Visual Servoing for the REEM Humanoid Robot's Upper Body

Don Joven Agravante, Jordi Pagès and François Chaumette

Abstract— **In this paper, a framework for visual servo control of a humanoid robot's upper body is presented. The framework is then implemented and tested on the REEM humanoid robot. The implementation is composed of 2 controllers - a head gaze control and a hand position control. The main application is precise manipulation tasks using the hand. For this, the hand controller takes top priority. The head controller is designed to keep both the hand and object in the eye field of view. For robustness, a secondary task of joint limit avoidance is implemented using the redundancy framework and a large projection operator proposed recently. For safety, joint velocity scaling is implemented. The implementation on REEM is done using the ROS and ViSP middleware. The results presented show simulations on Gazebo and experiments on the real robot. Furthermore, results with the real robot show how visual servoing is able to overcome some deficiency in REEM's kinematic calibration.**

I. INTRODUCTION

Humanoid robots are designed with a human form-factor. The desire for this kind of design is rooted in the need to bring robots into the everyday environment. Not only is the robot's form important, but also the method of control. The robot's actions need to be tightly coupled with its perception of the environment. Visual servoing is a form of control that directly incorporates visual information into the control loop [1], [2]. Incorporating visual feedback makes it robust to errors that may come from a lack of calibration, system noise or the nature of the environment. For these reasons, visual servoing is an attractive choice for performing manipulation tasks as opposed to open-loop approaches such as the one described in [3].

Early works on using visual servoing for enabling a robot to grasp objects are reported in [4] and [5]. Since then, it has been utilized in more complex humanoid robots. For the HRP-2, visual servoing was used to grasp a ball while walking in [6] and to control dynamic walking in [7]. In [8], a complete system is described - the visual servoing control law, object detection and localization and end-effector design of a custom-built humanoid robot. For the ARMAR III humanoid, visual information is used in conjunction with information from motor encoders and force sensors for a grasping task [9]. In [10], neural networks are used to learn the robot and image Jacobians instead of using their analytical models. These are then used in both open-loop and closed-loop approaches. The authors also explain the significance of both approaches and their coexistence in neuroscience.

In this paper, visual servoing is put in the context of anthropomorphic humanoid robots. A generalized framework is presented which is implemented in PAL Robotics' REEM to precisely position its right hand using the torso and arm joints. Furthermore, the head is controlled to keep the target object and hand in the field of view. Compared to previous implementations on humanoids, the controller described in this work utilizes the large projection operator [11] to take advantage of redundancy. Furthermore, joints shared between tasks (for example the torso) are explicitly handled in the control law through feedforward components.

The remaining sections of this article are organized as follows: Section II briefly recalls important concepts in visual servoing. In Section III the framework created is described generalizing it for any humanoid. Section IV details the implementation for REEM. The results of this implementation are presented in Section V. Finally Section VI concludes with a short summary and plans for future works.

II. VISUAL SERVOING

The most basic visual servo model and control law is shown by Eq. (1). The mathematical notation used throughout this article is adapted from [1], [12].

$$\begin{cases} \dot{\mathbf{e}} = \mathbf{L_e}\mathbf{v} \\ \mathbf{v} = -\lambda\widehat{\mathbf{L_e^+}}\mathbf{e} \end{cases} \tag{1}$$

The first equation in (1) shows the model - that $\dot{\mathbf{e}}$ (the change in the visual errors over time) is related to $\mathbf{v}$ (the camera velocity). This relation is given by the interaction matrix $\mathbf{L_e}$. To arrive to the second equation (the actual control law), an exponential decoupled decrease is designed such that $\dot{\mathbf{e}} = -\lambda\mathbf{e}$, and the Moore-Penrose pseudoinverse is used on $\mathbf{L_e}$, denoted by $\mathbf{L_e^+}$. However, in most cases only an estimate can be obtained [1]. This is represented by the "hat" in the notation, $\widehat{\mathbf{L_e^+}}$. Eq. (1) is easily extended into direct joint velocity control [12]. In addition, the redundancy framework is also introduced as follows:

$$\begin{cases} \dot{\mathbf{e}} = \mathbf{J_e}\dot{\mathbf{q}} \\ \dot{\mathbf{q}} = -\lambda\widehat{\mathbf{J_e^+}}\mathbf{e} + \mathbf{Pg} \end{cases} \tag{2}$$

A direct joint space control is obtained by using the well-known robot Jacobian, $\mathbf{J}(\mathbf{q})$, which relates $\dot{\mathbf{q}}$ (joint velocities) to $\mathbf{v}$. For example, a common eye-in-hand system

This work was supported in part by the Erasmus Mundus Grant from the European Commission

D.J. Agravante was with Heriot-Watt University (UK), Universitat de Girona (Spain) and Université de Bourgogne (France) as an MSc student of Erasmus Mundus Masters in Computer Vision and Robotics (VIBOT) and is now with CNRS-LIRMM donjoven.agravante@lirmm.fr

J. Pagès is with PAL Robotics, S.L. jordi.pages@pal-robotics.com

F. Chaumette is with INRIA Rennes - Bretagne Atlantique, IRISA Francois.Chaumette@irisa.fr

[12] defines the task Jacobian $\mathbf{J_e}$ as $\mathbf{J_e} = \mathbf{L_e}{}^c\mathbf{V}_N\mathbf{J}(\mathbf{q})$, where ${}^c\mathbf{V}_N$ (a velocity twist transformation matrix) is used to change the reference point between N (the reference of $\mathbf{J}(\mathbf{q})$) and c (the reference of $\mathbf{L_e}$, which is the camera frame). The derivation obtaining the second equation of Eq. (2) is similar to that in Eq. (1). The redundancy framework is added, with $\mathbf{P}$ (a projection operator) and $\mathbf{g}$ (a secondary task vector). The classical gradient projection method [13] can be used, leading to $\mathbf{P} = \mathbf{P_e} = (\mathbf{I}_n - \mathbf{J_e^+}\mathbf{J_e})$ with $\mathbf{I}_n$ the $n \times n$ identity matrix where n is the number of degrees of freedom (DOF). The secondary task can be used for a variety of applications such as joint-limit avoidance, obstacle avoidance and occlusion avoidance [14], [15], [16]. Here, only joint-limit avoidance is considered, which is important for safe and reliable operation. In [14], an approach to joint-limit avoidance is detailed, which is utilized here. Figure 1 illustrates the basic behavior of the cost function. The ends represent the joint limits. The middle is a "safe configuration" where joint limit avoidance is not active. Just outside this zone, is a transition area where a sigmoid function is used to slowly introduce the avoidance action. The last zone is where the avoidance action is in full effect. Furthermore, an adaptive gain is used to make sure that the joint limits are always avoided [14].

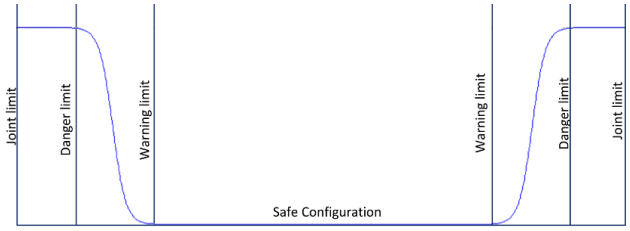


Fig. 1. Joint limit avoidance scheme

Recently, a new large projection operator was proposed to increase the availability of executing secondary tasks [11]. It relaxes the constraints of the classical projector which imposes that the exponential decrease of each term in the error vector is undisturbed. The new constraint is imposed such that only the exponential decrease of the norm needs to be undisturbed [11]. The definition is shown in Eq. (3)

$$\mathbf{P}_{||\mathbf{e}||} = \mathbf{I}_n - \frac{1}{\mathbf{e}^T\mathbf{J_e}\mathbf{J_e^T}\mathbf{e}}\mathbf{J_e^T}\mathbf{e}\mathbf{e}^T\mathbf{J_e} \tag{3}$$

However, a drawback of $\mathbf{P}_{||\mathbf{e}||}$ is the existence of a singularity when $\mathbf{e} \to \mathbf{0}$. To prevent any undesired effect, a switching scheme is used with the classical projector [11]. This is shown in Eq. (4).

$$\mathbf{P}_\lambda = \overline{\lambda}(||\mathbf{e}||)\ \mathbf{P}_{||\mathbf{e}||} + (1 - \overline{\lambda}(||\mathbf{e}||))\ \mathbf{P_e} \tag{4}$$

In Eq. (4), a sigmoid function, $\overline{\lambda}(||\mathbf{e}||)$ is used to smoothly transition from the large projection operator $\mathbf{P}_{||\mathbf{e}||}$ to the classical projection operator $\mathbf{P_e}$. This is such that when $||\mathbf{e}|| >> 0$ then $\overline{\lambda}(||\mathbf{e}||) = 1$ and only $\mathbf{P}_{||\mathbf{e}||}$ is active. Conversely, when $||\mathbf{e}||$ approaches 0 then $\overline{\lambda}(||\mathbf{e}||) = 0$ and only $\mathbf{P_e}$ is active.

III. VISUAL SERVOING ON A HUMANOID ROBOT

Before describing the hand control and gaze control for a humanoid robot, the important reference frames need to be identified. These are shown in Figure 2 together with the variable naming convention to be used in this paper.

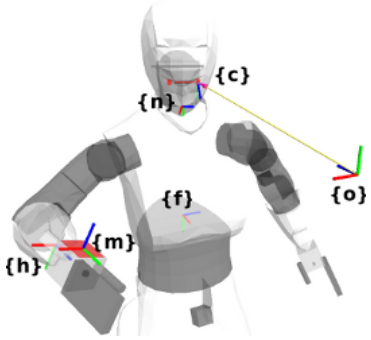


Fig. 2. Important coordinate frames on REEM

A. Hand Control

Precise hand control is necessary in any manipulation task. An "eye-to-hand" form of visual servoing is used for this. In this framework, all DOFs of the upper body which contribute to the hand motion are considered. This includes all the torso and arm joints. Doing this allows for a larger range of motion. These joints are collectively denoted by $\mathbf{q_h}$. The modeling and control law is then shown as follows:

$$\begin{cases} \dot{\mathbf{e}} = \mathbf{J_h}\dot{\mathbf{q}}_\mathbf{h} \\ \dot{\mathbf{q}}_\mathbf{h} = -\lambda\widehat{\mathbf{J}_\mathbf{h}^+}\mathbf{e} + \mathbf{P}_\lambda\mathbf{g} \end{cases} \tag{5}$$

Eq. (5) follows from the general form in Eq. (2). To continue describing the controller, a design choice is needed: which type of visual feature shall be used. Here, a position-based visual servoing (PBVS) is chosen that can produce a straight line trajectory in Cartesian space of a frame attached to the hand. For this, the 6-DOF pose of such a frame is necessary. Ideally, a vision algorithm that can obtain a pose estimate of the hand frame $\{h\}$ should be used. An alternative is presented here, using a visual marker on the hand (as in Fig 2). Here, augmented reality markers are utilized which allow for monocular pose estimation. This provides the pose of the marker $\{m\}$ in the camera reference frame $\{c\}$, which can be expressed as a homogeneous transformation matrix ${}^c\mathbf{M}_m$. Another requirement is the goal frame definition. For object manipulation, the goal is defined with respect to the object, such that ${}^o\mathbf{M}_{m^*}$ is known. An example of this is shown in Section IV. Lastly, it is required to have a pose estimate of the object frame $\{o\}$. Ideally, a vision algorithm that can obtain this should be used. As the main focus of this paper is on the control aspect, another visual marker is used to easily obtain ${}^c\mathbf{M}_o$. Using these information, the relative pose of the current and desired marker frame can be obtained by ${}^{m^*}\mathbf{M}_m = {}^o\mathbf{M}_{m^*}^{-1}\ {}^c\mathbf{M}_o^{-1}\ {}^c\mathbf{M}_m$. From this, the error ($\mathbf{e}$) and corresponding interaction matrix ($\mathbf{L_e}$) can be

defined from a known form [1]:

$$\mathbf{e} = ({}^{m^*}\mathbf{t}_m, {}^{m^*}\theta\mathbf{u}_m) \qquad \mathbf{L_e} = \begin{bmatrix} {}^{m^*}\mathbf{R}_m & \mathbf{0} \\ \mathbf{0} & \mathbf{L}_{\theta\mathbf{u}} \end{bmatrix}$$
$$\mathbf{L}_{\theta\mathbf{u}} = \mathbf{I}_3 - \frac{\theta}{2}[\mathbf{u}]_\times + \left(1 - \frac{sinc\ \theta}{sinc^2\frac{\theta}{2}}\right)[\mathbf{u}]_\times^2 \tag{6}$$

Eq. (6) defines a classical PBVS scheme [1]. The 6-vector error definition is composed of ${}^{m^*}\mathbf{t}_m$ (the translation) and ${}^{m^*}\theta\mathbf{u}_m$ (the axis-angle representation of the rotation). The 6×6 interaction matrix is mainly composed of ${}^{m^*}\mathbf{R}_m$ (the 3×3 rotation matrix), $[\mathbf{u}]_\times$ (the skew-symmetric matrix of the axis vector) and θ (the angle). All components are easily extracted from the 6-DOF pose. This PBVS scheme produces a 3D straight line trajectory from frame $\{m\}$ to the goal frame $\{m^*\}$ in perfect conditions [1]. This is the main reason for using this scheme. With the definitions from Eq. (6), $\mathbf{J_h}$ can now be defined as follows:

$$\mathbf{J_h} = \mathbf{L_e}{}^{\mathbf{m}}\mathbf{V_h}{}^{\mathbf{h}}\mathbf{J}(\mathbf{q_h}) \tag{7}$$

Usually an "eye-to-hand" configuration has a negative sign in the task Jacobian [12]. However, this only applies when the features are defined in the camera frame $\{c\}$. As they are defined in frame $\{m\}$, the negative sign is not present in Eq. (7). Furthermore, the robot Jacobian expressed in frame $\{h\}$ is available. So only ${}^{\mathbf{m}}\mathbf{V_h}$ is necessary to resolve the difference in reference frames. This is favorable as it is a constant matrix.

B. Gaze Control

A gaze control is needed to keep both the hand (marker) and object in the field of view. Since the arm and torso DOFs are used for hand control, the neck joints ($\mathbf{q_n}$) are used for this. To do this, the task consists of centering on the image the midpoint of the 2 frames: $\{m\}$ and $\{o\}$. An "eye-in-hand" image-based visual servoing (IBVS) is used here. To improve the controller's response, feed-forward terms are added from knowledge of the commanded torso and arm motion, which have an effect on the camera and hand poses. Two important effects are modeled from this. The first is the camera motion. It is caused by motion of the torso joints ($\mathbf{q_t}$) only. The second is the hand motion. It is caused by all the joints involved in the hand controller ($\mathbf{q_h}$). The complete model is shown as follows:

$$\dot{\mathbf{e}} = \mathbf{J_n}\dot{\mathbf{q}}_\mathbf{n} + \mathbf{J_t}\dot{\mathbf{q}}_\mathbf{t} + \mathbf{J_h}\dot{\mathbf{q}}_\mathbf{h} + \frac{\partial \mathbf{e}}{\partial t} \tag{8}$$

In Eq. (8), the primary error term is $\mathbf{J_n}\dot{\mathbf{q}}_\mathbf{n}$, which is the part that we want to control. The motion of the image feature resulting from camera motion due to the torso joints is modeled as $\mathbf{J_t}\dot{\mathbf{q}}_\mathbf{t}$. The motion of the image feature resulting from hand motion is modeled as $\mathbf{J_h}\dot{\mathbf{q}}_\mathbf{h}$. A final term ($\frac{\partial \mathbf{e}}{\partial t}$) is added which can be used to compensate other perturbations (i.e. unknown object motion). To obtain the control law from this model, a similar derivation from Eq. (2) is done resulting in:

$$\dot{\mathbf{q}}_\mathbf{n} = \widehat{\mathbf{J}_\mathbf{n}^+}(-\lambda\mathbf{e} - \widehat{\mathbf{J_t}\dot{\mathbf{q}}_\mathbf{t}} - \widehat{\mathbf{J_h}\dot{\mathbf{q}}_\mathbf{h}} - \widehat{\frac{\partial \mathbf{e}}{\partial t}}) + \mathbf{P}_\lambda\mathbf{g} \tag{9}$$

As before, only estimates can be obtained of the actual model parameters. To completely describe the control law in Eq. (9), the three task Jacobians ($\mathbf{J_n}, \mathbf{J_t}, \mathbf{J_h}$) need to be determined. These are given as follows:

$$\begin{cases} \mathbf{J_n} = \mathbf{L_e}{}^{\mathbf{c}}\mathbf{J}(\mathbf{q_n}) \\ \mathbf{J_t} = \mathbf{L_e}{}^{\mathbf{c}}\mathbf{J}(\mathbf{q_t}) \\ \mathbf{J_h} = -\frac{1}{2}\ \mathbf{L_e}{}^{\mathbf{c}}\mathbf{J}(\mathbf{q_h}) \end{cases} \tag{10}$$

In Eq. (10), the robot Jacobians are defined corresponding to the joints considered. Moreover, by defining these and $\mathbf{L_e}$ in the same camera frame $\{c\}$, no transform matrix $\mathbf{V}$ is necessary. The task Jacobians $\mathbf{J_n}$ and $\mathbf{J_t}$ are fairly straightforward "eye-in-hand" cases where camera motion is considered. $\mathbf{J_h}$ has a different form as it is derived from the midpoint motion. The 3D midpoint of the frames is obtained by $\mathbf{p}_{mid} = \frac{1}{2}(\mathbf{p}_{hand} + \mathbf{p}_{object})$. From this, the velocity of the midpoint is $\mathbf{v}_{mid} = \frac{1}{2}(\mathbf{v}_{hand} + \mathbf{v}_{object})$. Considering a stationary object, then $\mathbf{v}_{object} = 0$ also making $\widehat{\frac{\partial \mathbf{e}}{\partial t}} = 0$. With this, the midpoint velocity becomes $\mathbf{v}_{mid} = \frac{1}{2}\mathbf{v}_{hand}$. Since the midpoint velocity cannot be estimated directly, it is obtained from the hand velocity such that $\mathbf{v}_{mid} = \frac{1}{2}\mathbf{J}(\mathbf{q_h})\dot{\mathbf{q}}_\mathbf{h}$ where $\dot{\mathbf{q}}_\mathbf{h}$ is the result of the hand controller. Furthermore a negative sign is necessary since motion of the hand is considered, which is an "eye-to-hand" case. The projection of the 3D midpoint into the image is then used as the feature. To complete the description of the controller, the well-known definition of an image point IBVS scheme's $\mathbf{e}$ and $\mathbf{L_e}$ are shown in Eq. (11). Here, (x, y) is defined from the midpoint location and Z corresponds to its depth, which is known from the pose estimate of both target and hand.

$$\begin{cases} \mathbf{e} = (x, y) \\ \mathbf{L_e} = \begin{bmatrix} \frac{-1}{Z} & 0 & \frac{x}{Z} & xy & -(1+x^2) & y \\ 0 & \frac{-1}{Z} & \frac{y}{Z} & 1+y^2 & -xy & -x \end{bmatrix} \end{cases} \tag{11}$$

IV. IMPLEMENTATION DETAILS

The visual servoing framework has been implemented for REEM - PAL Robotics latest humanoid design, which is shown in Figure 3. It has a fully human form factor from the waist up and a wheeled mobile base to move around. The head has 2 neck joints to be used for the gaze control. For the hand controller: 2 DOFs are available in the torso and each arm has 7 DOFs. Here, only the right arm is considered.

Fig. 3. The humanoid robot REEM

The implementation for REEM is done completely in C/C++ within the framework of the Robotic Operating

System (ROS). The visual servoing block diagram is shown in Figure 4. In this figure, the different open-source software packages are overlaid onto the corresponding areas where they are used and represented as colored blocks.

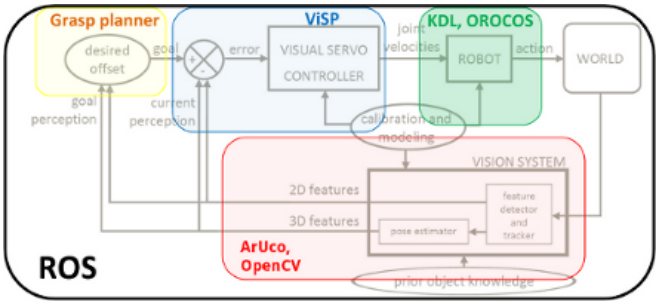


Fig. 4. The software architecture showing where the open-source packages are used

The most used library is ViSP [17] made by the Lagadic group of INRIA/IRISA. For the robot kinematics, the Kinematics and Dynamics Library (KDL) [18], from the Open Robot Control Software (OROCOS) project was used. In particular, KDL was mainly used to get the robot Jacobian. For the vision system, the open source Computer Vision library (openCV) [19] was used along with ArUco (an augmented reality library) [20]. Here, openCV was mainly used as a base - providing simple image processing and the camera calibration framework. ArUco is mainly used for the markers, providing a pose estimate of multiple markers from a monocular system. For now, the grasp planner is implemented as a simple look-up table of known objects and known pre-grasp poses.

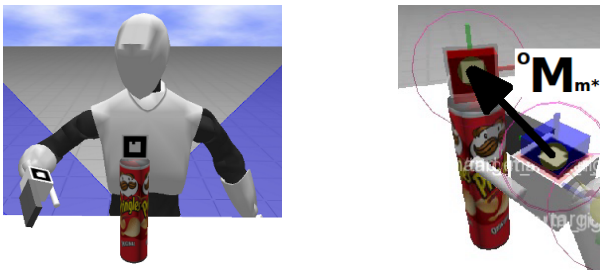
Over each controller, a "velocity scaling" is implemented. When one or more $\dot{q}$ of the initial result is over the limit, then the solution $\dot{\mathbf{q}}$ is scaled down to respect all limits. The result gives added safety to limit the movements while respecting the control law.

To safely test and verify the implementation, a simulation environment was created in Gazebo [21]. This consists of a REEM simulation model, a table, the object (a pringles can) and the markers. Fig. 5 (a) shows the start configuration used in most tests where both the object and hand are in the field of view. Figure 5 (b) shows that the goal is defined by ${}^{o}\mathbf{M}_{m^*}$, which is the homogeneous transformation matrix between the object marker frame $\{o\}$ and the hand marker desired frame $\{m^*\}$. The same figure also shows that a marker was placed on top of the object to facilitate detection and localization.

V. RESULTS

A. Simulation

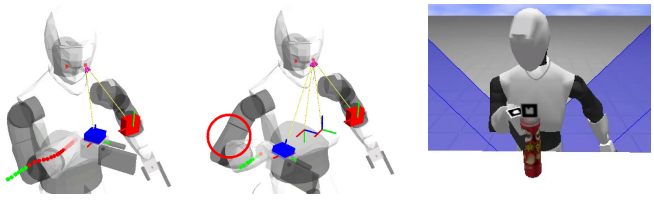
For the preliminary tests, the gazebo simulation was used. The task is to position the hand in a "pre-grasp" pose for the object by using visual information from the object and hand markers. The simulation is made to be as realistic as possible, where the individual motor controllers are simulated, and the marker detection algorithm is running in the loop. The first tests done were to verify the modeling and implementation.



(a) initial position (b) goal definition

Fig. 5. The pre-defined initial and goal pose for a pre-grasping task

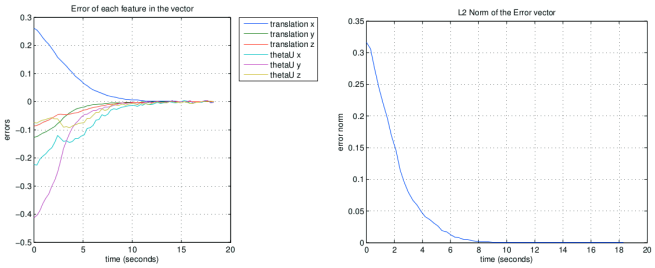
A big part of the framework is joint limit avoidance which is critical for robustness. To validate this portion, only the hand controller is running and it excludes the torso joints. A result with this is shown in Fig 6 where the trajectory of the hand (from start to goal) is shown in colored blocks. Green indicates that all joints are far from the limits, while red indicates that joint limit avoidance is activated. Fig 6 (a) shows the complete trajectory. Fig 6 (b) shows the robot pose at the time when the elbow joint nears its limit thus activating the avoidance algorithm. Lastly Fig 6 (c) shows that a satisfactory convergence pose is achieved despite the joint limit event.



(a) Trajectory (b) Elbow Joint limit (c) Convergence pose

Fig. 6. Simulation result with joint limit avoidance activated

To verify the results, the plots of the errors and error norm are presented. Fig 7 (a) shows that some of the errors increase around $t \approx 3sec$. This corresponds to the elbow joint nearing its limit as shown in Fig 6 (b). Even though some errors no longer follow an exponential decrease, this is maintained for the norm at all times, as shown in Fig 7 (b). This data shows the desired behavior from the large projection operator.



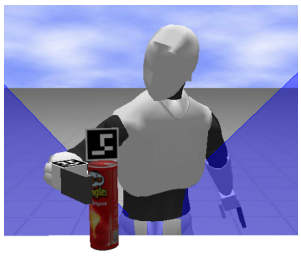
(a) errors (b) norm of the errors

Fig. 7. Plots of the errors showing joint limit avoidance effects

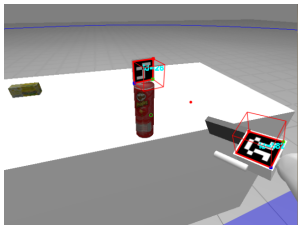
For the complete framework, several tests were done

including different start/end configurations. The results from a typical run are shown in Fig. 8. It shows that the goal pose is achieved by the hand. Furthermore, throughout the servoing process, the hand and object are kept within the field of view thanks to the gaze controller.

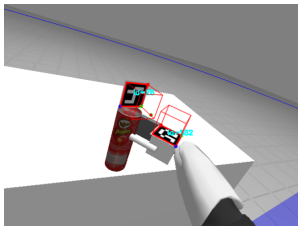
(a) initial robot pose



(b) final robot pose



(c) initial image input



(d) final image input

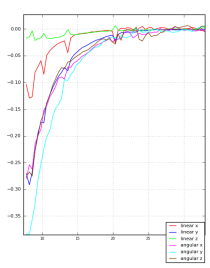
Fig. 8. Simulation results showing the initial and final pose along with the image used for visual servoing

Figure 8 shows that the upper body controller achieves what it was designed to do. For a more quantitative analysis of the design performance, data on the defined error function are presented in the graphs of Fig. 9. The hand controller errors in Fig. 9 (a) show that each component has an exponential decay (with some noise). This shows that the constraints of the hand positioning task are achieved. Fig 9 (b) shows the errors for the gaze controller which also have the desired exponential decay of each component. To show the action of the feedforward components in the gaze controller, the simulation scenario was repeated without these. The result is depicted in Fig 9 (c). It shows that the errors increase at the beginning. This is caused by the motion of the hand control task (mainly large torso motions). Although it is seen that after this, the controller can still drive the errors to zero.

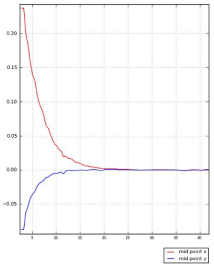
B. Real Experiments

After verification by simulation, similar scenarios were tested using the actual REEM humanoid (see accompanying video). The results of these experiments show a similar performance to the simulation runs. A typical experiment is shown in Fig. 10. These results show that a good qualitative performance is achieved with the implemented design.

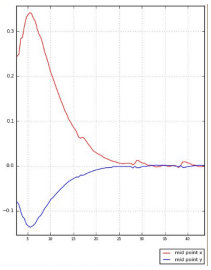
Again, data of the errors are gathered for analysis. These are shown in Figure 11. The graphs show a similar result from the simulation along with an observable increase in the noise. A lot of the vision algorithm performance is



(a) hand controller error



(b) gaze controller error (complete)



(c) gaze controller error without feedforward terms

Fig. 9. Plots of the error for the hand and gaze controllers (simulation)

(a) initial robot pose

(b) final robot pose

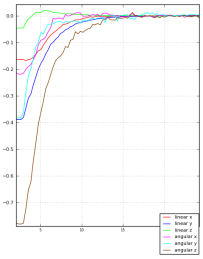
(c) initial image input

(d) final image input

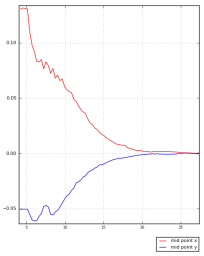
Fig. 10. Results of experiments on the actual robot, showing the initial and final pose along with the image used for visual servoing

unmodeled and can contribute significantly to the noise here.

One of the most important "real-world" effects found in the real experiments and not in the simulations is the robot calibration errors. These errors make the robot kinematic model different from the actual one. A common example is caused by the mechanical tolerances in the manufacture of parts. In the case of REEM, an example is shown where the motor encoders are not properly "zeroed" resulting in a significant bias. Figure 12 (a) shows the camera view of the actual robot hand overlaid by a "ghost" of the hand model which represents its internal knowledge (using the kinematic model and encoder information). On the same figure, the marker pose estimate from vision is shown on top of the actual robot hand as well as the expected marker pose of the model. Both are represented by a RGB coordinate frame. In the ideal case both marker frames would coincide (perfect model, sensor data). Fig. 12 (b) shows the discrepancy more clearly, showing only the robot model obtained from the internal sensors. On the figure, the red frame is the marker



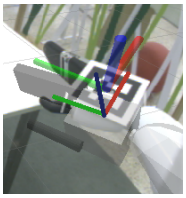
(a) hand controller error



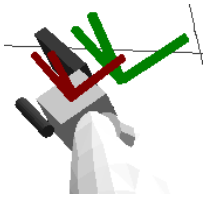
(b) gaze controller error (complete)

Fig. 11. Plots of the error for the hand and gaze controllers (experiments on the real robot)

on the model while the green frame is the visual estimate of the marker pose. The visual pose estimate is shown to be fairly accurate in Fig. 12 (a) indicating that errors exist in the internal knowledge. Even with this much error, the visual servoing algorithm achieves the desired performance.



(a) world view



(b) internal view

Fig. 12. The net effect of robot calibration errors on REEM's hand and the marker frame

VI. CONCLUSION

In this paper, a complete framework for visual servoing on the upper body of a humanoid robot has been presented. It is shown to be robust and performs well in both simulation and real experiments. The framework is composed of 2 controllers - a hand controller designed to achieve the manipulation task and a gaze controller designed to keep the hand and the object in the field of view. Furthermore, a joint limit avoidance scheme was implemented using a large projection operator that allows for more availability of the secondary task. A joint velocity scaling has also been added for safety. This system was implemented and tested on the REEM humanoid. The results show that visual servoing is able to overcome the calibration deficiency and shows satisfactory performance for accurate pre-grasp positioning.

The presented framework also sets the stage for building upon the tasks shown. An interesting continuation would be a framework for 2-handed manipulation. Incremental improvements to specific areas have also to be done. For example, having the same framework running without using markers by replacing it with visual hand and object detection and localization algorithms. Another is reducing the use of 3D information on the gaze controller. Furthermore, integration to high-level planning algorithms is also envisioned, such as the grasp planner outlined before. Another area of investigation is in improving the redundancy framework to create more human-like motions automatically.

REFERENCES

[1] F. Chaumette and S. Hutchinson, "Visual servo control, part i: Basic approaches," *IEEE Robotics and Automation Magazine*, vol. 13, pp. 82–90, December 2006.

[2] S. Hutchinson, G. Hager, and P. Corke, "A tutorial on visual servo control," *IEEE Transactions on Robotics and Automation*, vol. 12, pp. 651–670, 1996.

[3] R. Rusu, I. Sucan, B. Gerkey, S. Chitta, M. Beetz, and L. Kavraki, "Real-time perception-guided motion planning for a personal robot," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pp. 4245–4252, IEEE, 2009.

[4] R. Horaud, F. Dornaika, C. Bard, and B. Espiau, "Visually guided object grasping," *IEEE Transactions on Robotics and Automation*, vol. 14, pp. 525–532, 1995.

[5] D. Kragic and H. I. Christensen, "Model based techniques for robotic servoing and grasping," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'02*, (Lausanne, Switzerland), pp. 299 – 304, 2002.

[6] N. Mansard, O. Stasse, F. Chaumette, and K. Yokoi, "Visually-guided grasping while walking on a humanoid robot," in *IEEE Int. Conf. on Robotics and Automation, ICRA'07*, (Roma, Italy), pp. 3041–3047, April 2007.

[7] C. Dune, A. Herdt, E. Marchand, O. Stasse, P.-B. Wieber, and E. Yoshida, "Vision based control for humanoid robots," in *IEEE/RAS International Conference on Intelligent Robot and Systems (IROS), Workshop on Visual Control of Mobile Robots (ViCoMor)*, 2011.

[8] G. Taylor and L. Kleeman, *Visual Perception and Robotic Manipulation: 3D Object Recognition, Tracking and Hand-Eye Coordination (Springer Tracts in Advanced Robotics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

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

[10] L. Natale, F. Nori, G. Sandini, and G. Metta, "Learning precise 3d reaching in a humanoid robot," in *Development and Learning, 2007. ICDL 2007. IEEE 6th International Conference on*, pp. 324–329, IEEE, 2007.

[11] M. Marey and F. Chaumette, "A new large projection operator for the redundancy framework," in *IEEE Int. Conf. on Robotics and Automation, ICRA'10*, (Anchorage, Alaska), pp. 3727–3732, May 2010.

[12] F. Chaumette and S. Hutchinson, "Visual servo control, part ii: Advanced approaches," *IEEE Robotics and Automation Magazine*, vol. 14, pp. 109–118, March 2007.

[13] A. Liegeois, "Automatic Supervisory Control of the Configuration and Behavior of Multibody Mechanisms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 7, pp. 868–871, Dec. 1977.

[14] M. Marey and F. Chaumette, "New strategies for avoiding robot joint limits: Application to visual servoing using a large projection operator," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'10*, (Taipei, Taiwan), pp. 6222–6227, October 2010.

[15] B. Nelson and P. Khosla, "Increasing the tracking region of an eye-in-hand system by singularity and joint limit avoidance," in *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*, pp. 418–423 vol.3, may 1993.

[16] E. Marchand and G. Hager, "Dynamic sensor planning in visual servoing," in *IEEE Int. Conf. on Robotics and Automation, ICRA'98*, vol. 3, (Leuven, Belgium), pp. 1988–1993, May 1998.

[17] E. 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, pp. 40–52, December 2005.

[18] R. Smits, "KDL: Kinematics and Dynamics Library." http://www.orocos.org/kdl.

[19] G. Bradski and A. Kaehler, *Learning OpenCV*. O'Reilly Media Inc., 2008.

[20] R. Munoz-Salinas, "Aruco: a minimal library for augmented reality applications based on opencv." http://www.uco.es/investiga/grupos/ava/node/26.

[21] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *In IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2149–2154, 2004.