HAL
open science

Vision-based manipulation with the humanoid robot Romeo

Giovanni Claudio, Fabien Spindler, François Chaumette

To cite this version:

Giovanni Claudio, Fabien Spindler, François Chaumette. Vision-based manipulation with the humanoid robot Romeo. IEEE-RAS International Conference on Humanoid Robotics, Humanoids 2016, Nov 2016, Cancun, Mexico. hal-01385408

HAL Id: hal-01385408

https://inria.hal.science/hal-01385408v1

Submitted on 21 Oct 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Vision-based manipulation with the humanoid robot Romeo

Giovanni Claudio, Fabien Spindler and François Chaumette[1]

Abstract— **The aim of this paper is to show how visual servoing can help a humanoid robot to realize manipulation tasks with one and two hands, in conjunction with a gaze control. In addition, the use of vision in a closed-loop control scheme allows to accomplish these tasks with high repeatability without an accurate calibration of the robot kinematic model, the intrinsic and extrinsic parameters of the camera. The first application shown is grasping: one arm is controlled in order to reach the desired pose for a successful grasp along with a gaze control that keeps hand and object in the camera field of view. This approach is extended to the grasping of cylindrical objects, ignoring in the control the orientation along the revolution axis, that could cause an unnecessary motion of the arm. Moreover, we show how to control both arms with a master/slave approach for a two-handed manipulation without any force control. The robot is then able to solve a ball-in-maze game in augmented reality manipulating a tray with two hands, just using vision. A secondary task to avoid the joint limits using a large projection operator is added to improve the reliability of the system. Finally, the framework is implemented and evaluated on the humanoid robot Romeo using the ViSP library. The source code of the libraries, demos and examples are completely accessible on GitHub, allowing an easy adaptation of the visual servoing framework to other robots.**

I. INTRODUCTION

Humanoid robots are designed to emulate aspects of human form and behavior. Despite industrial manipulators and mobile robots, humanoid robots take advantage of human environments and equipments avoiding the need to alter surroundings and objects. These robots usually share similar kinematics to humans, as well as similar sensing.

About 550 million years ago, something triggered a great explosion of different life forms on earth. According to Andrew Parker's "Light-Switch Theory" [1], the reason for sudden diversification of animals was due to the development of the first form of the eye, which greatly increased the predator-prey importance of natural selection. Indeed, one of the most important sensory perceptual-cognitive systems of our brain is vision. In fact, according to some studies, 50% of our brain is involved in visual processing and 70% of all our sensory receptors are in our eyes [2]. With this in mind, it goes without saying why vision has become essential in robotics. Thanks to vision, the robots, before considered "blind" machines, acquire a new sense, improving their dynamic interaction with the environment and giving them the ability to perform daily human tasks.

Humanoid robots, despite industrial robots, usually have a low repeatability due to the discrepancy between model and real robot, because of inaccurate manufacturing processes and errors in the measurement of the joint positions. One way to solve this issue is to adopt the visual servoing approach, that allows to define a closed-loop scheme using information coming from a vision sensor.

An important task that humanoids should accomplish is grasping [3][4]. During the past decades, several works have addressed this challenge by proposing sensor-based control methods [5][6][7]. In [8] the robot HRP-2 grasps a simple ball while walking thanks to visual servoing. Using the robot ARMAR-III, a hybrid approach, which combines visual estimations with orientations computed through the kinematic model, is used to control the movement of a humanoid arm in [9]. The authors of [10] present a framework for visual servo control of the REEM robot upper body for a box grasping. In [11] a vision/force coupling approach is used to guide the robot hand towards the grasp position and perform a task manipulation taking into account external forces. Finally, in [12] a method for fast visual grasping of unknown objects with a multi-fingered robotic hand is described.

In the last decades, with the advent of humanoid robots and bi-manual industrial robots, the interest regarding dual arm manipulation has increased. The most common approach adopted is the hybrid force/position control and impedance control [13]. In [14] the target object and both hands are tracked alternately, and a combined open-closed-loop controller that uses a feedback from the force/torque sensors is used for positioning the hands with respect to the target. In absence of force sensors, other approaches have to be investigated. In this paper, we show a solution for dual arm manipulation based solely on vision.

Aside from grasping a cylindrical object, where we propose a clever set of visual features allowing an adequate behavior, this paper does not present any particular new methodological contributions. In fact, the aim of this work is to offer to the community a complete and fully explained application and integration of visual servoing approaches to control a humanoid robot, as well as the complete corresponding C++ code. We also show how visual servoing is robust to modeling and calibration errors by intentionally adding errors in the intrinsic and extrinsic camera parameters. Moreover, the framework is applicable to any humanoid robot to achieve tasks such as manipulation with one arm, dual arm coordination and gaze control. Here it is tested on the Romeo robot from SoftBank Robotics.

The paper is organized as follows: in Section II basic notions of visual servoing are recalled. In Section III the following visual servoing schemes are presented: an Image-Based Visual Servoing (IBVS) for gaze control, a Pose-Based

[1]G. Claudio, F. Spindler and F. Chaumette are with Inria at IRISA in Rennes, France. (giovanni.claudio@inria.fr, fabien.spindler@inria.fr, francois.chaumette@inria.fr)

This work was supported by Equipex Robotex and Oseo Romeo 2 project.

Visual Servoing (PBVS) for the grasping of a generic object, a special PBVS for the grasping of a cylindrical object and finally a master/slave approach for two-handed manipulation using only vision. Section IV shows the implementation and the results obtained on Romeo and Section V the software involved. Finally, Section VI features a summary and ideas for future works.

II. Visual servoing

Visual servo control refers to the use of information coming from vision sensors and extracted using computer vision algorithms to control the motion of a dynamic system, such as robotic arms, mobile robots or virtual cameras. The aim of a vision-based control scheme is to minimize an error $\|\mathbf{e}(t)\|$ built from the difference between an actual feature vector and the corresponding desired feature vector [15]:

$$\mathbf{e}\,(t) = \mathbf{s}(t) - \mathbf{s}^* \tag{1}$$

With a vision sensor providing 2D measurements, potential visual features are numerous. An IBVS uses directly 2D data information as features (for instance coordinates of image points, moments, lines and areas), while a PBVS uses 3D parameters extracted from the image measurements. In addition, it is possible to combine 2D and 3D visual features (Hybrid Visual Servoing) to take advantage of each approach while avoiding their respective drawbacks.

Considering a robot, a camera could be attached to its end-effector (eye-in-hand visual servoing) or it can be external observing the scene (eye-to-hand visual servoing). A visual servoing scheme can be defined to control in velocity the joints of a robot in order to reduce the error expressed in (1), computed using the information coming from the camera:

$$\dot{\mathbf{q}} = -\lambda \widehat{\mathbf{J}_\mathbf{e}^+}\mathbf{e} - \widehat{\mathbf{J}_\mathbf{e}^+}\widehat{\frac{\partial \mathbf{s}}{\partial t}} + \mathbf{P}_\lambda \mathbf{g} \tag{2}$$

where:

- $\widehat{\mathbf{J}_\mathbf{e}^+}$ is an estimation of the Moore-Penrose pseudo-inverse of the task Jacobian, a combination of the interaction matrix and the articular Jacobian of the robot. The form of $\mathbf{J}_\mathbf{e}$ depends on the type of visual servoing task adopted (see Section III).
- $\widehat{\frac{\partial \mathbf{s}}{\partial t}}$ is an estimation of the features velocity due to the generally unknown target motion.
- $\mathbf{P}_\lambda$ is the large projector operator proposed in [16] that allows to perform a secondary task even when the main task is full rank. It is defined as:

$$\mathbf{P}_\lambda = \bar{\lambda}\left(\|\mathbf{e}\|\right)\mathbf{P}_{\|\mathbf{e}\|} + \left(1 - \bar{\lambda}\left(\|\mathbf{e}\|\right)\right)\mathbf{P}_\mathbf{e} \tag{3}$$

with $\mathbf{P}_\mathbf{e} = (\mathbf{I}_n - \mathbf{J}_\mathbf{e}^+\mathbf{J}_\mathbf{e})$ the classical projection operator and $\mathbf{P}_{\|\mathbf{e}\|}$ the new projection operator that imposes the exponential decrease of the norm of the error, instead of each term of the error vector:

$$\mathbf{P}_{\|\mathbf{e}\|} = \mathbf{I}_n - \frac{1}{\mathbf{e}^\intercal\mathbf{J}_\mathbf{e}\mathbf{J}_\mathbf{e}^\intercal\mathbf{e}}\mathbf{J}_\mathbf{e}^\intercal\mathbf{e}\mathbf{e}^\intercal\mathbf{J}_\mathbf{e} \tag{4}$$

To ensure the convergence of the system, since that (4) is singular when $\mathbf{e} = 0$, a switching strategy is defined using a sigmoid function $\bar{\lambda}\left(\|\mathbf{e}\|\right)$, in order to switch from $\mathbf{P}_{\|\mathbf{e}\|}$ to $\mathbf{P}_\mathbf{e}$ when the norm of the error is reaching zero.

- $\mathbf{g}$ is a vector that defines the secondary task, it can be designed for instance to avoid joint limits, obstacles, occlusions or singularities.

In this paper, we designed the secondary task for avoiding the joint limits as described in [17] to make the velocity controller more reliable.

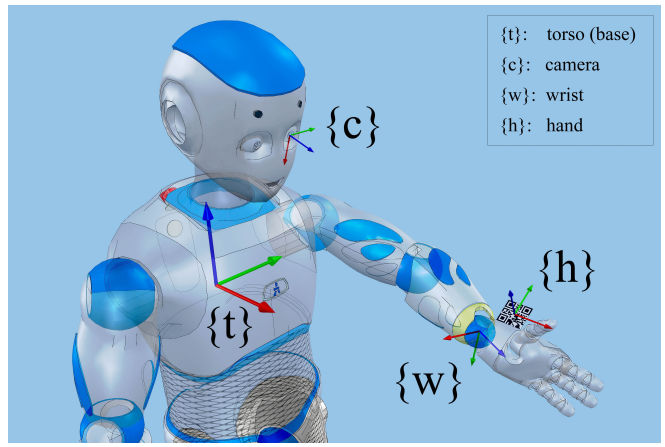

Fig. 1. Romeo frames definition: $\{t\}$ is the fixed base frame placed in the torso, $\{c\}$ is the camera frame on the eye of the robot, $\{w\}$ is the wrist frame in the last joint of the chain of the arm and $\{h\}$ is the frame placed on the hand target.

III. Visual servoing applied to a humanoid robot

Visual servoing is nowadays well established for the control of manipulator and mobile robots. An humanoid robot can be seen as a tree structure with a base link and several terminal links. Several branches can be defined starting from the base to each terminal link such as hands, feet, and eyes (see Figure 1). Each branch can be considered as a serial robot and therefore it is possible to control it with the same technique. The analysis increases in complexity with the addition of more branches, to perform different tasks or in coordination. In this section we present some visual servoing schemes for a humanoid robot, to accomplish tasks such as tracking a target with the gaze, grasp an object and use its hands together. The methodological approaches for grasping an object (Section III-A) and the gaze control (Section III-D) stem from the work presented in [10] on the robot REEM while the scheme for grasping a cylindrical object (Section III-B), for dual arm manipulation (Section III-C) and the applications (Section IV) are presented here for the first time.

A. Visual servoing for grasping

Humanoid robots need to perform everyday tasks similar to those performed by humans. One of the most important is the manipulation of objects. We will define here a control scheme that allows the robot to grasp a known object. We consider an eye-to-hand PBVS to control the arm joints using a camera in the robot's head. During the control, because of

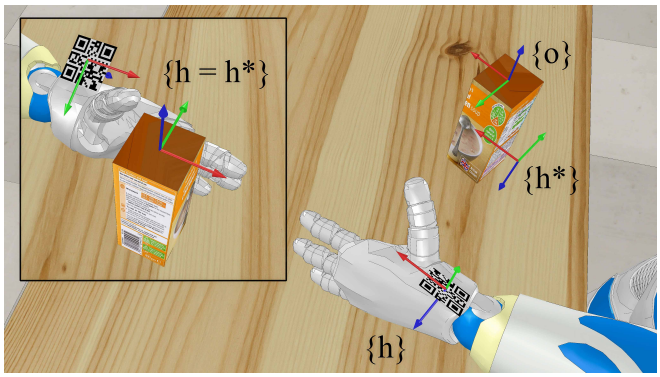

Fig. 2. Frames definition for grasping an object. The frame $\{o\}$ is attached to the object, $\{h\}$ to the marker on the hand and $\{h^*\}$ represents the desired pose of the hand. The transformation matrix ${}^{o}\mathbf{M}_{h^*}$ is constant. At the end of the PBVS the frames $\{h\}$ and $\{h^*\}$ will be coincident.

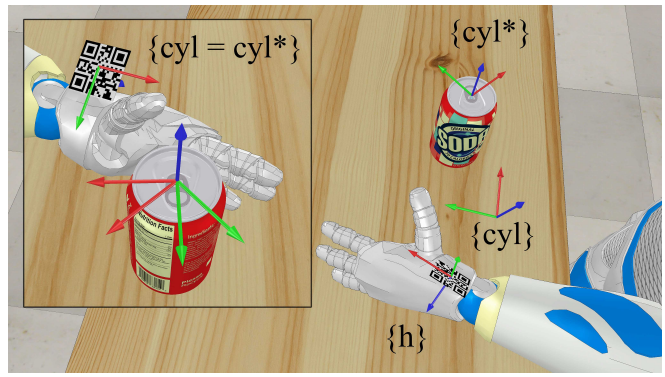

Fig. 3. Frames definition for grasping of a cylindrical object. $\{cyl*\}$ is the frame attached to the object, $\{h\}$ to the marker on the hand and $\{cyl\}$ represents the actual pose that we are controlling. The transformation matrix ${}^{h}\mathbf{M}_{cyl}$ is constant. At the end of the PBVS the frames $\{h\}$ and $\{h^*\}$ have to be coincident except for the rotation around the vertical axis, that is not relevant for the grasping, being the object symmetrical.

inaccuracy in the kinematic model of the robot, we need to compute the actual pose of the hand at each iteration, using vision. Moreover, the pose of the object to grasp (see Figure 2) is also required at each frame. The 6D pose of the hand ${}^{c}\mathbf{M}_{h}$ and of the object ${}^{c}\mathbf{M}_{o}$ can be estimated for instance using visual markers placed on them, or using a Model-Based Tracker (MBT) if the model is known. A goal frame ${}^{c}\mathbf{M}_{h^*}$ is also required to compute the servo. The pose ${}^{c}\mathbf{M}_{h^*}$ is equal to the pose of the object to grasp multiplied by a constant transformation ${}^{o}\mathbf{M}_{h^*}$, that can be easily learned by placing the hand at the desired position with respect to the object and using the following equation:

$$ {}^{o}\mathbf{M}_{h^*} = \left({}^{h^*}\mathbf{M}_{c}\,{}^{c}\mathbf{M}_{o}\right)^{-1} \tag{5} $$

During the visual servoing we want to minimize the error between the pose $\{h\}$ and $\{h^*\}$:

$$ {}^{h^*}\mathbf{M}_{h} = {}^{o}\mathbf{M}_{h^*}^{-1}\,{}^{c}\mathbf{M}_{o}^{-1}\,{}^{c}\mathbf{M}_{h} \tag{6} $$

From ${}^{h^*}\mathbf{M}_{h}$ we can extract the error vector composed by the error in translation and in orientation using the angle-axis representation:

$$ \mathbf{e}_{h} = \left({}^{h^*}\mathbf{t}_{h}, {}^{h^*}\theta\mathbf{u}_{h}\right) \tag{7} $$

and the corresponding interaction matrix as shown in [15]:

$$ \begin{gathered} \mathbf{L}_{\mathbf{e}} = \begin{bmatrix} {}^{h^*}\mathbf{R}_{h} & \mathbf{0}_{3} \\ \mathbf{0}_{3} & \mathbf{L}_{\theta\mathbf{u}} \end{bmatrix} \\ \mathbf{L}_{\theta\mathbf{u}} = \mathbf{I}_{3} - \frac{\theta}{2}[\mathbf{u}]_{\times} + \left(1 - \frac{\mathrm{sinc}\theta}{\mathrm{sinc}^{2}\frac{\theta}{2}}\right)[\mathbf{u}]_{\times}^{2} \end{gathered} \tag{8} $$

Finally, the joint velocity vector is computed from (2) setting $\widehat{\frac{\partial \mathbf{s}}{\partial t}}$ equal to zero:

$$ \begin{gathered} \dot{\mathbf{q}}_{h} = -\lambda\mathbf{J}_{h}^{+}\mathbf{e}_{h} + \mathbf{P}_{\lambda}\mathbf{g} \\ \text{with } \mathbf{J}_{h} = \mathbf{L}_{\mathbf{e}}\,{}^{h}\mathbf{V}_{w}\,{}^{w}\mathbf{J}_{w}\left(\mathbf{q}_{h}\right) \end{gathered} \tag{9} $$

where ${}^{h}\mathbf{V}_{w}$ is the constant twist transformation matrix to express the Jacobian ${}^{w}\mathbf{J}_{w}$ (which refers to the wrist joint) with respect to the hand frame $\{h\}$:

$$ {}^{h}\mathbf{V}_{w} = \begin{bmatrix} {}^{h}\mathbf{R}_{w} & \left[{}^{h}\mathbf{t}_{w}\right]_{\times}{}^{h}\mathbf{R}_{w} \\ \mathbf{0}_{3} & {}^{h}\mathbf{R}_{w} \end{bmatrix} \tag{10} $$

This scheme produces, under ideal condition, a pure straight line of the hand in the Cartesian space.

B. Visual servoing for cylindrical objects

In Section III-A we have seen how to deal with the grasping of a generic known object. Here we propose a new technique to improve the grasping for cylindrical objects based on the fact that the rotation around the revolution axis of the object is not relevant. In fact using a classic PBVS with all 6 Degrees Of Freedom (DOF) constrained, a small rotation of the object could cause a large motion of the arm, augmenting the risk of reaching joint limits. For this reason, we modify the PBVS scheme proposed in Section III-A.

Let ${}^{c}\mathbf{M}_{cyl^*}$ be the transformation between the actual frame of the cylindrical object with respect to the camera frame (as shown in Figure 3). To define the error vector and the interaction matrix, the transformation ${}^{cyl^*}\mathbf{M}_{cyl} = {}^{c}\mathbf{M}_{cyl^*}^{-1}\,{}^{c}\mathbf{M}_{cyl}$ is used, where ${}^{c}\mathbf{M}_{cyl} = {}^{c}\mathbf{M}_{h}\,{}^{h}\mathbf{M}_{cyl}$. The transformation matrix ${}^{c}\mathbf{M}_{h}$ refers to the hand target with respect to the camera and it is obtained by vision. The transformation ${}^{h}\mathbf{M}_{cyl}$ is a constant matrix from the hand target to the object after having placed the hand manually at the desired pose for a successful grasping (learned pose computed by vision). The features vector $\mathbf{s}$ has to be designed to ignore the rotation around the revolution axis of the object (illustrated as the blue axis $\mathbf{z}_{cyl^*}$ of the frame $\{cyl^*\}$ in Figure 3). The following feature vector is considered:

$$ \mathbf{s} = \begin{bmatrix} {}^{cyl^*}\mathbf{t}_{cyl} \\ {}^{cyl^*}\mathbf{z}_{cyl} \times \mathbf{z}_{d} \end{bmatrix} \tag{11} $$

where ${}^{cyl^*}\mathbf{z}_{cyl}$ is the third column vector of the rotational matrix ${}^{cyl^*}\mathbf{R}_{cyl}$. Since we want the revolution axis $\mathbf{z}_{cyl^*}$ of the cylinder and the axis related to the actual pose of the hand $\mathbf{z}_{cyl}$ to be coincident, we set $\mathbf{z}_{d} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^{\intercal}$.Therefore, due to the cross product involving $\mathbf{z}_{\mathbf{d}}$ in (11), the last element of the feature vector $\mathbf{s}$ is always equal to zero.

Finally, we can compute the error vector $\mathbf{e}_{c} = \mathbf{s}$ with $\mathbf{s}^{*} = 0$ and the interaction matrix $\mathbf{L}_{\mathbf{e}_{c}}$:

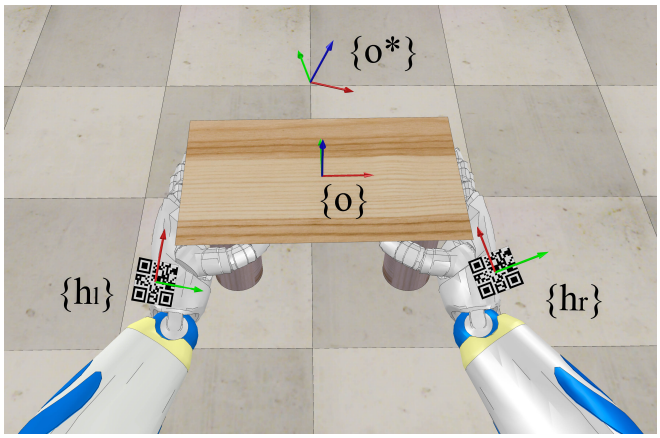

Fig. 4. Frames definition for a dual arm manipulation visual servoing scheme. $\{o\}$ is the frame attached to the object, $\{h_l\}$ and $\{h_r\}$ to the hand markers and $\{o^*\}$ represents the desired pose of the object.

$$\mathbf{L}_{\mathbf{e}_c} = \begin{bmatrix} {}^{cyl^*}\mathbf{R}_{cyl} & \mathbf{0}_3 \\ \mathbf{0}_3 & -\left[{}^{cyl^*}\mathbf{z}_{cyl}\right]_\times \left[\mathbf{z}_d\right]_\times \end{bmatrix} \tag{12}$$

We can notice that the rank of $\mathbf{L}_{\mathbf{e}_c}$ is equal to 5 since its last row is composed of six zero elements (due to the last component of $\mathbf{e_c}$, that is always equal to zero). This is coherent with the fact that, for this task, only 5DOF can be controlled, because of the cylinder symmetry.

For this case then, the task Jacobian expressed in (9) has the following form:

$$\mathbf{J}_h = \mathbf{L}_{\mathbf{e_c}} {}^{cyl}\mathbf{V}_w {}^{w}\mathbf{J}_w\left(\mathbf{q}_h\right)$$

Thanks to this scheme, when the visual servoing has converged, the axis $\mathbf{z}_{cyl}$ will be aligned to $\mathbf{z}_{cyl^*}$ so that the rotation error around $\mathbf{z}_{cyl^*}$ will be ignored as expected (see Figure 3).

C. Visual servoing for dual arm control

In this section, we consider the two-handed manipulation problem, where two arms create a closed kinematic chain holding a rigid body object with fixed grasp handles. For dual arm manipulation, common solutions are hybrid force/position control and impedance control [18][19], but if the arms are not equipped with any force sensors, as in our case, an alternative solution is needed. The aim here is to control both arms using only vision information, in order to apply any translation and rotation to the grabbed object, just knowing its pose with respect to the camera.

We consider the robot grabbing a tray as in Figure 4. The matrices ${}^c\mathbf{M}_{h_l}$ and ${}^c\mathbf{M}_{h_r}$ are the transformations from the camera to the left and right hand targets while ${}^c\mathbf{M}_o$ is the matrix expressing the pose of the object with respect to the camera.

1) Initialization: In this phase, the constant transformations between each hand target and the object are computed:

$$\begin{aligned} {}^{h_l}\mathbf{M}_o &= {}^c\mathbf{M}_{h_l}^{-1}{}^c\mathbf{M}_o \\ {}^{h_r}\mathbf{M}_o &= {}^c\mathbf{M}_{h_r}^{-1}{}^c\mathbf{M}_o \end{aligned} \tag{13}$$

2) Dual arm visual servoing: Even if each arm has 7 DOF, the mobility of the grasped object, when distant from joint limits and singularities, is equal to 6. A master/slave approach is used, where the left arm is the leader and the right arm is the follower. A PBVS is defined to control directly the pose of the object $\{o\}$, considered now as part of the chain of the robot, instead of the hand pose $\{h\}$ as in Section III-A. The joint velocities can be computed, setting $\widehat{\frac{\partial \mathbf{s}}{\partial t}}$ equal to zero, with the following control scheme:

$$\begin{aligned} \dot{\mathbf{q}}_{h_l} &= -\lambda \mathbf{J}_{o_l}^{+}\mathbf{e}_o + \mathbf{P}_\lambda \mathbf{g} \\ \text{with } \mathbf{J}_{o_l} &= \mathbf{L}_{\mathbf{e}} {}^{o}\mathbf{V}_{w_l} {}^{w_l}\mathbf{J}_{w_l}\left(\mathbf{q}_{h_l}\right) \end{aligned} \tag{14}$$

In this case ${}^o\mathbf{V}_{w_l}$ is the constant twist transformation matrix used to express the Jacobian of the left arm $\mathbf{J}_{w_l}$ in the object frame $\{o\}$, and is computed from the transformation matrix ${}^o\mathbf{M}_{w_l} = \left({}^{w_l}\mathbf{M}_{h_l}{}^{h_l}\mathbf{M}_o\right)^{-1}$. $\mathbf{L}_{\mathbf{e}}$ is the classical interaction matrix for PBVS (8) and $\mathbf{e}_o = \left({}^{o^*}\mathbf{t}_o, {}^{o^*}\theta\mathbf{u}_o\right)$ is the error between the desired and actual pose of the object.

Assuming that the two arms are not changing their relative pose with the object and themselves, we can impose the following kinematic constraint:

$$\mathbf{v}_o = \mathbf{J}_{o_r}\dot{\mathbf{q}}_{h_r} = \mathbf{J}_{o_l}\dot{\mathbf{q}}_{h_l} \tag{15}$$

Solving (15) for $\dot{\mathbf{q}}_{h_r}$, the joint velocities to apply to the follower arm are given by:

$$\dot{\mathbf{q}}_{h_r} = \mathbf{J}_{o_r}^{+}\mathbf{J}_{o_l}\dot{\mathbf{q}}_{h_l} \tag{16}$$

This scheme allows moving the object by coordinating both hands from the current position ${}^c\mathbf{M}_o$ to the desired one ${}^c\mathbf{M}_{o^*}$ following, in the ideal case, a 3D straight line trajectory.

D. Gaze control

Gaze control is a fundamental behavior for a humanoid robot because it allows keeping the interested target(s) in the field of view, overcoming the issue of a narrow field of view of the camera. For the gaze control several joints of the humanoid robot can be involved, such as the joints of trunk, neck, head and eyes.

An eye-in-hand IBVS is implemented, where the task consists of tracking an image point $\mathbf{x}$ in the camera frame. For this task we choose as feature the vector $\mathbf{s} = (x, y)$, containing the coordinates of the image point $\mathbf{x}$. The image point could be the center of an object to grasp, a hand, a face or even the midpoint of some of them. The error will be in this case $\mathbf{e_x} = (\mathbf{s} - \mathbf{s}^*)$, with $\mathbf{s}^* = (x^*, y^*)$, the vector containing the coordinates of the desired image point. The joints velocities $\dot{\mathbf{q}}_g$ to minimize the error can be computed using the following equation:

$$\dot{\mathbf{q}}_g = -\lambda \widehat{\mathbf{J}_g^{+}}\mathbf{e_x} - \widehat{\mathbf{J}_g^{+}}\widehat{\frac{\partial \mathbf{s}}{\partial t}} + \mathbf{P}_\lambda \mathbf{g} \tag{17}$$

The task Jacobian $\mathbf{J}_g$ is equal to:

$$\mathbf{J}_g = \mathbf{L_x} {}^c\mathbf{V}_e {}^e\mathbf{J}_e\left(\mathbf{q}_g\right) \tag{18}$$

with ${}^c\mathbf{V}_e$ the twist transformation matrix to express the Jacobian ${}^e\mathbf{J}_e$ (which refers to the last joint of the chain,

the eye) with respect to the camera frame $\{c\}$. $\mathbf{L_x}$ is the classical interaction matrix related to the coordinates of an image point $\mathbf{x} = (x, y)$:

$$\mathbf{L_x} = \begin{bmatrix} \frac{-1}{Z} & 0 & \frac{x}{Z} & xy & -(1+x^2) & y \\ 0 & -\frac{1}{Z} & \frac{x}{Z} & 1+y^2 & -xy & -x \end{bmatrix} \quad (19)$$

where Z is the depth of the observed point. If the tracked object is known, the depth value can be computed from its vision-based pose, otherwise a constant positive coarse value is sufficient for this purpose.

The term $\widehat{\frac{\partial \mathbf{s}}{\partial t}}$ is added to take into consideration perturbations to the system. If the motion of the target object is difficult to estimate, this term can be set equal to zero accepting that a tracking error will be observed. Otherwise, when the target on the hand is tracked, a feed-forward term can be added to remove the tracking error: we can set $\widehat{\frac{\partial \mathbf{s}}{\partial t}} = \mathbf{L_x}{}^c\mathbf{V}_w{}^w\mathbf{J}_w(\mathbf{q}_h)\,\dot{\mathbf{q}}_h$ as already done in [10].

IV. Implementation on Romeo

Romeo, a humanoid robot developed by SoftBank Robotics, is intended to be in the future a genuine personal assistant and companion. Romeo is 1,40 meters tall and it weights 40 kg. It is equipped with 37 motors, four cameras, four microphones, speakers, a tactile sensor on the head and IMUs. Two mobile cameras are placed on the eyes while other two are fixed in the forehead.

In order to overcome the large calibration inaccuracy of the kinematic model and the sensor errors (see Figure 5), the 6D pose of the hand is estimated using a known dimension target and ViSP pose estimation algorithm. For the experiments, two kinds of targets were used alternatively: a QR-Code and a target composed by four blobs (one red and the others black). Both targets are automatically detected and a coarse estimation of their fixed pose ${}^w\mathbf{M}_h$ with respect to the last joint of the arm is sufficient. Due to the low resolution of the images, the detection and tracking resulted more robust using the target with blobs.

During all the experiments we used the left eye camera. In fact, when dealing with objects of known dimensions, only one camera is needed, avoiding any calibration to know the relative pose between two cameras. The robot Romeo is connected via Ethernet cable to a remote computer, where all the libraries are installed and where the main algorithm is running. The images are acquired from the robot camera with QVGA (320x240) resolution at a frame-rate of approximately 15Hz. The general control scheme, which runs at the frequency of the acquisition rate of the camera, can be summarized as follow:

1) Get a new image from the robot.
2) Compute the actual visual features vector $\mathbf{s}^*$.
3) Get the actual state of the robot $\mathbf{q}$ (joint positions).
4) Get the robot Jacobians (from torso to arm and eye).
5) Compute $\mathbf{e}$, $\mathbf{L_e}$ and the task Jacobian.
6) Compute the control law to get the joint velocities $\dot{\mathbf{q}}$.
7) Add the secondary task velocities.
8) Send a new command to the robot (all the joint velocities are applied at the same time).

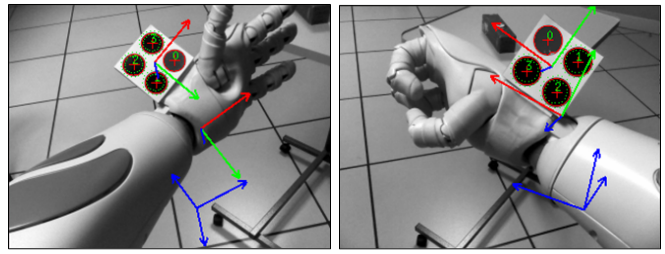

Fig. 5. Calibration errors in the kinematic model. The blue frame represents the "sensed" WristPitch frame computed using the robot model and the encoder sensor. The RGB frames represent the "sensed" target pose and the one computed by vision.

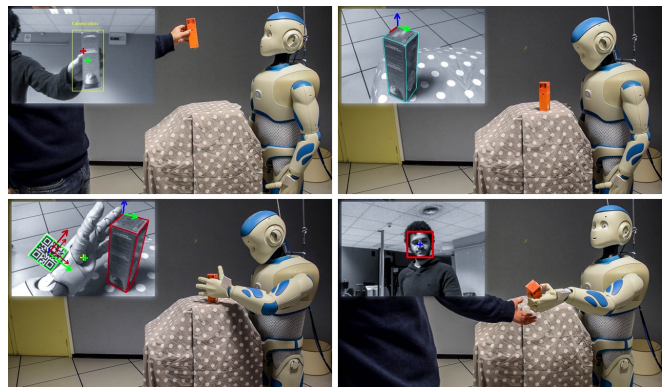

Fig. 6. Romeo detects a box and the table, grasp it and it delivers it to a human.

In this section we describe how the visual servoing schemes presented in Section III have been implemented and validated on Romeo (see accompanying video):

- Romeo grasps a box on a table and finally delivers it to a human [20].
- Romeo grasps a can and puts it back on a table [21].
- Romeo manipulates with its hands a tray in order to solve a ball-in-maze game in augmented reality [22].

A. Grasping an object using vision

1) Grasping a box: The implementation is mainly composed by two velocity controllers: an arm and a head-eye gaze controller. Each of Romeo's arm is composed by seven DOF: two in the shoulder, two in the elbow and three in the wrist. Since the hand is not equipped with any force sensor, only vision is used to determine when the hand is near enough to the object to perform a successful grasping. Since the box has known dimensions, the MBT available in ViSP is used to compute its pose. Here the steps of the demonstration are described:

a) Romeo detects the box using color information and tracks it with its gaze (Figure 6.A). For this task, six joints are controlled: two in the neck, two in the head and two in the eye. This is a typical eye-in-hand IBVS, by having a 2D image point as feature: the center of the box (see Section III-D). For the detection by color, OpenCV is used.
b) Once the box is on the table, Romeo estimates the box pose to initialize automatically the MBT (Figure 6.B).

Here the ViSP object localization algorithm uses key points to detect and estimate the object pose using its known model and texture.

c) Gaze control: An IBVS is implemented to see the box on the right part of the image. This is necessary to see both the hand and the box in the narrow field of view of the camera.
d) Open-loop motion of the arm to get close to the box using the robot odometry. This is not a repeatable motion and a closed-loop is necessary to grasp the box successfully.
e) The QR-Code on the hand is automatically detected and its pose is computed using the four image points of the corners and its dimension.
f) Grasping phase steps (Figure 6.C):
   - IBVS (see Section III-D) controlling the gaze to keep both the hand (QR-Code) and the box in the field of view of the eye's camera. The visual feature used is the image midpoint of the QR-Code and the box in the image.
   - PBVS (see Section III-A) to move the hand from the actual pose (extracted estimating the pose of the QR-Code) to the desired one (the grasping pose computed from the actual pose of the box). Note that the box can be placed at any reachable location and that the arm will adjust reactively its pose if the box is moved during the grasping process.
g) Once the desired pose is reached, Romeo closes its hand and grasps the box.
h) Romeo raises the head looking for a human. Once a face is detected an IBVS starts to track it.
i) Finally, Romeo delivers the box to the human moving the arm toward him (Figure 6.D).

2) Grasping a can: This case is similar to the previous task, except that the object to grasp has a cylindrical shape. For this reason the approach presented in Section III-B is used. Thanks to this visual servoing scheme, the hand moves, following a straight line in the Cartesian space, to the nearest convenient pose for a successful grasping. If a rotation around the vertical axis is applied to the can, the final desired pose of the hand will not change (as shown in the video [21] and in Figure 9.C). This allows to improve the robustness of the grasping. Also in this case, the ViSP MBT is used to track the can. In the demonstration, Romeo grasps the can, raises it and finally puts it back on the table. Figure 7 shows the task error and the joint velocities relative to the PBVS, which moves the arm from its actual position to the desired one. As expected, the component $\mathbf{r}_z$ is always zero. The experiment was repeated while adding intentionally an error in the intrinsic and extrinsic parameters in order to show that, thanks to visual servoing, the system is robust against this kind of perturbations (see Figure 8).

B. Dual arm manipulation - Ball-in-Maze game

To demonstrate that it is possible to perform a dual arm manipulation only with vision, without any force control, we developed an augmented reality demonstration using the control scheme explained in Section III-C.

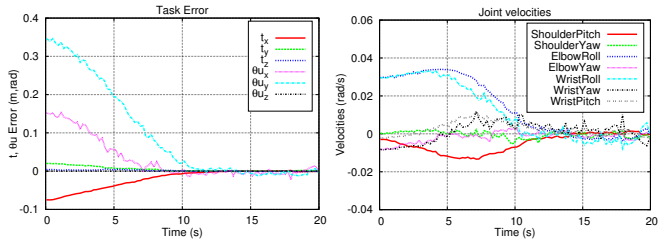

Fig. 7. Task error and joint velocities of Romeo's left arm while grasping a can using a PBVS (see Section IV-A.2).

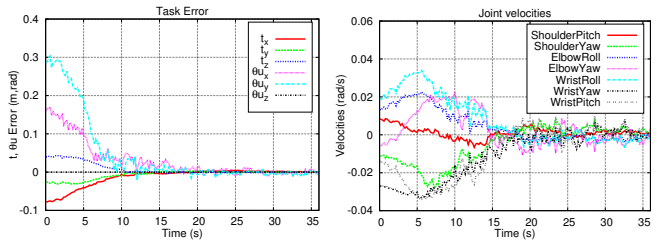

Fig. 8. Task error and joint velocities of Romeo's left arm while grasping a can using PBVS and adding some bias. The bias added to the estimated extrinsic parameters were: 3cm in each translation axis and 2 degrees in rotation. The bias added to the camera intrinsic parameters were: +40 pixels in the focal lengths and +10 pixels in the principal point.

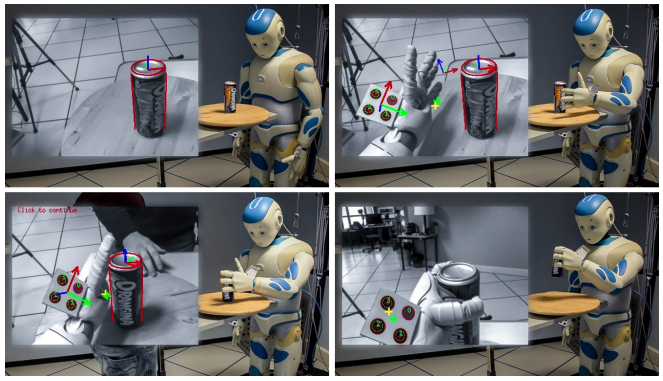

Fig. 9. Romeo grasps a can ignoring the orientation along the revolution axis in the control.

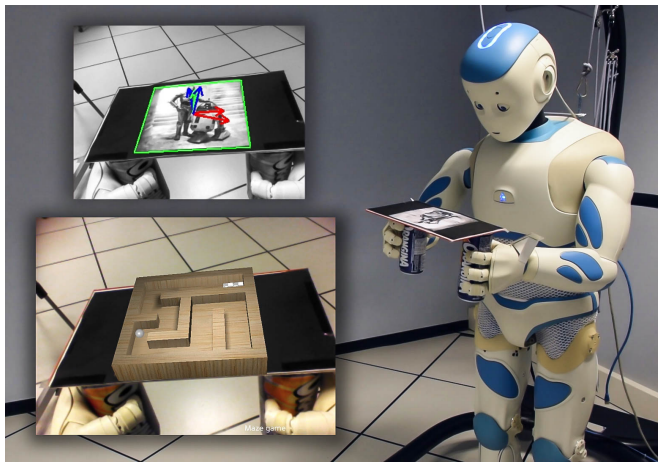

Fig. 10. Romeo solves a ball-in-maze game in augmented reality using two hands. Only vision is used to control both arms.

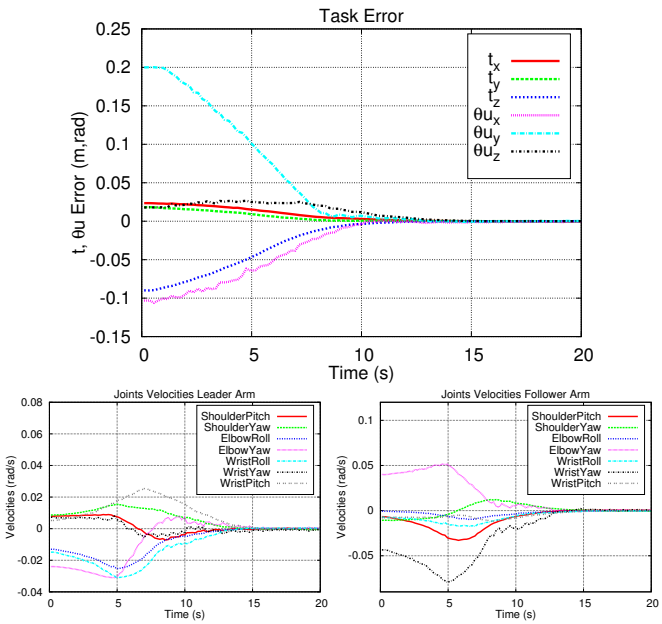

Fig. 11. Plots related to the initial Master/Slave PBVS experiment shown in Section IV-B, which moves the tray at a desired position before starting the ball-in-maze game.

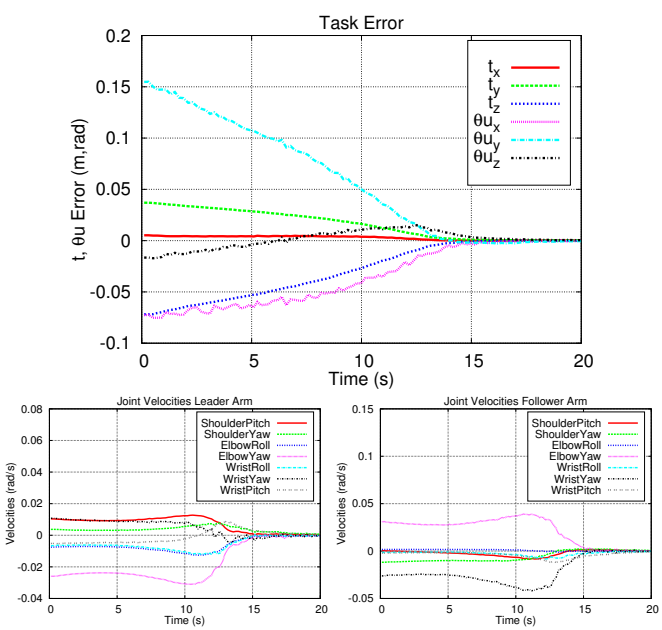

Fig. 12. Same Master/Slave PBVS experiment shown in Figure 11 while adding some bias in the intrinsic parameter: +30 pixels on the focal lengths and +15 pixel on the principal point. As we can see, the system is robust against this kind of perturbations

In this case we consider a coordinated manipulation where the two arms (14 joints in total) have to hold and manipulate the same object. The motion of the arms has to be well synchronized otherwise the object could break, fall or fail to reach the desired pose. The two arms are holding a tray from two handles as in Figure 10. A known picture is placed on the tray, which is automatically detected and then tracked using the template tracker in ViSP, that computes its 6D pose with respect to the camera. First of all, the transformation matrices from the hand targets to the object have to be computed using (13). Since the field of view of the camera is not large enough to see in the same images the two hand targets and the object, it is necessary to compute before the transformation of the right hand, then move the head, and finally compute the transformation for the left hand. While the head is moving, the picture is continuously tracked, even if only a portion of it is visible in the image (see video [22]).

Once the calibration phase is completed, we need to track only the picture. The tray is moved to a convenient position (parallel to the ground) in order to start the game (see Figure 11 and 12). Then a virtual maze is added in augmented reality on the top of the tray and its pose is directly linked with the pose of the picture. The aim of the game is to roll the virtual ball from its actual position to the end of the maze. First, Romeo checks where the ball is in the maze using the simulation information and then, it rotates the tray in order to roll the ball to the next corridor until the end of the maze, by taking advantage of its walls and controlling both arms simultaneously.

V. Software

A tutorial and the source code to reproduce the grasping demos (box and can) can be found in GitHub[1], as well as for the dual arm manipulation demo[2].

The main software used to develop this framework are the SoftBank Robotics SDK C++, ViSP [23], ViSPNaoqi, Panda3D [24], Metapod [25] and OpenCV [26].

ViSP (standing for Visual Servoing Platform) is a modular cross platform library that allows prototyping and developing applications using visual tracking and visual servoing technics to control the robots. It provides a set of visual features that can be tracked using real-time image processing and computer vision algorithms.

ViSPNaoqi is a bridge between ViSP and the SoftBank Robotics SDK C++, the library that contains all the tools to manage the robot. ViSPNaoqi is used mainly to grab images from Romeo, estimate the intrinsic and extrinsic camera parameters, control the robot in velocity and get the actual articular Jacobians. The library is available online[3] and it can also be used with other robots compatible with NAOqi (like Nao and Pepper).

Panda3D is a game engine, a framework for 3D rendering and game development for Python and C++ programs. It is

[1] https://github.com/lagadic/romeo_tk/wiki/Romeo-Grasping-Demo
[2] https://github.com/lagadic/ar_maze_romeo
[3] https://github.com/lagadic/visp_naoqi

used for the augmented reality demonstration, to simulate the ball rolling on the maze, and for handling the collisions with the floor and the wall. Two threads are created for the ball-in-maze demonstration: the first one is related to the image processing and control of Romeo, while the second one takes care of the ball-in-maze simulation in augmented reality.

In order to compute the kinematic model of the robot, the software Metapod is used. Metapod (for META-Programming Optimized Dynamics library) provides robot dynamics algorithms making use of R. Featherstone's Spatial Algebra [27] to describe forces, motions and inertias. To create the model, the URDF description of the robot is necessary.

OpenCV (Open Source Computer Vision) is a library aimed to develop real-time computer vision algorithm. Here, it is used for the face recognition, color detection, color segmentation, and indirectly by ViSP.

VI. CONCLUSIONS

In this paper, several visual servoing schemes have been presented: an IBVS for gaze control, a PBVS for grasping of a generic object, a special PBVS for grasping of a cylindrical object, and finally a master/slave approach for two-handed manipulation using only vision. These schemes lead to the implementation of real applications, achieved on the humanoid robot Romeo: box grasping, can grasping, and a two-handed manipulation for holding a tray, in order to solve a ball-in-maze game in augmented reality. The implementation on Romeo demonstrates that these tasks can be executed with high reliability and robustness despite an imprecise kinematic model, backslash of the joints, small resolution and low frame rate of the images coming from Romeo's camera. In addition, to improve the robustness, if one of the tracked targets is lost, Romeo re-detects it automatically and initializes the tracker with the new estimated pose. Moreover, the source code is entirely available to the community in order to simplify the implementation of our work on other humanoid robots.

This work gives room for numerous extensions and improvements. The next step is to adopt a MBT to track and estimate the 6D pose of the hand, allowing to get rid of the additional visual targets. The algorithms (or part of them) could be run directly on Romeo's CPUs in order to overcome the bottleneck of communication between a remote computer and the robot. Moreover, this work could be joined with a whole-body control framework, in order to take into account other important tasks such as the overall stability of the robot. When the robot will have a stable walk, the aim is to control its gait and the direction of its head using visual information. When walking, the images acquired by the cameras will probably be very unstable because of the steps of the robot. Tracking algorithms have to be redefined or modified to take into account the large displacements of objects in the images even if the gaze control should diminish this effect.

REFERENCES

[1] J. Lupovitch and G. A. Williams, "In the blink of an eye: How vision sparked the big bang of evolution," *Archives of Ophthalmology*, vol. 124(1):142, 2006.

[2] E. N. Marieb and K. Hoehn, *Human anatomy & physiology*. Pearson Education, 2007.

[3] K. B. Shimoga, "Robot grasp synthesis algorithms: A survey," *The Int. Journal of Robotics Research*, vol. 15(3):230-266, 1996.

[4] A. Bicchi and V. Kumar, "Robotic grasping and contact: a review," in *IEEE ICRA*, vol. 1(1):348-353, 2000.

[5] R. Horaud, F. Dornaika, and B. Espiau, "Visually guided object grasping," *IEEE ICRA*, vol. 14(4):525-532, 1998.

[6] D. Kragić, A. T. Miller, and P. K. Allen, "Real-time tracking meets online grasp planning," in *IEEE ICRA*, vol. 3(1):2460-2465, 2001.

[7] H. Fujimoto, L.-C. Zhu, and K. Abdel-Malek, "Image-based visual servoing for grasping unknown objects," in *IECON/IEEE Industrial Electronics Society*, vol. 2(1):876-881, 2000.

[8] N. Mansard, O. Stasse, F. Chaumette, and K. Yokoi, "Visually-guided grasping while walking on a humanoid robot," in *IEEE ICRA*, 2007, pp. 3041–3047.

[9] N. Vahrenkamp, S. Wieland, P. Azad, D. Gonzalez, T. Asfour, and R. Dillmann, "Visual servoing for humanoid grasping and manipulation tasks," in *8th IEEE-RAS Int. Conference Humanoid Robots*, 2008, pp. 406–412.

[10] D. Agravante, J. Pages, and F. Chaumette, "Visual servoing for the REEM humanoid robot's upper body," in *IEEE ICRA*, 2013, pp. 5253–5258.

[11] M. Prats, P. Martinet, A. P. Del Pobil, and S. Lee, "Vision force control in task-oriented grasping and manipulation," in *IEEE/RSJ IROS*, 2007, pp. 1320–1325.

[12] V. Lippiello, F. Ruggiero, B. Siciliano, and L. Villani, "Visual grasp planning for unknown objects using a multifingered robotic hand," *IEEE/ASME Transactions on Mechatronics*, vol. 18(3):1050-1059, 2013.

[13] C. Smith, Y. Karayiannidis, L. Nalpantidis, X. Gratal, P. Qi, D. V. Dimarogonas, and D. Kragic, "Dual arm manipulation a survey," *Robotics and Autonomous systems*, vol. 60(10):1340–1353, 2012.

[14] N. Vahrenkamp, C. Böge, K. Welke, T. Asfour, J. Walter, and R. Dillmann, "Visual servoing for dual arm motions on a humanoid robot," in *on 9th IEEE/RAS Int. Conference Humanoid Robots*, 2009, pp. 208–214.

[15] F. Chaumette and S. Hutchinson, *Visual servoing and visual tracking.* In Handbook of Robotics, B. Siciliano, O. Khatib (eds.), Chap. 24, pp. 563-583, Springer, 2008.

[16] M. Marey and F. Chaumette, "A new large projection operator for the redundancy framework," in *IEEE ICRA*, 2010, pp. 3727–3732.

[17] ——, "New strategies for avoiding robot joint limits: Application to visual servoing using a large projection operator," in *IEEE/RSJ IROS*, 2010, pp. 6222–6227.

[18] F. Caccavale, P. Chiacchio, A. Marino, and L. Villani, "Six-DOF impedance control of dual-arm cooperative manipulators," *IEEE/ASME Trans. on Mechatronics*, vol. 13(5):576-586, 2008.

[19] R. Bonitz and T. Hsia, "Internal force-based impedance control for cooperating manipulators," *IEEE ICRA*, vol. 12(1):78-89, 1996.

[20] Video, "Romeo humanoid robot grasping demonstration," https://youtu.be/TgpLfgFSca8.

[21] ——, "Romeo grasping a can by visual servoing," https://youtu.be/6yB5pQm4s_c.

[22] ——, "Two-handed manipuation: Romeo solving a ball-in-maze game." https://youtu.be/-wIzJ2Ckifg.

[23] E. Marchand, F. Spindler, and F. Chaumette, "ViSP for visual servoing: a generic software platform with a wide class of robot control skills," *Robotics & Automation Magazine, IEEE*, vol. 12(4):40-52, 2005.

[24] M. Goslin and M. R. Mine, "The Panda3D graphics engine," *Computer*, vol. 37(10):112-114, 2004.

[25] M. Naveau, J. Carpentier, S. Barthelemy, O. Stasse, and P. Soueres, "METAPOD - Template META-programming applied to dynamics: CoP-CoM trajectories filtering," in *14th IEEE-RAS Int. Conference on Humanoid Robots (Humanoids)*, 2014, pp. 401–406.

[26] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly Media, Inc., 2008.

[27] R. Featherstone, *Rigid body dynamics algorithms*. Springer, 2014.