steps (Fig. 4, Fig. 5)

To better evaluate the accuracy of the teleoperation, we disabled the repulsors and computed the tracking error for 8 test trajectories (Table II). On average, the position error is of

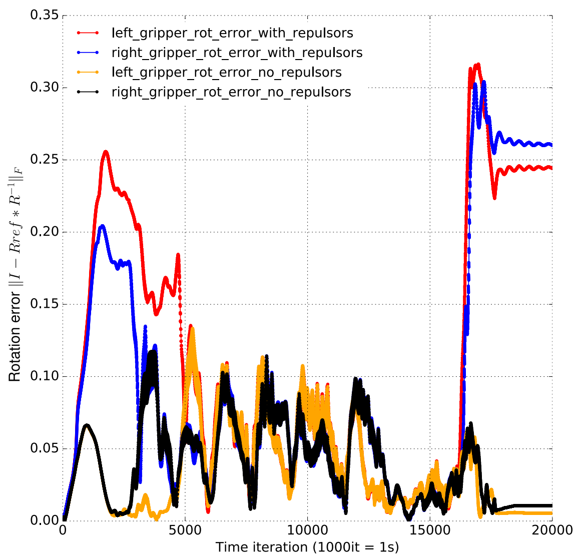

Fig. 5. **Grippers Rotation error during teleoperation** This corresponds to the Fig. 3 trajectory. At the beginning and end of the teleoperation experiment, the human hands are overlaid to Talos hips and the repulsors prevent any collision. Rotation error is computed by $\|I - Rref * R^{-1}\|_F$ [14] where $\|.\|_F$ is the Frobenius matrix norm, $Rref$ is the human hand reference orientation and R is the gripper current orientation. $\|I - Rref * R^{-1}\|_F$ gives an error in the range $[0, 2\sqrt{2}]$

TABLE II

LEFT AND RIGHT GRIPPER POSITION AND ROTATION ERRORS WITH DISABLED REPULSORS

	Mean	Standard deviation	Min	Max
Position Error	0.034 m	0.016 m	$1e^{-5}$ m	0.099 m
Rotation Error	0.025	0.021	$1.4e^{-6}$	0.134

Rotation error is computed by $\|I - Rref * R^{-1}\|_F$ [14] where $\|.\|_F$ is the Frobenius matrix norm, $Rref$ is the human hand reference orientation and R is the gripper current orientation. $\|I - Rref * R^{-1}\|_F$ gives an error in the range $[0, 2\sqrt{2}]$

about 3.5 cm. It is not yet fully satisfying for precise applications. Nevertheless precision might be improved thanks to sensor measurement feedback (as it is open loop control) and the Table I controller configuration might not be the optimal one. The following video shows these simulated experiments: `https://youtu.be/y5YIvLpxyr4` (the Trajectory0 of the video corresponds to Fig. 3, 4, and 5).

The stabilization has been disabled during those tests to simplify the interpretation of the tracking errors. Nevertheless, when some of the test trajectories are played faster, the simulated Talos falls. Our stabilizer avoid these falls (see the last part of the video). We suspect that on the real robot the stabilizer will have an even bigger impact because the Talos is usually more stable in our simulation.

For now, the robot only follows the hands and the head of the operator. In future work, we will extend this approach to track the feet, the contact points, and the center of mass. In addition, the current approach is generic enough to be adapted to other robots with minor changes.

References

[1] L. Penco, B. Clément, V. Modugno, E. M. Hoffman, G. Nava, D. Pucci, N. G. Tsagarakis, J.-B. Mouret, and S. Ivaldi, "Robust real-time whole-body motion retargeting from human to humanoid," in *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2018, pp. 425–432.

[2] L. Penco, N. Scianca, V. Modugno, L. Lanari, G. Oriolo, and S. Ivaldi, "A multimode teleoperation framework for humanoid loco-manipulation: An application for the icub robot," *IEEE Robotics & Automation Magazine*, vol. 26, no. 4, pp. 73–82, 2019.

[3] L. Penco, E. M. Hoffman, V. Modugno, W. Gomes, J.-B. Mouret, and S. Ivaldi, "Learning robust task priorities and gains for control of redundant robots," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2626–2633, 2020.

[4] V. Modugno, U. Chervet, G. Oriolo, and S. Ivaldi, "Learning soft task priorities for safe control of humanoid robots with constrained stochastic optimization," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2016, pp. 101–108.

[5] K. Bouyarmane and A. Kheddar, "On weight-prioritized multitask control of humanoid robots," *IEEE Transactions on Automatic Control*, vol. 63, no. 6, pp. 1632–1647, 2017.

[6] A. D. Prete, N. Mansard, O. E. Ramos, O. Stasse, and F. Nori, "Implementing torque control with high-ratio gear boxes and without joint-torque sensors," in *Int. Journal of Humanoid Robotics*, 2016, p. 1550044.

[7] J. Carpentier, G. Saurel, G. Buondonno, J. Mirabel, F. Lamiraux, O. Stasse, and N. Mansard, "The pinocchio c++ library – a fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives," in *IEEE International Symposium on System Integrations (SII)*, 2019.

[8] J. Carpentier and N. Mansard, "Analytical derivatives of rigid body dynamics algorithms," in *Robotics: Science and Systems*, 2018.

[9] J. Lee, M. X. Grey, S. Ha, T. Kunz, S. Jain, Y. Ye, S. S. Srinivasa, M. Stilman, and C. K. Liu, "Dart: Dynamic animation and robotics toolkit," *Journal of Open Source Software*, vol. 3, no. 22, p. 500, 2018.

[10] S. Caron, A. Kheddar, and O. Tempier, "Stair climbing stabilization of the hrp-4 humanoid robot using whole-body admittance control," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 277–283.

[11] S. Kajita, M. Morisawa, K. Miura, S. Nakaoka, K. Harada, K. Kaneko, F. Kanehiro, and K. Yokoi, "Biped walking stabilization based on linear inverted pendulum tracking," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 4489–4496.

[12] D. Roetenberg, H. Luinge, and P. Slycke, "Xsens mvn: Full 6dof human motion tracking using miniature inertial sensors," *Xsens Motion Technologies BV, Tech. Rep*, vol. 1, 2009.

[13] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 4, pp. 376–380, 1991.

[14] D. Huynh, "Metrics for 3d rotations: Comparison and analysis," *Journal of Mathematical Imaging and Vision*, vol. 35, pp. 155–164, 10 2009.

[15] J. Tan, K. Liu, and G. Turk, "Stable proportional-derivative controllers," *IEEE Comput. Graph. Appl.*, vol. 31, no. 4, pp. 34–44, 2011.