Reaching and Grasping of Objects by Humanoid Robots through Visual Servoing

Paola Ardón[1], Mauro Dragone[2], and Mustafa Suphi Erden[2]

[1] School of Mathematical and Computer Science at Heriot-Watt University and School of Informatics at University of Edinburgh, Edinburgh, UK,
Paola.Ardon@hw.ac.uk,
[2] School of Engineering and Physical Sciences at Heriot-Watt University and Edinburgh Centre for Robotics, Edinburgh, UK

Abstract. Visual servoing allows to control the motion of a robot using information from its visual sensors to achieve manipulation tasks. In this work we design and implement a robust visual servoing framework for reaching and grasping behaviours for a humanoid service robot with limited control capabilities. Our approach successfully exploits a 5-degrees of freedom manipulator, overcoming the control limitations of the robot while avoiding singularities and stereo vision techniques. Using a single camera, we combine a marker-less model based tracker for the target object, a pattern tracking for the end-effector to deal with the robot's inaccurate kinematics, and alternate pose based visual servo technique with eye-in-hand and eye-to-hand configurations to achieve a fully functional grasping system. The overall method shows better results for grasping than conventional motion planing and simple inverse kinematics techniques for this robotic morphology, demonstrating a 48.8% of increment in the grasping success rate.

Keywords: Robotics, grasping, visual servoing, Pepper humanoid robot

1 Introduction

Grasping is considered to be simple for humans, yet it is not so simple for robots expected to operate in a dynamic environment. It requires information of the object's position, shape and environment among others. Studies demonstrate that for humans most of this information is obtained through vision [1–3]. Hence, the importance of visual control techniques that allow to handle the motion of a robotic system with the information extracted from the vision sensors.

The objective of this work is to enable visual servoing (VS)-based reaching and grasping behaviours for a service robot with limited control capabilities. We demonstrated our system using a Pepper humanoid robot from Softbank Robotics [4, 5].

There are many works that apply VS techniques on robots with morphologies that account for greater than 6-degrees-of-freedom (DOF) manipulators and apply stereo vision for target pose calculation such as [6–11]. However, our method

uses VS techniques with the purpose of reaching and grasping on a service robot with a limited control system using a single camera. [7]

The proposed approach uses Pepper's 5-DOF and a single camera located in its mouth. We combine a marker-less model based tracker (MBT) for the target object, a pattern tracking for the end-effector to deal with the inaccuracy of the robot kinematics and lack of proprioceptive sensing, and alternate pose-based-visual-servoing (PBVS) with eye-in-hand-and/eye-to-hand configurations to achieve a fully functional grasping system. The time performance proves it to be suitable for real time applications, being less than a minute to complete the reaching and grasping tasks.

We demonstrate that VS can be effectively used to enable a service robot like Pepper, which has limited range of motion and poor control features, to detect and grasp objects. We test the impact of our VS implementation and demonstrate that the grasping is substantially more successful when we use VS in contrast to the case when we use only motion planning without VS. The code of the implementation is available to download in an open repository along with the guidelines for the installation of the needed modules[3]. Additionally, given the structure of the implementation, it can easily be extended to different objects and other service robots.

This paper is organized as follows: Section 2 explains the state-of-the-art methods for VS, Section 3 shows the implemented control technique and the software used for the implementation, Section 4 details the pattern tracking algorithm to obtain the position of the robot's end-effector and the a marker-less model based tracker that gives the object's position. Finally, Section 5 analyses the results and provides suggestions for future work.

2 State of the Art

In order to achieve a successful application that controls the motion of a robot we need to combine VS and robot end-effector control techniques.

Regardless of the chosen control scheme we look to reduce the error over time, $e(t)$, between the actual and desired position of the end effector with respect to the target [3]. This error is defined as:

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

where $\boldsymbol{m(t)}$ represents the set of visual measurements that are used to compute a vector of k visual features $\boldsymbol{s(m(t), a)}$ [3]. $\boldsymbol{a}$ is the set of potential additional data. This can be the camera intrinsic parameters or the 3D model of the object to track. And $\boldsymbol{s}^*$ is the vector that stores the desired values of the features or desired final position.

[3] https://bitbucket.org/paolaArdon/master_thesis_vs_pepper

2.1 Control and Visual Servoing

VS schemes vary on how to construct s and s^* which influences on the interaction matrix L and the robot end-effector configuration. There are two main configurations for the robot end effector: (i) *eye-in-hand*, which is when the camera is attached to the moving hand, thus moves with the end-effector, and, (ii) *eye-to-hand* which is when the camera observes the target and the moving hand from a fixed position. There are mainly three types of control techniques:

- Image based VS when usually the desired position is composed of the image coordinates of different points belonging to the target.
- Pose based VS extracts the desired position from the 3D model of the object which directly depends on the camera intrinsic parameters [3].
- Hybrid VS combines the advantages of both image based visual servoing (IBVS) and PBVS techniques. However, it is highly sensitive to noise and it is computational expensive [2].

2.2 Related Work

Many approaches have been directed towards the integration and implementation of robust VS techniques for reaching and grasping behaviours. These techniques range from learning [10–13] to VS with marker-less objects using stereo vision and edge detection [14].

One of the first studies on VS [15] used a real-time tracking algorithm in conjunction with a predictive filter to allow a robotic arm to track a moving object. The work [9] proposes a method to align the end effector with the tracking target; [16] proposes new redundancy-based solutions to avoid robot joint limits of a manipulator on virtual humanoid robots, [17] applies this redundancy solution on a walking HRP2-humanoid robot. In [18] the authors present a hybrid visual servoing (HVS) control scheme for grasping that proves to be robust for real-time applications. The paper [18] shows the robustness of the system with ARMAR III arm robot; where the control scheme is based on estimating the hand position, in case of failed visual hand tracking, with the combination of visual, force and motor encoder data sensors.A similar study [19] demonstrates an application which does not rely on force sensors for the reaching and grasping but only on visual data. The paper [20] shows a combination of PBVS with a robust laser scanner that grabs features such as colour in an indoor environment. This is combined with stereo measurements that ensures the efficiency of the grasping action even if the object is unknown.

For this application we achieve a VS method that does not rely on data other than the extracted from the visual sensors and therefore do not need stereo vision for target pose calculation nor a 6-DOF manipulator. Our system can be applied on real time applications using service robots with limited morphologies for the grasping task.

3 Proposed Solution & Architecture

In this section we focus on VS for the humanoid robot Pepper for which we apply a PBVS with eye-in-hand and eye-to-hand configuration. In order to extract the visual information, we use the bottom 2D camera, located in the mouth along with the right end-effector of the robot.

The goal frame ${}^{c}\boldsymbol{M}_{h*}$ is equal to the transformation defining the position of the object with respect to the camera, ${}^{h*}\boldsymbol{M}_{c}$, multiplied by a constant transformation ${}^{o}\boldsymbol{M}_{h*}$ [19]. This constant transformation is learned by placing the hand at the desired position with respect to the object, saving the vector h^*. The transformation defining the desired pose of the hand with respect to the camera is then obtained as:

$$ {}^{c}\boldsymbol{M}_{h*} = \left({}^{h*}\boldsymbol{M}_{c}\ {}^{c}\boldsymbol{M}_{o} \right)^{-1} \tag{2} $$

In order to reduce the error of the hand, e_h, we define the transformation matrix of the current pose, h, in relation to the desired pose, h^*, as ${}^{h*}\boldsymbol{M}_{h} = {}^{o}\boldsymbol{M}_{h*}^{-1}\ {}^{c}\boldsymbol{M}_{o}^{-1}\ {}^{c}\boldsymbol{M}_{h}$. The error of the current position of the hand with respect to the target is given as:

$$ \boldsymbol{e}_h = \left({}^{h*}\boldsymbol{t}_h, {}^{h*}\theta\boldsymbol{u}_h \right), \tag{3} $$

where ${}^{h*}\boldsymbol{t}_h$ represents the translation and ${}^{h*}\theta\boldsymbol{u}_h$ the rotation control of the current position h with respect to the desired position $h*$ of the manipulator. The interaction matrix used in our approach is the one defined in [3] as:

$$ \boldsymbol{L}_{=} \begin{bmatrix} {}^{h*}\boldsymbol{R}_h & \boldsymbol{0}_3 \\ \boldsymbol{0}_3 & \boldsymbol{L}_{\theta_u} \end{bmatrix} \tag{4} $$

given that $\boldsymbol{L}_{\theta_u} = \boldsymbol{I}_3 - \frac{\theta}{2}[\boldsymbol{u}]_{\times} + \left(1 - \frac{sinc\theta}{sinc^2\frac{\theta}{2}}\right)[\boldsymbol{u}]_{\times}^2$

${}^{h*}\boldsymbol{R}_h$ represents the rotation matrix that determines the orientation of the current camera frame with respect to the desired frame. The control scheme is expressed in joint space as [3]:

$$ \dot{\boldsymbol{q}}_{\boldsymbol{h}} = -\lambda\widehat{\boldsymbol{J}_{\boldsymbol{e}}^{+}}\boldsymbol{e} + \boldsymbol{P}_{\boldsymbol{\lambda}}\boldsymbol{g}\ , \tag{5} $$

where, λ is the gain of the servo, $\widehat{\boldsymbol{J}_{\boldsymbol{e}}^{+}}$ is a combination of the interaction matrix and the articular Jacobian of the robot [21]. e is defined as $e = s - s^*$, and $\boldsymbol{J}_e$ depends on the VS task. $\boldsymbol{P}_{\boldsymbol{\lambda}}$ is the large projection operator that allows the system to perform a secondary task [21]. It is defined in [3] as: $P_\lambda = \overline{\lambda}(||e||)P_{||e||} + (2 - \overline{\lambda}(||e||))P_e$, where $P_e = (I_n - J_e^{+}J_e)$ is the classical projector and $P_{||e||}$ is the new projection operator that imposes the exponential decrease of the norm of the error instead of each term of the error vector. The sigmoid function $\overline{\lambda}(||e||)$ is used to switch from $P_{||e||}$ to P_e [21]. g is a vector that defines the secondary task [19]. In our case we use it to avoid joint limits so that the velocity controller is more reliable.

This approach produces a straight trajectory in the Cartesian space guaranteeing the exponential decay in the error measurement as seen in Fig. 4a. The implementation of the PBVS for Pepper is represented in Fig. 1 where the features and the desired position are obtained from the MBT. The hand tracker, instead of coming from the robot's odometry, comes from the tracking of the patterns. The joints control in velocity is implemented through a joint velocity control package from visual servoing platform (VISP) called *pepper_ control* [22].

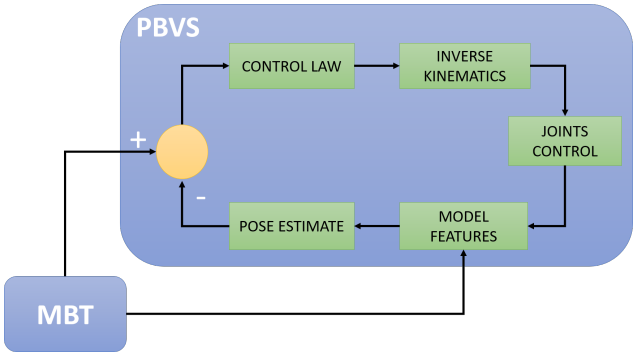


Fig. 1: Closed PBVS control scheme for Pepper humanoid robot.

3.1 System Considerations

Since Pepper's arm has only 5-DOF, in our implementation we decided to use an additional 1-DOF, the base. For the first stage, we apply a combination of PBVS with eye-in-hand configuration, where the current position is defined by the transformation matrix between the torso and the head-pitch joint, and the goal position is the one defined by the box. The second stage consists of a combination of PBVS with an eye-to-hand configuration, where the current position is the one given by the right hand and the desired position is where the hand should arrive to grasp the box. Pepper robot does not account with accurate sensor measurements, thus we added markers on the end manipulator to be tracked by vision, as explained in Section 4.

Secondly, a adaptive gain λ is used to reduce the time of convergence in order to speed up the servo. This parameter is manually set in the system. If its value is too high there is a risk of oscillations at the time of convergence, which compromises the precision of the method.

3.2 Software

We use Robotic Operating System (ROS) that facilitates libraries and tools to help the development of robot applications [23]. Some of the third party libraries used for this architecture are:

- **naoqi_driver** is a driver module between Aldebaran's NAOqiOS and ROS. It publishes all sensor and actuator data to handle the behaviour and grasping tasks [24], [25].
- **WhyCon** libraries offer a vision-based tracker system specifically for low frame rate cameras, just as Pepper's [26].
- **vision_visp** which is a ROS node that provides VISP algorithms as ROS components [27].

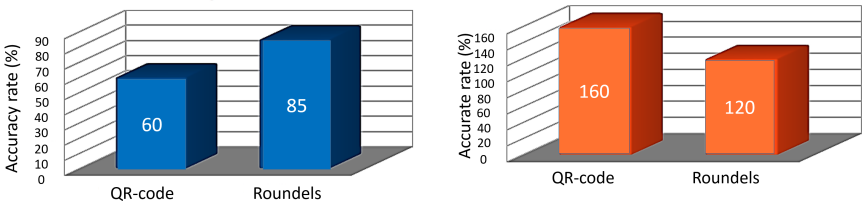
4 End Effector & Object Recognition

Due to the inaccuracy of the robot's position control and sensing this task is achieved with the help of vision by placing markers on the end effector. We tested two methods for the hand tracking system:

1. Quick response code (QR) from OpenCV libraries [28–31].
2. Roundels detection from WhyCon Libraries which is the focus of this section.

Because of the low image resolution the detection and tracking is done with the roundels detection which showed to be more robust, as seen in Fig. 2. This method is solely based on the efficient detection of black and white roundels, such as the ones shown in Fig. **??**. The roundels inner and outer diameter need to be known. The tracking is divided in detections and location of these patterns [32]. The detection combines a flood-fill segmentation algorithm with an efficient thresholding technique [32].

The position of each circle, x_c, is calculated by eigen analysis, which is represented in camera coordinate frame. We are interested on getting the orientation and position of Pepper's hand. To achieve this we use four circular patterns, following the scheme in [33], that helps us define the transformation between a global $\boldsymbol{x}$ and the camera coordinate system $\boldsymbol{x}_c$, represented as $x = T(x_c - t_0)$, where $\boldsymbol{T}$ is the similarity transformation matrix [32], and $\boldsymbol{t}_0$ represents the coordinate system origin. In our case we have $\boldsymbol{x_0}, \boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3}$ to define $\boldsymbol{t_0}$ as the centre of mass of all the patterns. We calculate the transformation between the vector $\boldsymbol{t}_0$ (camera coordinate frame) and matrix $\boldsymbol{T}$ (global coordinate system), which is the matrix of interest. We can see in Fig. 2 that both methods have



(a) Accuracy rate. (b) Detection update.

Fig. 2: Comparison of recognition performance for both methods.

an acceptable time frame performance, under 200 *ms*. However, the roundels

detection update is slightly faster than the QR codes ($120ms$ vs $160ms$). Fig. 2a shows the accuracy rate of both detection methods where the roundels method shows to be 25% higher than the QR codes, meaning that it is less likely to fail at detecting the pattern.

To track our object we use a 3D MBT from VISP that allows the tracking of a marker-less object when the computer-aided-design (CAD) model is provided. The MBT is a twofold process [34]: (a) to extract the features of the object we use the Kanadae-Lucas-Tomasi (KLT) method [35, 36]; (b) to determine the pose of the camera to match the features [34] we use a virtual VS to match the tracked features with the obtained 3D model.

5 Results & Final Discussion

In this section we present the unified results of combining both tracking methods and the PBVS eye-hand configuration to achieve a grasping application. The system flow is shown in Fig. 3.

(a) (b) (c) (d)

Fig. 3: PBVS with Pepper humanoid robot. Before starting the servo of the base, the system stores the homogeneous matrix relating the actual position of the robot with the stored desired position, *cMdBox*. a) Using eye-in-hand configuration: the blue vector shows the desired position. The error e_h is continuously calculated during the servo of the base; b) using eye-to-hand: the matrix describing the 3D position of the box with respect to the hand, *ehM*, and the homogeneous matrix relating the actual box position with the desired position, *dhM*, are calculated; c) the error e_h for the manipulator is continuously calculated until it reaches approximately 0; and d) the system decides it is safe to grasp the object and closes the end effector. The calculations are done using the equations of Section 3.

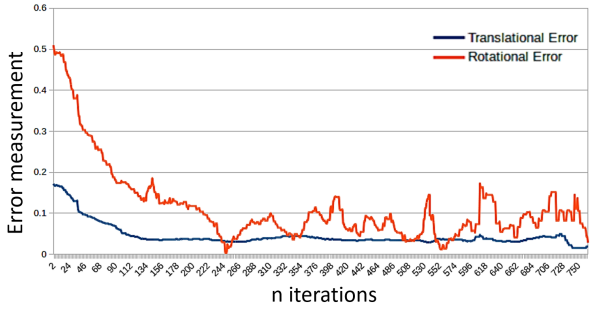
The experiments are done on Linux Trusty system using a 5GHz network to connect to the Naoqi 2.4 version on Pepper robot. We measure the number of successful attempts. Specifically, for our purposes we consider a successful grasp if the box is not dropped by the end-effector during the reaching process and can be lifted-up. We consider a failed attempt whenever the box was dropped in the process or if it was approached correctly but the robot did not lift it up.

Fig. 4a shows the exponential decrement of the error measurements in *cm* and *rad*. Where:

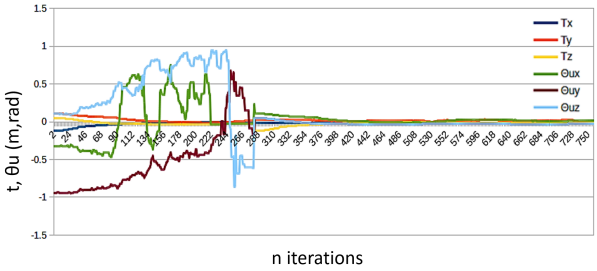
- $\lambda_0 = \lambda(0)$ is the gain in 0, which is for very small values of $||e||$,

– $\lambda_\infty = \lambda_{||e||\to\infty}\lambda(||e||)$ is the gain to infinity, which is for high values of $||e||$,
– And, λ_0' is the slope of λ at $||e|| = 0$

We obtain an average error of 1 *mm* and 5 *deg*. In this case, the task error arrives to convergence in 17 *seconds*. A detailed image of the decay on the tasks error for the arm VS can be seen on Fig. 4b.



(a) Translational and rotational MSE.



(b) Task error for the translation and rotation at grasping.

Fig. 4: Error measurements for: $\lambda_0 = 0.7$, $\lambda_\infty = 0.08$ and $\lambda_0' = 3$. Convergence for the error = 17 sec.

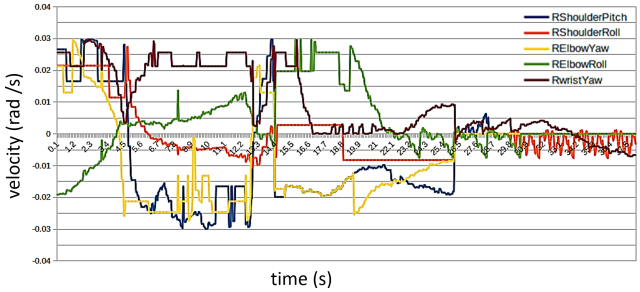
We want to know the right value of the parameters so that our approach is as efficient as possible. Given our implementation we care about tuning the adaptive gain values. Fig. 5 shows the output of the applied velocities when we vary λ_∞. Table. 1 shows a summary of the results from varying the λ_∞ parameter. Where we can compare the threshold values chosen for the translational and rotational errors , $||e_{u_t}||$ and $||e_{u_\theta}||$ respectively and their convergence time.

Table 1: Summary for tuning parameters

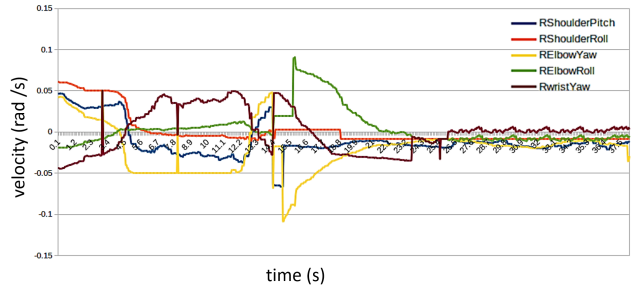
λ_∞	$\|\|e_{u_t}\|\|$	$\|\|e_{u_\theta}\|\|$	Conv. Time
0.02	1 mm	1 deg	30 sec
0.07	4 mm	3 deg	25 sec
0.1	6 mm	8 deg	12 sec

From Table. 1 we see that the higher the gain the faster the system arrives to convergence, which is the case for $\lambda_\infty = 0.1$ where the convergence time is only

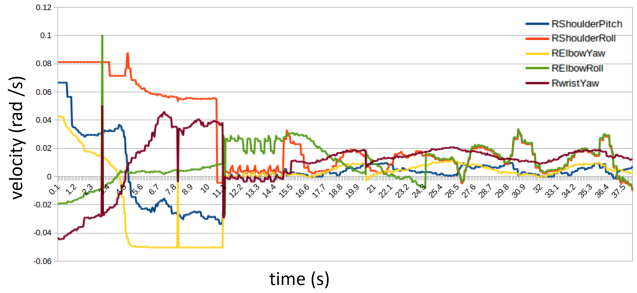
12 seconds. On the down side for this case, the imprecision is higher. Table. 1 is a summary of the error threshold used in the different cases of varying the gain.



(a) $\lambda_0 = 0.06$, $\lambda_\infty = 0.02$ and $\lambda'_0 = 3$. Convergence: 30 sec.



(b) $\lambda_0 = 0.06$, $\lambda_\infty = 0.07$ and $\lambda'_0 = 3$. Convergence: 25 sec.



(c) $\lambda_0 = 0.06$, $\lambda_\infty = 0.1$ and $\lambda'_0 = 3$. Convergence: 12 sec.

Fig. 5: Varying λ_∞ on Pepper right hand joints.

In all the cases, changing our adaptive gain the system arrives to the goal position travelling a relative straight line in the Cartesian space.

5.1 Discussion

We see that the combination of the hybrid marker-less MBT for the box, the roundels tracker for Pepper's hand along with an adaptive gain for the control scheme allows for reasonable execution times, as observed on Table. 1 where the

longest convergence time is 30 seconds. Given the modularity of the implementation it can easily be adapted to other humanoid robots and objects.

Fig. 5 shows the velocities discontinuities and oscillations. From the varying λ_∞ we observe that the applied velocities are high when the error exponential decrement is high and small when the error decrement is small. The oscillations are directly related to the λ_0 or λ_∞ gain value.

The greater the λ_∞ value the faster the system arrives to a convergence time, as observed in table 1. However, the threshold error also needs to be greater so that the robot does not drop the box during the reaching process. The system achieves a precision of 1 *mm* and 1 *deg* when λ_∞ is really small (0.02), however it takes around half a minute to grasp the box. For our purposes we care more about precision than computational time. Therefore we use a combination of $\lambda_\infty = 0.07$ but a $\lambda_0 = 0.02$ in the final implementation. As a result, we achieve the position quite fast but reducing the oscillations when the error is small so that we have a higher precision when grasping. For this task we take 28 *seconds* to complete the arm PBVS task. Out of 25 consecutive trials Pepper successfully grasped the object 17 times. Obtaining a sensitivity rate of 80%. Which means that we rarely miss the target object.

During the different tests it was noticed that the attempts where Pepper miss-predicted the goal position were the ones where the hybrid MBT fails. Either it gets lost because of illumination, miss-calculates the features and/or is not correctly initialized by the user.

5.2 Comparing Methods

As a summary, Table. 2 contrasts the reaching task by using *MoveIt!* [37] with our PBVS technique. *MoveIt!* is a state-of-the-art software for manipulation available with ROS which can be connected with Aldebaran [38] software.

Table 2: Comparing *MoveIt!* and PBVS.

	MoveIt	PBVS
Time (sec)	**2.6**	28
Success Rate	23.20%	**72%**
Sensitivity Rate	1	**0.8**

As observed in Table. 2, the final PBVS outperforms *MoveIt!* software in success grasping rate. We demonstrated that VS can be effectively used to enable a service robot like Pepper, which has limited range of motion and low power actuation, to detect and grasp objects. We test the impact of VS in this scheme and demonstrate that the grasping is substantially more successful when we use VS compared to the case when we use only motion planning.

6 Conclusions and Future Work

This work presents a grasping application for service robots with limited range of motion and poor control features, to detect and grasp objects using a single

camera for the visual feedback. We consider our application to have potential, giving room to some extensions. Some of them being:

- The biggest setback in terms of processing is the tracking of the target features through the network. Therefore a good improvement would be to integrate this tracker into the robot's central processing unit (CPU) instead of having it running on the external computer.
- Both hands could be integrated into the servo alternating their usage.
- Remove the visual markers on the hand and integrate a MBT to track its position instead.
- To initialize the target it needs to be inside the camera frame. A nice extension would be to add a tracking learning detection (TLD) tracker so that it looks for the box in the room before initializing the MBT.

References

1. Schack, T., Ritter, H.: The cognitive nature of action functional links between cognitive psychology, movement science, and robotics. Progress in Brain Research **174**, 231–250 (2009)
2. Espiau, B., Chaumette, F., Rives, P.: A new approach to visual servoing in robotics. pp. 313–326. IEEE (1992)
3. Siciliano, B., Khatib, O.: Springer handbook of robotics. Springer Science & Business Media (2008)
4. Aldebaran cartesian control. http://www.bx.psu.edu/~thanh/naoqi/naoqi/motion/control-cartesian.html. Accessed: 2017-02-03
5. Aldebaran aldebaran – pepper robot specifications. http://doc.aldebaran.com/2-0/family/juliette_technical/. Accessed: 2017-05-05
6. Lippiello, V., Ruggiero, F., Siciliano, B., Villani, L.: Preshaped visual grasp of unknown objects with a multi-fingered hand. In: Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on, pp. 5894–5899. IEEE (2010)
7. Corke, P., Good, M.: Controller design for high-performance visual servoing. IFAC Proceedings Volumes **26**(2), 629–632 (1993)
8. Rizzi, A.A., Koditschek, D.E.: An active visual estimator for dexterous manipulation. IEEE Transactions on Robotics and Automation **12**(5), 697–713 (1996)
9. Horaud, R., Dornaika, F., Espiau, B.: Visually guided object grasping. ieee Transactions on Robotics and Automation **14**(4), 525–532 (1998)
10. Kraft, D., Detry, R., Pugeault, N., Baseski, E., Piater, J.H., Kruger, N.: Learning objects and grasp affordances through autonomous exploration. In: ICVS (2009)
11. Macura, Z., Cangelosi, A., Ellis, R., Bugmann, D., Fischer, M.H., Myachykov, A.: A cognitive robotic model of grasping (2009)
12. Levine, S., Pastor, P., Krizhevsky, A., Ibarz, J., Quillen, D.: Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. The International Journal of Robotics Research p. 0278364917710318 (2016)
13. Morales, A., Chinellato, E., Fagg, A.H., Pobil, A.P.D.: An active learning approach for assessing robot grasp reliability. In: 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566) (2004)

Thanks to Giovanni Claudio for his help on the use of ViSP and bridging Pepper robot with ROS.

14. Vicente, P., Jamone, L., Bernardino, A.: Towards markerless visual servoing of grasping tasks for humanoid robots. In: Robotics and Automation (ICRA), 2017 IEEE International Conference on, pp. 3811–3816. IEEE (2017)
15. Allen, P.K., Yoshimi, B., Timcenko, A.: Real-time visual servoing. In: Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on, pp. 851–856. IEEE (1991)
16. Chaumette, F., Marchand, É.: A redundancy-based iterative approach for avoiding joint limits: Application to visual servoing. IEEE Transactions on Robotics and Automation **17**(5), 719–730 (2001)
17. Mansard, N., Stasse, O., Chaumette, F., Yokoi, K.: Visually-guided grasping while walking on a humanoid robot. In: Robotics and Automation, 2007 IEEE International Conference on, pp. 3041–3047. IEEE (2007)
18. Vahrenkamp, N., Wieland, S., Azad, P., Gonzalez-Aguirre, D.I., Asfour, T., Dillmann, R.: Visual servoing for humanoid grasping and manipulation tasks. In: Humanoids (2008)
19. Claudio, G., Spindler, F., Chaumette, F.: Vision-based manipulation with the humanoid robot romeo. In: Humanoids (2016)
20. Taylor, G., Kleeman, L.: Grasping unknown objects with a humanoid robot (2002)
21. Marey, M., Chaumette, F.: A new large projection operator for the redundancy framework. 2010 IEEE International Conference on Robotics and Automation pp. 3727–3732 (2010)
22. Inria peppercontrol. https://github.com/lagadic/pepper_control. Accessed: 2017-02-03
23. ROS ros.org. http://wiki.ros.org/. Accessed: 2017-02-02
24. ROS naoqi driver. http://wiki.ros.org/naoqi_driver. Accessed: 2017-02-03
25. Inria visp naoqi bridge. http://jokla.me/software/visp_naoqi/. Accessed: 2017-02-01
26. Irse whycon. https://github.com/lrse/whycon. Accessed: 2017-02-02
27. ROS vision visp. https://github.com/lagadic/vision_visp. Accessed: 2017-02-03
28. QRCode optical flow. http://docs.opencv.org/3.2.0/d7/d8b/tutorial_py_lucas_kanade.html. Accessed: 2017-02-03
29. OpenCV opencv team. http://opencv.org/. Accessed: 2017-02-02
30. Kato, Y., Deguchi, D., Takahashi, T., Ide, I., Murase, H.: Low resolution qr-code recognition by applying super-resolution using the property of qr-codes. In: ICDAR (2011)
31. Belussi, L., Hirata, N.S.T.: Fast qr code detection in arbitrarily acquired images. In: SIBGRAPI (2011)
32. Nitsche, M., Krajnik, T., vCizek, P., Mejail, M., Duckett, T.: Whycon: An efficent, marker-based localization system (2015)
33. INRIA whycon tracking. https://github.com/lagadic/pepper_hand_pose. Accessed: 2017-02-03
34. Comport, A.I., Marchand, É., Pressigout, M., Chaumette, F.: Real-time markerless tracking for augmented reality: the virtual visual servoing framework. IEEE Trans. Vis. Comput. Graph. **12**, 615–628 (2006)
35. INRIA visp edge tracking. http://visp-doc.inria.fr/manual/visp-2.6.0-tracking-overview. Accessed: 2017-02-03
36. Inria visp. https://visp.inria.fr/. Accessed: 2017-02-03
37. ROS moveit simpple grasps. https://github.com/davetcoleman/moveit_simple_grasps/. Accessed: 2017-28-04
38. Aldebaran movement detection. http://doc.aldebaran.com/2-4/naoqi/vision/almovementdetection.html#almovementdetection. Accessed: 2017-04-13