Visual servoing in an optimization framework for the whole-body control of humanoid robots

Don Joven Agravante, Giovanni Claudio, Fabien Spindler, François Chaumette

Visual servoing in an optimization framework for the whole-body control of humanoid robots

Don Joven Agravante[1], Giovanni Claudio[1], Fabien Spindler[1], and François Chaumette[1]

***Abstract*—In this paper, we show that visual servoing can be formulated as an acceleration-resolved, quadratic optimization problem. This allows us to handle visual constraints, such as field of view and occlusion avoidance, as inequalities. Furthermore, it allows us to easily integrate visual servoing tasks into existing whole-body control frameworks for humanoid robots, which simplifies prioritization and requires only a posture task as a regularization term. Finally, we show this method working on simulations with HRP-4 and real tests on Romeo.**

***Index Terms*—Visual servoing, optimization and optimal control, humanoid robots.**

I. Introduction and background

VISUAL servoing is a method of control that directly incorporates visual information into the control loop [1]. It has been shown that it is effective in various robotics applications. In particular, it is often used for gaze control and grasping with humanoids [2]–[4]. On the other hand, optimization is a process that seeks to obtain the minimum (or maximum) of a particular objective under some constraints [5]. Optimization algorithms have proven to be a good framework for control, especially on complex systems such as humanoid robots. This is evidenced by various groups, working on humanoids, adapting similar optimization-based approaches [6]–[12]. The purpose of this paper is to show how visual servoing can be effectively used within these optimization frameworks. In doing so, we can gain the advantages of the optimization approaches (e.g., handling multiple tasks and constraints), yet retain the main research results of the visual servoing community. Specifically, we want to keep using the various visual features and the corresponding Jacobians that are well suited to be used in control, such as those collected and implemented in [13].

In the literature related to visual servoing, the problem of enforcing constraints has been one of the key issues because of the need to keep features in the field of view and avoid occlusions. In the state of the art, these are usually handled by firstly planning a path/trajectory in image-space and then tracking the result with visual servoing [14]–[17]. One way to do the planning phase is with optimization [15]. So by using optimization for visual servoing itself, we can proceed without the planning phase but still handle constraints. This was one of the main ideas in [18] which is probably the closest related work. In [18], visual servoing with constraints is formulated as a Model Predictive Control (MPC) problem that is solved in an optimization framework. This was validated in a simulation of a free-flying camera and four image points for features. Compared to [18], we firstly pose a novel second-order model. This is important to be consistent with dynamics. Secondly, we do not use predictive control and the cost function is formed differently such that we can use quadratic optimization, avoiding the complexities of using non-convex optimization. MPC implies an additional computational cost since the state and control vectors are multiplied by N previewed steps. We choose to avoid this for now. Meanwhile, the cost function formulation allows us a wider choice of convergence properties as motivated in Sec. IIA. Thirdly, we use the model to form inequalities for both the field of view constraint and occlusion avoidance. Finally, we show that our formulation integrates well to existing task objectives and constraints used to control a humanoid platform. There are also other works similar to [18]. Notably, [19] had similarly framed the visual servoing problem in an MPC and optimization context. This was integrated inside of an existing MPC for walking pattern generation (WPG) of a humanoid. Being an MPC formulation of visual servoing, our novelty claims relative to it are the same except for application to a humanoid platform. In regards to this, we create the visual servoing objectives and constraints directly in the whole-body controller while [19] does it in the WPG. Historically, the WPG is solved ahead of time and separately so that it can produce reference trajectories for the whole-body controller to track [20]. Recently, [21] has shown that both problems can be solved together. Leveraging this result, we argue that formulating visual servoing as *just another* task/constraint for the whole-body controller is better. Not only does it ensure consistency with actuation constraints (e.g., joint limits), it also simplifies the overall framework for task prioritization, i.e., both *walking* and *visual servoing* are just another set of tasks/constraints in the same optimization problem.

The rest of the paper is structured as follows. Sec. II defines the base formulation. We then show how inequalities can naturally represent the field of view and occlusion constraints in Sec. III. Simulation results are presented in Sec. IV and Sec. V shows some tests on a real humanoid robot. Finally, Sec. VI concludes and outlines some possible future works.

Manuscript received: September, 2, 2016; Revised November, 22, 2016; Accepted December, 15, 2016.

This paper was recommended for publication by Editor Dongheui Lee upon evaluation of the Associate Editor and Reviewers' comments. This work was supported in part by the BPI Romeo 2 and the H2020 Comanoid projects (www.comanoid.eu)

[1]All authors are with Inria Rennes - Bretagne Atlantique, Lagadic group, Campus de Beaulieu, 35042 Rennes, France. firstnames.lastname@inria.fr

Digital Object Identifier (DOI): see top of this page.

II. Base Formulation

Classically, visual servoing techniques use a first-order motion model of the visual features and the Moore-Penrose pseudoinverse to solve the system of equations, formally:

$$\dot{\mathbf{e}} = \mathbf{L_e}\mathbf{v}, \tag{1}$$

$$\mathbf{v} = -\lambda \widehat{\mathbf{L}_\mathbf{e}^+}\mathbf{e}, \tag{2}$$

where $\mathbf{e}$ is an error vector of the chosen visual features, $\mathbf{v}$ the velocity of the camera, $\mathbf{L_e}$ is the visual feature's Jacobian (or interaction matrix) with $\widehat{\mathbf{L}_\mathbf{e}^+}$ the pseudoinverse of its estimate (denoted by the hat), and finally λ is a gain to be tuned. Notice the two parts: the model (1) and the control law (2). A primary idea of this paper is that re-using (1) is beneficial while (2) can be done more generally by optimization. Using the pseudoinverse for velocity-resolved control is not unique to visual servoing. In fact, the exact same method can be found in general robotics literature as instantaneous inverse kinematics. A useful parallel can be drawn between the pseudoinverse and optimization, as explained in [22], where an equivalent optimization problem can be created. For example, (2) is the same as:

$$\begin{aligned} \mathbf{v} = \underset{\mathbf{v}}{\text{argmin}} \quad & \|\mathbf{v}\|^2 \\ \text{subject to} \quad & \widehat{\mathbf{L}_\mathbf{e}}\mathbf{v} = -\lambda\mathbf{e}. \end{aligned} \tag{3}$$

In fact, several parallels can be drawn between *classical* robot control approaches and optimization, as presented in [23]. Contrary to [23], which focused on a novel method to solve the optimization problem, this paper concentrates on the problem construction while leveraging optimization *solvers* such as those explained in the reference text [5].

To start detailing the base formulation, let us first generalize $\mathbf{e}$ to be any task vector in the operational space, then we define the function, $f_\mathbf{e}$, that maps the joint positions $\mathbf{q}$ to this space:

$$\mathbf{e} = f_\mathbf{e}(\mathbf{q}). \tag{4}$$

A classical example is using Cartesian space to define $\mathbf{e}$ and the function $f_\mathbf{e}$ is known from forward kinematics. Assuming $f_\mathbf{e}$ is twice differentiable, we can define:

$$\dot{\mathbf{e}} = \mathbf{J_e}\dot{\mathbf{q}}, \tag{5}$$

$$\ddot{\mathbf{e}} = \mathbf{J_e}\ddot{\mathbf{q}} + \dot{\mathbf{J}}_\mathbf{e}\dot{\mathbf{q}}, \tag{6}$$

where $\mathbf{J_e}$ is known as the task Jacobian. Note how (5) has a similar form to (1). Furthermore, (6) has a well known form and can be used for effective Cartesian space control in an optimization framework [8], [24]. Returning first to (5), it can be related to (1) by expanding into:

$$\dot{\mathbf{e}} = \mathbf{L_e}\mathbf{J_p}\dot{\mathbf{q}}, \tag{7}$$

by setting $\mathbf{J_e} = \mathbf{L_e}\mathbf{J_p}$. Here, $\mathbf{J_p}$ is the Jacobian of a corresponding robot frame $\mathbf{p}$. For example, if $\mathbf{p}$ is the camera frame, then (7) is equal to (1). Continuing, (6) then becomes:

$$\ddot{\mathbf{e}} = \mathbf{L_e}\mathbf{J_p}\ddot{\mathbf{q}} + \mathbf{L_e}\dot{\mathbf{J}}_\mathbf{p}\dot{\mathbf{q}} + \dot{\mathbf{L}}_\mathbf{e}\mathbf{J_p}\dot{\mathbf{q}}. \tag{8}$$

We now have (7) and (8), in the same form as (5) and (6). With these, we can formulate an optimization problem consistent with the framework of [8].

A. Quadratic programming objective

Recall that a general optimization problem can be written as finding $\mathbf{x}$ such that:

$$\begin{aligned} \underset{\mathbf{x}}{\text{argmin}} \quad & f_o(\mathbf{x}) \\ \text{subject to} \quad & f_c(\mathbf{x}) \leq \mathbf{0}, \end{aligned} \tag{9}$$

where $f_o(\mathbf{x})$ is the objective function and the inequality $f_c(\mathbf{x}) \leq \mathbf{0}$ is the constraint which is infinitely more important than the objective. A constrained quadratic programming (QP) problem can be formulated when:

$$f_o(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top\mathbf{Q}\mathbf{x} + \mathbf{c}^\top\mathbf{x}, \tag{10}$$

$$f_c(\mathbf{x}) = \mathbf{A}\mathbf{x} - \mathbf{b}. \tag{11}$$

For example, (3) is a QP by defining $\mathbf{x} = \mathbf{v}$. The objective function can then be formed like (10) where $\mathbf{Q}$ is an identity matrix and $\mathbf{c}$ is a vector of zeros. For the constraint, the general form of (9) is an inequality. But we can form equality constraints, such as that in (3), with inequalities by creating artificial upper and lower limits which are equal. Doing this for (3), we can recover the form of (11) where $\mathbf{A} = [\widehat{\mathbf{L}_\mathbf{e}}^\top \quad -\widehat{\mathbf{L}_\mathbf{e}}^\top]^\top$ and $\mathbf{b} = [-\lambda\mathbf{e}^\top \quad \lambda\mathbf{e}^\top]^\top$. However, most QP solver interfaces can explicitly handle equalities, so this is not always needed. For example, active set strategies [5] can benefit from knowing there is one equality that is always active rather than two inequalities to regularly check. Once we have a well-formed QP, it can be solved reliably [5]. To simplify the explanations that follow, we define the argument of the optimization, $\mathbf{x} = \ddot{\mathbf{q}}$, being consistent with the acceleration-resolved framework [8].

A QP is useful since we can define objectives as Euclidean norms. For example, a commonly used cost function is that of a tracking control objective:

$$f_o(\mathbf{x}) = \frac{1}{2}\left\|k(\mathbf{e}_\text{des} - \mathbf{e}) + b(\dot{\mathbf{e}}_\text{des} - \dot{\mathbf{e}}) + (\ddot{\mathbf{e}}_\text{des} - \ddot{\mathbf{e}})\right\|^2, \tag{12}$$

where $\mathbf{e}_\text{des}, \dot{\mathbf{e}}_\text{des}, \ddot{\mathbf{e}}_\text{des}$ define a desired trajectory in the task space and k, b are gains to tune. Note that this is a design choice. This corresponds to the design choice of $\dot{\mathbf{e}} = -\lambda\mathbf{e}$ in the *basic* visual servoing control law of (2). The advantage of (12) is that it corresponds to a mass-spring-damper system and can be tuned as such. For example, normally we set $b = 2\sqrt{k}$ for a critically damped behavior. A particular case of (12) consists in positioning all the joints such that $\mathbf{e} = \mathbf{q}$ and $\mathbf{J_e} = \mathbf{I}$, along with defining $\dot{\mathbf{e}}_\text{des} = \ddot{\mathbf{e}}_\text{des} = \mathbf{0}$ so that:

$$f_o(\mathbf{x}) = \frac{1}{2}\left\|k(\mathbf{q}_\text{des} - \mathbf{q}) - b\dot{\mathbf{q}} - \ddot{\mathbf{q}}\right\|^2. \tag{13}$$

We will refer to this as a posture task similar to [8]. For humanoid robots having many joints, this is used as a low-priority task to make sure the QP solution is unique.

For visual servoing, we substitute (7), (8) into (12) to recover the form of (10) where:

$$\begin{aligned} \mathbf{Q} &= \mathbf{J_p}^\top\mathbf{L_e}^\top\mathbf{L_e}\mathbf{J_p}, \\ \mathbf{c} &= -\mathbf{J_p}^\top\mathbf{L_e}^\top(k(\mathbf{e}_\text{des} - \mathbf{e}) + b(\dot{\mathbf{e}}_\text{des} - \mathbf{L_e}\mathbf{J_p}\dot{\mathbf{q}}) \\ &\quad + \ddot{\mathbf{e}}_\text{des} - \mathbf{L_e}\dot{\mathbf{J}}_\mathbf{p}\dot{\mathbf{q}} - \dot{\mathbf{L}}_\mathbf{e}\mathbf{J_p}\dot{\mathbf{q}}). \end{aligned} \tag{14}$$

Note that this differs from (3) because the control law is used as an objective function instead of a constraint. However, if we use a slack variable, $\mathbf{s}$, in (3) for relaxing the constraint, then the constraint effectively becomes an objective:

$$\begin{aligned} \mathbf{v} = \underset{\mathbf{v},\mathbf{s}}{\text{argmin}} \quad & \|\mathbf{v}\|^2 + w\,\|\mathbf{s}\|^2 \\ \text{subject to} \quad & \widehat{\mathbf{L_e}}\mathbf{v} - \mathbf{s} = -\lambda\mathbf{e}, \end{aligned} \tag{15}$$

where w is a weight used to adjust the priority. The slack variable *trick* can be applied to any constraint, including inequalities [22]. So the difference between an objective and constraint is effectively only the priority. The design choice of lessening the importance of the visual servoing solution fits humanoid robots that already have several constraints that are more important. Another change is the use of the posture task (13) in place of the velocity norm objective. Lastly, note that adding several independent QP objectives together results in the same QP form of (10) so (15) can be written as a QP similarly to (3).

B. Particularities of visual servoing

To use (14), we need to detail some variables. Firstly, $\mathbf{e}$ is defined as one of the visual servoing features from the literature - e.g., point, line, circle, image moments, luminance, etc. [13]. These come together with a definition of the corresponding interaction matrix, $\mathbf{L_e}$. Recall that we can *stack* the features and Jacobians [1]. Although this is possible, a better way in the optimization framework is to define a separate task. This allows better handling of prioritization - whether weights [24], a hierarchy [25], or both are used. Next, a robot body part, $\mathbf{p}$, (or an associated surface) is selected to be the servo end point. This similarly comes with the Jacobian, $\mathbf{J_p}$. Note that a slight modification of the Jacobian is needed in the case of eye-to-hand systems as opposed to eye-in-hand systems that servo the camera body as illustrated in [26]. Lastly, we are missing the definition of $\dot{\mathbf{L}}_\mathbf{e}$. An approximation can be made that the term $\dot{\mathbf{L}}_\mathbf{e}\mathbf{J_p}\dot{\mathbf{q}}$ is negligible in the context of (14). However, it is possible to obtain $\dot{\mathbf{L}}_\mathbf{e}$ as shown next.

One of the most common and simplest *image-based* features is the point. It is defined by:

$$\mathbf{e} = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} X/Z \\ Y/Z \end{bmatrix}, \tag{16}$$

where the 3D coordinates of the point in reference to the camera frame are $\{X, Y, Z\}$, where Z is the depth. The equivalent pixel-space coordinates are easily obtained with the camera intrinsic parameters [1]. Its corresponding Jacobian is:

$$\mathbf{L_e} = \begin{bmatrix} \frac{-1}{Z} & 0 & \frac{x}{Z} & xy & -(1+x^2) & y \\ 0 & \frac{-1}{Z} & \frac{y}{Z} & 1+y^2 & -xy & -x \end{bmatrix}. \tag{17}$$

Taking the time derivative we get:

$$\dot{\mathbf{L}}_\mathbf{e} = \begin{bmatrix} \frac{\dot{Z}}{Z^2} & 0 & \frac{\dot{x}Z - x\dot{Z}}{Z^2} & \dot{x}y + x\dot{y} & -2x\dot{x} & \dot{y} \\ 0 & \frac{\dot{Z}}{Z^2} & \frac{\dot{y}Z - y\dot{Z}}{Z^2} & 2y\dot{y} & -\dot{x}y - x\dot{y} & -\dot{x} \end{bmatrix}. \tag{18}$$

The image point derivatives $\{\dot{x}, \dot{y}\}$ can be obtained from (7) while $\dot{Z}$ is obtained by:

$$\dot{Z} = \begin{bmatrix} 0 & 0 & -1 & -yZ & xZ & 0 \end{bmatrix} \mathbf{v}, \tag{19}$$

which comes from the spatial velocity definition.

Another example comes from the class of *pose-based* features. The relative translation feature is defined as:

$$\mathbf{e} = {}^\mathbf{d}\mathbf{t}_\mathbf{p}, \tag{20}$$

which is a 3-dim vector corresponding to the translation of the robot body part, p, with d, the target/desired frame used as the reference. Its Jacobian is:

$$\mathbf{L_e} = \begin{bmatrix} {}^\mathbf{d}\mathbf{R}_\mathbf{p} & \mathbf{0} \end{bmatrix}, \tag{21}$$

which contains the corresponding rotation matrix. The derivative of a rotation matrix can be associated to the angular velocity placed in skew-symmetric matrix form $[\boldsymbol{\omega}]_\times$ such that:

$$\dot{\mathbf{L}}_\mathbf{e} = \begin{bmatrix} {}^\mathbf{d}\dot{\mathbf{R}}_\mathbf{p} & \mathbf{0} \end{bmatrix} = \begin{bmatrix} (-[\boldsymbol{\omega}]_\times {}^\mathbf{d}\mathbf{R}_\mathbf{p}) & \mathbf{0} \end{bmatrix}. \tag{22}$$

III. Inequality Forms: Field of View Maintenance and Occlusion Avoidance

In visual servoing, we often want to formulate additional tasks relating to the visibility of features. These can be broadly classified into two. The first is to ensure that features remain within the field of view. In fact, a control law driving the feature error to zero does not explicitly prevent the features from leaving the field of view during the transition. Secondly, there are often other objects (or even the robot's other body parts) that have the possibility to block the field of view causing an occlusion, which we want to avoid. In these two cases, there is no specific target, making it difficult to formulate the problem as an equality, as the case in Sec. II. However, both of these can be easily formed as inequalities. Recall also that with slack variables the inequality does not need to be a strict constraint [22]. The question then becomes that of correctly prioritizing the different tasks, which we leave up to the designer of specific use cases.

Maintaining the field of view implies that the feature of interest, $\mathbf{e}_{\text{fov}}$, should remain inside some defined image bounds such that:

$$\underline{\mathbf{e}} \leq \mathbf{e}_{\text{fov}} \leq \overline{\mathbf{e}}, \tag{23}$$

where $\underline{\mathbf{e}}, \overline{\mathbf{e}}$ symbolizes the lower and upper limits respectively. Contrarily, avoiding occlusion can be formulated by defining $\mathbf{e}_{\text{occ}}$ as some features related to the occluding object. We then desire to keep it outside of certain image bounds such that:

$$\mathbf{e}_{\text{occ}} \leq \underline{\mathbf{e}} \qquad \textit{or} \qquad \overline{\mathbf{e}} \leq \mathbf{e}_{\text{occ}}. \tag{24}$$

These behaviors can be formulated similarly. Recall that inverting the inequality amounts to simply negating both sides and that the task vector can be stacked. So without loss of generality, the following explanations only show a single direction of the FoV constraint, that is: $\mathbf{e} \leq \overline{\mathbf{e}}$.

A. Base inequality formulation

Any appropriate task vector can be used to formulate an inequality, as it is with the objective function. Recall that we want to have a second-order form. For this, a second-order approximation can be defined:

$$\mathbf{e}_{\text{k+1}} = \mathbf{e}_{\text{k}} + \dot{\mathbf{e}}_{\text{k}}\Delta t + \frac{1}{2}\ddot{\mathbf{e}}_{\text{k}}\Delta t^2, \tag{25}$$

where Δt is a time step from discretization. We can then constrain this by:

$$\mathbf{e}_{\text{k+1}} \leq \overline{\mathbf{e}}. \tag{26}$$

We can get the linear form needed by first substituting (25) into (26). Doing so also removes the need for the subscript k, which we drop to be concise. Next, we use (7, 8) in (25) to obtain the joint space expression. Finally, recalling that we use $\mathbf{x} = \ddot{\mathbf{q}}$, we recover the form of (11) where:

$$\begin{aligned}\mathbf{A} =& \frac{1}{2}\mathbf{L_e}\mathbf{J_p}\Delta t^2,\\ \mathbf{b} =& \overline{\mathbf{e}} - \mathbf{e}_{\text{k}} - \mathbf{L_e}\mathbf{J_p}\dot{\mathbf{q}}\Delta t - \frac{1}{2}\Delta t^2\left(\mathbf{L_e}\dot{\mathbf{J}}_\mathbf{p}\dot{\mathbf{q}} + \dot{\mathbf{L}}_\mathbf{e}\mathbf{J_p}\dot{\mathbf{q}}\right).\end{aligned} \tag{27}$$

Almost all other terms were detailed before. The only thing left to define are the limits, in this case $\overline{\mathbf{e}}$. For example, let us first define $\mathbf{e}$ to be image points as in (16). This is a common and versatile definition because it can be extended by sampling the object of interest with several different points. Since we are concerned with visibility, the limits are best described in pixel space. For example, to define $\overline{x}$ such that it corresponds to the image border in pixel space, $\overline{u}$, we have:

$$\overline{x} = \frac{\overline{u} - c_x}{f_x}, \tag{28}$$

where c_x is the principal point, f_x is the focal length, both of which are obtained from calibration of the intrinsic camera parameters. This can be done similarly for y and lower limits.

Finally, note that (26) can be viewed as a 1-step preview horizon, making it similar to the form used in [18]. Because of this, it also has the same disadvantages of not being very stable in a numerical sense. The solution in [18] is to extend the preview horizon. This improves performance but has the disadvantage of extra computational cost (particularly since the number of constraints is increased). Differently, we improve the performance by slightly reforming (26) as shown next.

B. Augmenting the behavior by avoidance functions

Usually it is better to avoid a hard constraint rather than wait for it to activate (often violently). We can replace the hard constraint in (26) by an avoidance function, $f(\mathbf{e}, \overline{\mathbf{e}})$. Another improvement can be made by constraining only the update (i.e., velocity and acceleration). Using both we have:

$$\dot{\mathbf{e}}_{\text{k}}\Delta t + \frac{1}{2}\ddot{\mathbf{e}}_{\text{k}}\Delta t^2 \leq f(\mathbf{e}, \overline{\mathbf{e}}). \tag{29}$$

Using a parametrization of the avoidance function commonly used in [8]:

$$\dot{\mathbf{e}}_{\text{k}}\Delta t + \frac{1}{2}\ddot{\mathbf{e}}_{\text{k}}\Delta t^2 \leq \xi\frac{\mathbf{e}_{\text{k}} - \mathbf{e}_{\text{s}}}{\mathbf{e}_{\text{i}} - \mathbf{e}_{\text{s}}} - \xi_{\text{off}} \quad \text{if } \mathbf{e}_{\text{k}} > \mathbf{e}_{\text{i}}, \tag{30}$$

where $\mathbf{e}_{\text{s}}$ is a *safety* bound, such that $\mathbf{e}_{\text{s}} = \overline{\mathbf{e}} - \delta$, while $\mathbf{e}_{\text{i}}$ is an *interactive* bound which defines the activation boundary of the constraint, so $\mathbf{e}_{\text{i}} = \overline{\mathbf{e}} - i$ where $i > \delta$, finally ξ is a tunable gain for the avoidance behavior and a small offset ξ_{off} ensures that the robot is moving away from the constraint direction instead of keeping still (zero update). We can now change the QP with this. Note that $\mathbf{A}$ retains its form. Then:

$$\begin{aligned}\mathbf{b} =& \xi\frac{\mathbf{e}_{\text{k}} - \mathbf{e}_{\text{s}}}{\mathbf{e}_{\text{i}} - \mathbf{e}_{\text{s}}} - \xi_{\text{off}} - \mathbf{L_e}\mathbf{J_p}\dot{\mathbf{q}}\Delta t\\ & - \frac{1}{2}\Delta t^2\left(\mathbf{L_e}\dot{\mathbf{J}}_\mathbf{p}\dot{\mathbf{q}} + \dot{\mathbf{L}}_\mathbf{e}\mathbf{J_p}\dot{\mathbf{q}}\right).\end{aligned} \tag{31}$$

Finally, note that the constraint activation condition needs to be handled explicitly in the implementation, by adding the constraint when $\mathbf{e}_{\text{k}} > \mathbf{e}_{\text{i}}$ and removing it otherwise.

IV. Simulation results

This section shows some representative examples of our verification tests with simulations using the HRP-4 robot model with a 5 ms control loop. These are shown as part of the accompanying video (sped up only due to video length). In all of these, we have some essential whole-body control tasks in addition to the visual servoing tasks described. The required tasks can be generally described as:

- maintaining balance (e.g., dynamics consistency, center of mass control)
- actuation limits (e.g., joint position and torque limits)
- maintaining contacts (e.g., null contacting body acceleration, keeping within the static friction cone, unilaterality of force)
- self-collision avoidance
- default posture task (e.g, Eq.(13))

Because whole-body control is still a very active area of research, different teams use various formulations as can be seen in some examples from the literature [6]–[12]. Here, we are using the same formulation as [8]. The objective functions are combined using a weighted sum:

$$f_{\text{total}}(\mathbf{x}) = \sum_{i=1}^{n} w_{\text{i}} f_{\text{i}}(\mathbf{x}). \tag{32}$$

A guideline to tune the weights to produce a pseudo-hierarchy is:

$$w_{\text{i}}\ \min(f_{\text{i}}(\mathbf{x})) > w_{\text{i-1}}\ \max(f_{\text{i-1}}(\mathbf{x})), \tag{33}$$

where task i has a higher priority than task $i-1$, and min() and max() represent minimal and maximal function values according to the expected/acceptable task error values. Typically, we have prioritized: (1) center of mass control, (2) visual servoing, and (3) posture, for the objectives while actuation limits, maintaining contacts, dynamics consistency and self-collision avoidance are explicit constraints. The interested reader can refer to [8] for more implementation and technical details on the QP (e.g., solvers and runtime). However, the visual servoing tasks presented here can be adapted easily to fit with other optimization-based whole-body control frameworks. Furthermore, a strict hierarchy [25] is also possible instead of (32) and (33).

A. Gaze with occlusion avoidance

In this simulation, we show how the gaze can be controlled simply by centering a single point feature, defined by (16), in the image. Since the feature is defined in image space without extrapolating the object pose, the method falls into Image-Based Visual Servoing (IBVS). Fig. 1 shows this demonstration. The feature is the 2D projection of the user-controlled interactive marker. Additionally, a simulated wall serves as an occluding object to the left of the robot. For simplicity, we assume prior knowledge of its location. To define the occluding features, the wall edge closest to the robot is sampled with point features. The task limits are the edge of the image border (this is then augmented with the safety and interactive margins). In the accompanying video, the gaze control motion without occlusion is shown first by moving the object to the right of the robot. This corresponds to about 3 to 8 sec of the plot in Fig. 2. Contrarily, moving the object to the left of the robot can result in an occlusion by the wall as seen in Fig. 1. This corresponds to around 9 sec onwards of the plot in Fig. 2. The occlusion avoidance forces the robot to *lean forward*, noticeably using the torso and leg joints to gaze while maintaining balance (Center of Mass control). Additionally note how a small part of the wall enters the bottom-left corner of the simulated image inset in Fig. 1 (gray triangle on the bottom left). This portion is in between the sampled points. Although adding more points can always be done, it is also possible to use other image features such as line segments, that can better represent the object.

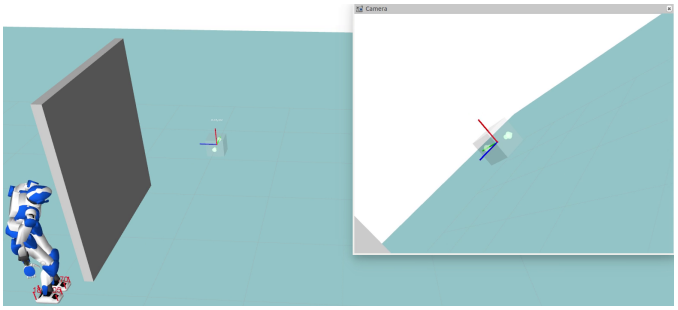

Fig. 1. A demonstration showing a gazing task using IBVS.

B. Hand servoing with modeling errors

Visual servoing is well-known for its robustness to some modeling errors. To show that this is still the case, we simulated a constant offset of 0.5 radian on the right shoulder pitch. This can be seen in Fig. 3. There are two different robot models here. The opaque one is the *real* robot model. The transparent one is the *wrong* robot model which has the offset in the right arm pose representing the mistaken internal knowledge. In this simulation, we used the *wrong* information for all the computations requiring the robot model, e.g., the robot Jacobians in (14). However, we update the visual servoing task (error and feature Jacobians) with the *real* pose, simulating the information provided from a visual pose estimate. For the control, we use Pose-Based Visual Servoing (PBVS) on the right wrist with the translation feature of (20) and a

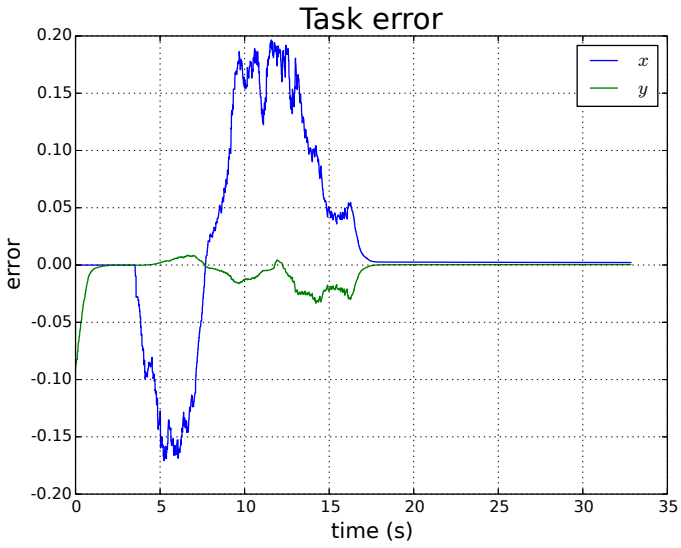

Fig. 2. Simulation data of the IBVS gazing task error discussed in Sec.IV-A

corresponding angle-axis orientation feature [1]. Note how in the accompanying video, the real robot model converges to the desired pose instead of the wrong robot model, which would have been the case if we used an *open-loop* method. Fig. 3 is representative of this, where the interactive marker was moved. The PBVS task errors of this demonstration are shown in Fig. 4.

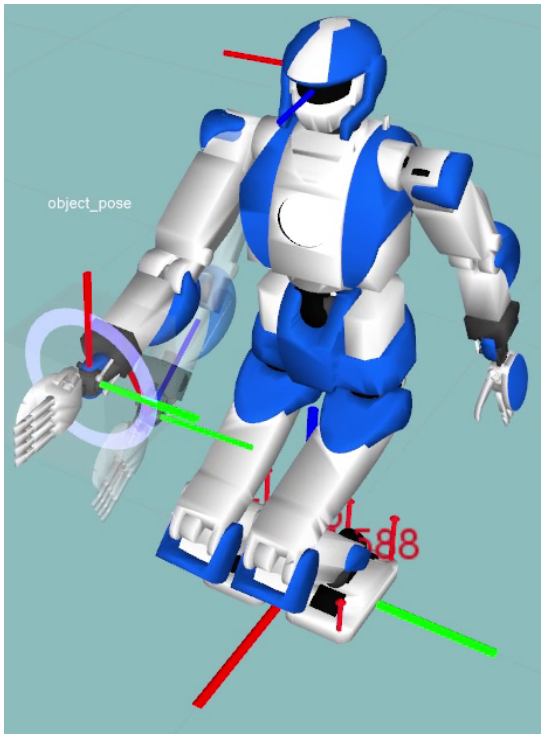

Fig. 3. Final pose of a PBVS task for the right wrist with a simulated modeling error. The opaque robot represents the *real* robot model while the *wrong* robot model is transparent. The error is introduced in the right shoulder joint. The task error plot of Fig 4 shows convergence to the target.

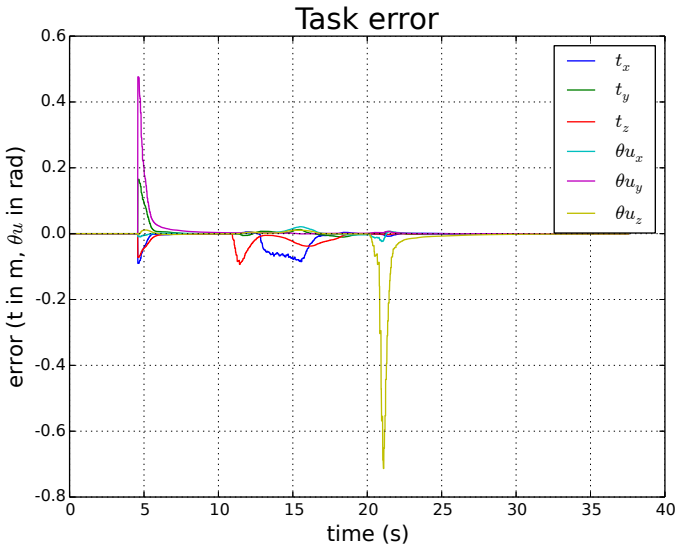

Fig. 4. Simulation data of the PBVS hand task error discussed in Sec.IV-B, the goal was moved 3 times after initial convergence

C. Combining with walking

Walking (and other locomotion modes) amounts to controlling the floating base of the humanoid while maintaining *balance*. This implies that contact states (footsteps) are handled. In this demonstration, we are using a WPG implementation with a reference velocity as an input [27], [28]. To use this together with our visual servoing tasks, a simple but effective method consists of defining the WPG inputs as a function of the visual servoing task errors. For example, if we have a PBVS task for the right hand and an IBVS gaze task (as in the previous simulations) then we can define the WPG reference as:

$$\dot{c}_{\text{xref}} = k_v(t_{\text{xPBVS}}), \quad \dot{c}_{\text{yref}} = 0, \quad \dot{\theta}_{\text{ref}} = -k_\theta(\theta_{\text{gaze}}),$$

where the WPG input reference velocities are $\dot{c}_{\text{xref}}, \dot{c}_{\text{yref}}, \dot{\theta}_{\text{ref}}$, while t_{xPBVS} is the translation part of the hand PBVS task corresponding to the x axis of the WPG frame (local frame on the robot), θ_{gaze} is the yaw angle of the gaze frame relative to the current WPG frame and k_v, k_θ are gains to tune. The idea is that the hand PBVS will guide walking forward. Walking sideways is not used. The gaze IBVS orients the robot such that it faces straight at the object. Finally, note that bounds are needed for t_{xPBVS} and θ_{gaze} that are used in the coupling. When a bound is exceeded, we use: $\mathbf{e}' = \frac{\mathbf{e}}{\max(\mathbf{e})}$ where $\mathbf{e}$ is the unbounded error, $\max(\mathbf{e})$ is the largest limit violation and $\mathbf{e}'$ is the result used. This preserves the vector direction. Fig. 5 shows a screenshot from the demonstration. Fig. 6 shows the relevant WPG control inputs: $\dot{c}_{\text{xref}}, \dot{\theta}_{\text{ref}}$. The start of the plot of Fig. 6 from time 0 to around 4 sec shows the clipping of the control due to the bound of 0.3 m/s on $\dot{c}_{\text{xref}}$. After this, up to around 35 sec, we show how the tasks converged to a fixed goal. Next, from around 35 sec onwards, we see how the tasks converged when the interactive goal was moved. Furthermore, there is a small oscillation (especially when the task is close to converging). This is due to the conflicting tasks (since we are not using strict hierarchies). Specifically, the visual servoing task designed in this paper conflicts with the Center of Mass servoing with references generated by the WPG. Although the coupling laws designed here seek to resolve this conflict, having a separate solver for the WPG means it cannot be fully resolved in this manner.

Although this ad hoc coupling is effective in this demonstration, it has some clear drawbacks. The purpose here was to show further that visual servoing can be used as just another whole-body task implying that it can work together seamlessly with balance control and changing contact states. For a more rigorous integration of walking, we think that methods such as [21] which consider all the conflicting tasks together could be a more suitable approach to the problem. However, since this requires a larger effort, we have opted to leave this out of the scope of this work.

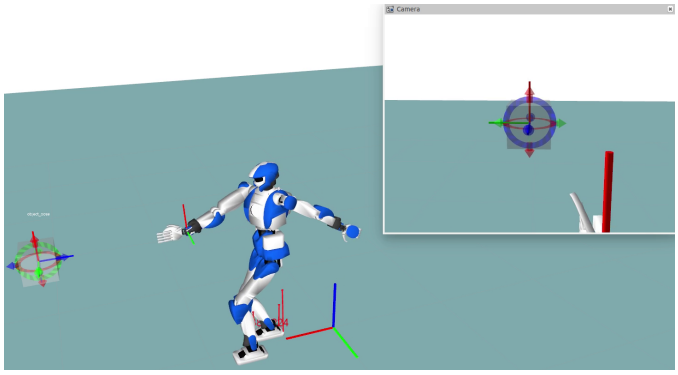

Fig. 5. A demonstration showing walking together with a hand PBVS task and an IBVS gaze

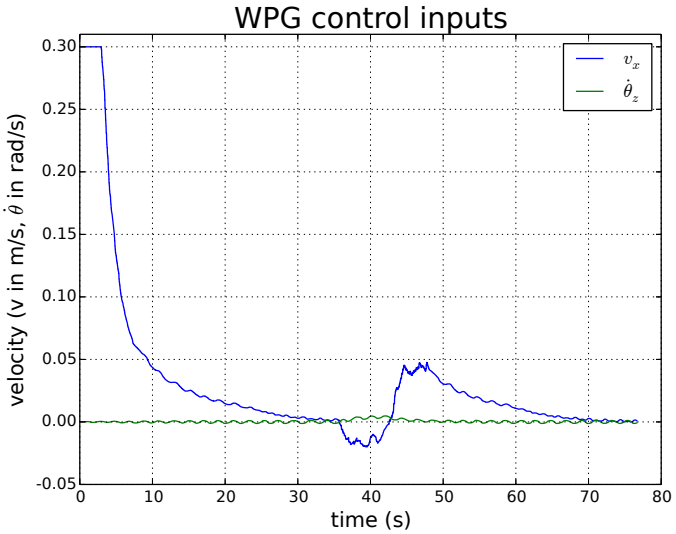

Fig. 6. Simulation data of the walking pattern generator velocity inputs coming from the visual servoing errors discussed in Sec.IV-C

V. Tests on a real platform

For validation on a real robot, we are using Romeo, a 37 DOF humanoid from SoftBank Robotics. In these tests, we used the left eye camera with a resolution of 320x240 at 30 Hz.

For the interface to Romeo, we used a velocity controller provided by SoftBank Robotics running at 10 Hz. The computed acceleration commands are numerically integrated to provide the required velocity commands.

A. Gaze control with Romeo

In this test, Romeo detects and tracks a moving visual target where an IBVS gaze task is used to keep the target in the center of the image (see Fig. 7). In this case, the circular target is a single marker from the open-source marker-based localization system called WhyCon [29]. Fig. 8 shows the IBVS task error throughout the test. We can see that the absolute task error always remains less than 0.3 despite the target being moved (in this case, no feedforward prediction of the motion was added either). Because of modeling errors, (i.e. intrinsic and extrinsic camera parameter errors and Romeo's mechanical model errors), we had to prevent overshooting by increasing the damping ratio of our control law (12) from 1 (used in simulations for a critically damped convergence) to 2.7, such that $b = 5.4\sqrt{k}$. Furthermore, along with the common tasks listed at the beginning of Sec. IV, two tasks were added for show. Although these do not help the visual servoing tasks in any way, they subjectively improve what appears in the video and screenshots. Firstly, an orientation task for the head is added to minimize the *unnatural-looking* head tilting. Secondly, another IBVS task is defined so that the right eye tracks the target as well (the eyes are independently actuated).

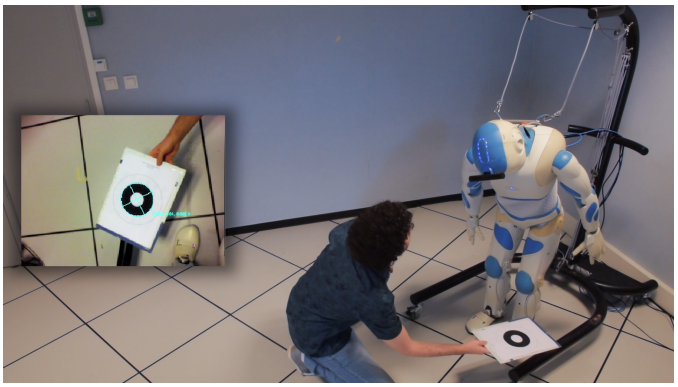

Fig. 7. A demonstration showing gaze control with Romeo using IBVS.

B. Grasping with Romeo

In this test, Romeo has to grasp a box (see Fig. 9). To do this, PBVS is used to move the hand towards a desired pose that is defined relative to the pose of the box. The box is also moved after the initial convergence, as seen in the video as well as the plot of Fig. 10 after around 15 sec. Simultaneously, IBVS is used for the gaze, as in Sec. V-A, this time keeping a point relative to both the box and hand as a target, similar to [3], [4]. Here, it is necessary to visually track both the hand and box. The ViSP [13] library is used for these. Specifically, we used the ViSP blob detection and pose estimation algorithm on the hand marker and the Model Based Tracker (MBT) for the box.

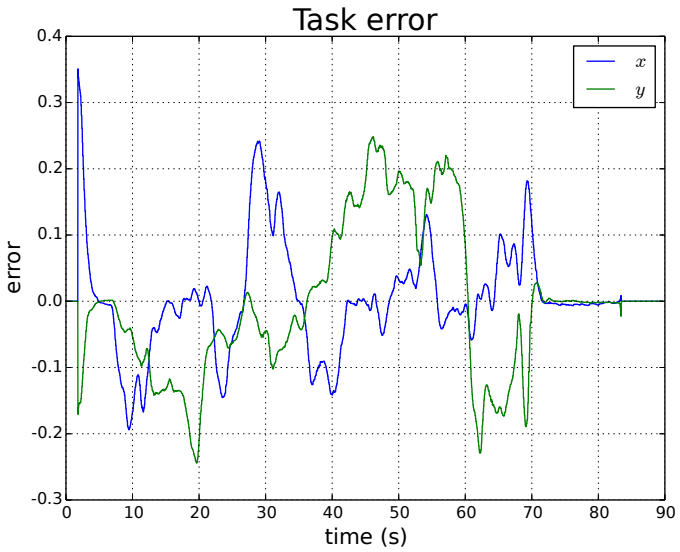

Fig. 8. Data from Romeo of the IBVS gazing task error discussed in Sec.V-A

This demonstration turned out to be particularly challenging with Romeo. Throughout the tests, we faced a problem of reduced stiffness in the knee joints, imposed by an internal temperature protection of the motors, to which we do not have access. This caused unwanted oscillations whenever it occurs. This can be noticed in the accompanying video towards the end of the grasping demonstration. This adds even more perturbations (on top of the camera and mechanical modeling errors). In spite of these, visual servoing made it possible for Romeo to succeed in the tasks.

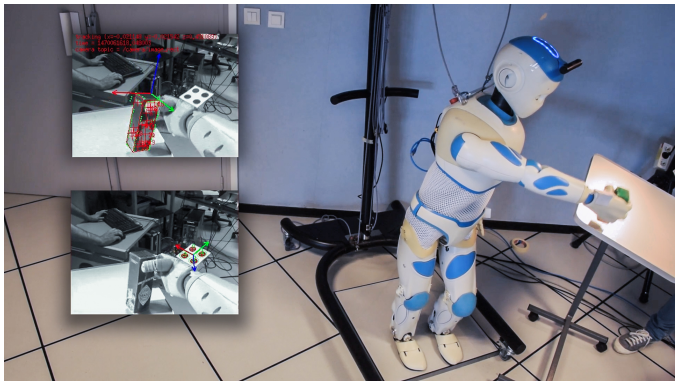

Fig. 9. A demonstration showing Romeo grasping a box using IBVS for the gaze and PBVS for servoing the hand.

VI. Conclusion and Future Work

In this paper, we showed how visual servoing can be formulated as a quadratic optimization problem. Furthermore, we formulated both equality and inequality constraints. Moreover, by defining an acceleration resolved form, we were able to easily integrate the visual servoing tasks into an existing whole-body control framework for humanoids. Results were then shown with simulations on HRP-4 and then real tests on Romeo.

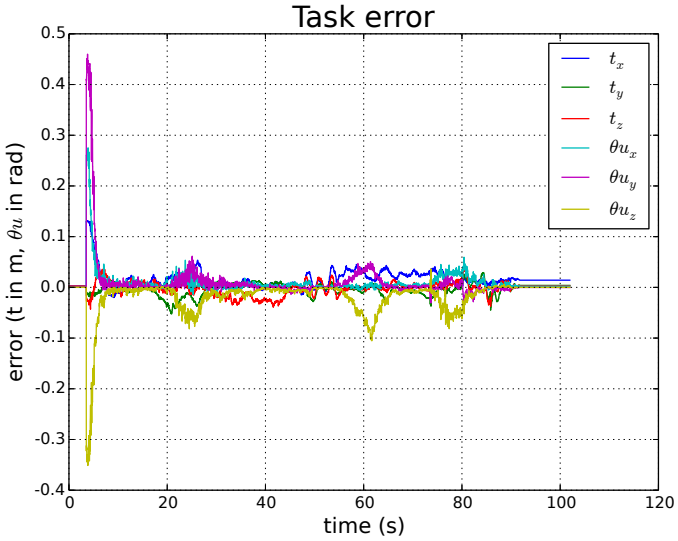

Fig. 10. Data from Romeo of the hand PBVS task error discussed in Sec.V-B

For future work, we can envision using the visual servoing tasks in more challenging scenarios. Some specific areas to be improved in this regard are: the combination with walking, and adding feedforward prediction for target motion tracking. Both of these were briefly outlined here. We also envision using other image features and visual constraints to improve the challenges on real robot platforms such as Romeo.

VII. Acknowledgement

This work was supported in part by the BPI Romeo 2 and the H2020 Comanoid projects (www.comanoid.eu).

References

[1] F. Chaumette and S. Hutchinson, "Visual servo control, Part I: Basic approaches," *IEEE Robotics and Automation Magazine*, vol. 13, no. 4, pp. 82–90, 2006.

[2] N. Mansard, O. Stasse, F. Chaumette, and K. Yokoi, "Visually-Guided Grasping while Walking on a Humanoid Robot," in *IEEE International Conference on Robotics and Automation*, pp. 3041–3047, April 2007.

[3] D. J. Agravante, J. Pagès, and F. Chaumette, "Visual servoing for the REEM humanoid robot's upper body," in *IEEE International Conference on Robotics and Automation*, pp. 5253–5258, May 2013.

[4] G. Claudio, F. Spindler, and F. Chaumette, "Vision-based manipulation with the humanoid robot Romeo," in *IEEE-RAS International Conference on Humanoid Robots*, pp. 286–293, Nov. 2016.

[5] J. Nocedal and S. Wright, *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering, Springer New York, 2000.

[6] L. Saab, O. E. Ramos, F. Keith, N. Mansard, P. Souères, and J. Y. Fourquet, "Dynamic Whole-Body Motion Generation Under Rigid Contacts and Other Unilateral Constraints," *IEEE Transactions on Robotics*, vol. 29, pp. 346–362, April 2013.

[7] J. Koenemann, A. D. Prete, Y. Tassa, E. Todorov, O. Stasse, M. Bennewitz, and N. Mansard, "Whole-body model-predictive control applied to the HRP-2 humanoid," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3346–3351, Sept 2015.

[8] J. Vaillant, A. Kheddar, H. Audren, F. Keith, S. Brossette, A. Escande, K. Bouyarmane, K. Kaneko, M. Morisawa, P. Gergondet, E. Yoshida, S. Kajita, and F. Kanehiro, "Multi-contact vertical ladder climbing with an HRP-2 humanoid," *Autonomous Robots*, vol. 40, no. 3, pp. 561–580, 2016.

[9] A. Herzog, N. Rotella, S. Mason, F. Grimminger, S. Schaal, and L. Righetti, "Momentum control with hierarchical inverse dynamics on a torque-controlled humanoid," *Autonomous Robots*, vol. 40, no. 3, pp. 473–491, 2015.

[10] S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, and R. Tedrake, "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot," *Autonomous Robots*, vol. 40, no. 3, pp. 429–455, 2016.

[11] M. Johnson, B. Shrewsbury, S. Bertrand, T. Wu, D. Duran, M. Floyd, P. Abeles, D. Stephen, N. Mertins, A. Lesman, J. Carff, W. Rifenburgh, P. Kaveti, W. Straatman, J. Smith, M. Griffioen, B. Layton, T. de Boer, T. Koolen, P. Neuhaus, and J. Pratt, "Team IHMC's Lessons Learned from the DARPA Robotics Challenge Trials," *Journal of Field Robotics*, vol. 32, no. 2, pp. 192–208, 2015.

[12] S. Feng, E. Whitman, X. Xinjilefu, and C. G. Atkeson, "Optimization-based Full Body Control for the DARPA Robotics Challenge," *Journal of Field Robotics*, vol. 32, no. 2, pp. 293–312, 2015.

[13] É. Marchand, F. Spindler, and F. Chaumette, "ViSP for visual servoing: a generic software platform with a wide class of robot control skills," *IEEE Robotics and Automation Magazine*, vol. 12, no. 4, pp. 40–52, 2005.

[14] Y. Mezouar and F. Chaumette, "Path planning for robust image-based control," *IEEE Transactions on Robotics and Automation*, vol. 18, pp. 534–549, Aug 2002.

[15] A. H. A. Hafez, A. K. Nelakanti, and C. V. Jawahar, "Path planning approach to visual servoing with feature visibility constraints: A convex optimization based solution," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1981–1986, Oct 2007.

[16] T. Shen and G. Chesi, "Visual Servoing Path Planning for Cameras Obeying the Unified Model," *Advanced Robotics*, vol. 26, no. 8-9, pp. 843–860, 2012.

[17] M. Kazemi, K. K. Gupta, and M. Mehrandezh, "Randomized Kinodynamic Planning for Robust Visual Servoing," *IEEE Transactions on Robotics*, vol. 29, pp. 1197–1211, Oct 2013.

[18] G. Allibert, E. Courtial, and F. Chaumette, "Predictive Control for Constrained Image-Based Visual Servoing," *IEEE Transactions on Robotics*, vol. 26, pp. 933–939, Oct 2010.

[19] M. Garcia, O. Stasse, J.-B. Hayet, C. Dune, C. Esteves, and J.-P. Laumond, "Vision-guided motion primitives for humanoid reactive walking: Decoupled versus coupled approaches," *International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 402–419, 2015.

[20] 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 *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4489–4496, Oct 2010.

[21] A. Sherikov, D. Dimitrov, and P.-B. Wieber, "Whole body motion controller with long-term balance constraints," in *IEEE-RAS International Conference on Humanoid Robots*, pp. 444–450, Nov 2014.

[22] O. Kanoun, F. Lamiraux, and P. B. Wieber, "Kinematic Control of Redundant Manipulators: Generalizing the Task-Priority Framework to Inequality Task," *IEEE Transactions on Robotics*, vol. 27, pp. 785–792, Aug 2011.

[23] E. Malis, "Improving vision-based control using efficient second-order minimization techniques," in *IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1843–1848 Vol.2, April 2004.

[24] K. Bouyarmane and A. Kheddar, "Using a multi-objective controller to synthesize simulated humanoid robot motion with changing contact configurations," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4414–4419, Sept 2011.

[25] A. Escande, N. Mansard, and P.-B. Wieber, "Hierarchical quadratic programming: Fast online humanoid-robot motion generation," *The International Journal of Robotics Research*, 2014.

[26] F. Chaumette and S. Hutchinson, "Visual servo control, Part II: Advanced approaches," *IEEE Robotics and Automation Magazine*, vol. 14, no. 1, pp. 109–118, 2007.

[27] D. J. Agravante, A. Sherikov, P. B. Wieber, A. Cherubini, and A. Kheddar, "Walking pattern generators designed for physical collaboration," in *IEEE International Conference on Robotics and Automation*, pp. 1573–1578, May 2016.

[28] A. Herdt, H. Diedam, P.-B. Wieber, D. Dimitrov, K. Mombaur, and M. Diehl, "Online walking motion generation with automatic footstep placement," *Advanced Robotics*, vol. 24, no. 5-6, pp. 719–737, 2010.

[29] T. Krajník, M. Nitsche, J. Faigl, P. Vaněk, M. Saska, L. Přeučil, T. Duckett, and M. Mejail, "A Practical Multirobot Localization System," *Journal of Intelligent & Robotic Systems*, vol. 76, no. 3, pp. 539–562, 2014.