Integrating 3D Features and Virtual Visual Servoing for Hand-Eye and Humanoid Robot Pose Estimation

Xavi Gratal, Christian Smith, Mårten Björkman and Danica Kragic

***Abstract*— In this paper, we propose an approach for vision-based pose estimation of a robot hand or full-body pose. The method is based on *virtual visual servoing* using a CAD model of the robot and it combines 2-D image features with depth features. The method can be applied to estimate either the pose of a robot hand or pose of the whole body given that its joint configuration is known. We present experimental results that show the performance of the approach as demonstrated on both a mobile humanoid robot and a stationary manipulator.**

I. Introduction

Most of the object grasping and manipulation tasks require the pose between the robot hand and the object to be known prior to or during execution of the grasp. Although power grasping may not need a precise pose of the robot hand relative to the object, precision grasps and in-hand manipulation require a high level of accuracy [1]. In many cases, the exact model of the robot arm may not be available and forward kinematics is not accurate enough to guide grasping [2]. Vision-based hand pose estimation can alleviate this problem and enable control without an extra step requiring position or image-based visual servoing.

Similar requirements arise when grasping and manipulation tasks are performed by several robots where the relative position of the robots with respect to each other must be known [3]. In this case, one robot can obtain its relative position with respect to another robot by identifying the full pose of the robot body or solely the pose of its hand.

In this paper, we propose an approach for vision-based pose estimation of a robot hand or full-body pose. The method is based on Virtual Visual Servoing that uses RGB-D images together with a CAD model of the robot, to continuously track the pose of a robot with respect to the camera, or between different parts of a robot. The main contributions of this work are:

- The integration of 2-D and 3-D information into the Virtual Visual Servoing framework. Our method, given an approximate initial pose estimate, refines it iteratively to obtain a more precise estimate. We show that the use of 3-D information improves the estimate in comparison to using only 2-D images.
- A method for pose tracking of a robot in joint space given that its configuration is known. This adds the challenge of having to track each of the links of the robot, which places special requirements on rendering for virtual visual servoing. As we will demonstrate, our system allows us to treat each joint in the same way that we treat each of the components of the motion of the robot, thus making it suitable for complex models.

The authors are with the Computer Vision and Active Perception Lab., Centre for Autonomous Systems, School of Computer Science and Communication, Royal Institute of Technology KTH, SE-100 44 Stockholm, Sweden. This work has been supported by EU FP7 grant 288533-RoboHow.cog, Swedish Research Council and Swedish Foundation for Strategic Research. e-mail: {javiergm|ccs|celle|dani}@kth.se

This paper is organized as follows: in Section II we review related work. The proposed methodology is presented in Section III and the results of the experimental evaluation are presented in Section IV. We conclude the paper in Section V.

II. Related Work

In general, it is possible to use any tracking method to retrieve the pose of the manipulator. The existing methods can be divided in two groups: appearance-based (also referred to as global) [6] and feature-based (also referred to as local). These methods differ mostly from each other in the kind of features that are used, the matching algorithm and the optimization method. Appearance-based methods have commonly been used for obtaining the pose of a moving camera [7], [8] or for coarse pose estimation of objects that occupy a substantial portion of the image or are easily segmented [9]–[12]. There are also approaches that rely on the use of fiducial markers [4], [5] that may limit the mobility of the manipulator, due to the requierment of markers being continuously in the visual field of the camera.

The features commonly employed for tracking are corners or edges. These are extracted using some interest point detector [13], [14] and then encoded into a local descriptor [15] to ease the matching of the features with the ones stored in the model. These kinds of features usually work better with textured objects, and can be problematic with robotic manipulators, which usually consist of flat, shiny surfaces which change with illumination. For the optimization part of the method, if the points are correctly matched and detectable in the views with arbitrary precision, three points are enough to solve the problem [16]. In general, more points are needed, and methods exist that are robust in the presence of noise due to incorrect matches or inaccuracy in point detection [17]–[19]. Most of the systems based on these methods use features extracted from 2-D images as input, and are thus highly sensitive to viewpoint changes. Our method, by using a full 3-D CAD model of the tracked object, is more tolerant to viewpoint changes. Some methods, such as [20], also use a CAD model for tracking the object, but can only support simple models, with a few hundreds of polygons, and lack direct support for tracking a complete kinematic chain.

Virtual Visual Servoing (VVS) [21] is an iterative optimization method where given a real image of the object for

which we want to track the pose and an initial estimation of the pose, a model is projected into the image at the estimated pose. Then, features are extracted both in the real and model image, and the difference between the position of the features is used to improve the pose estimation. In our previous work [22], the original method is extended to make use of a full 3D CAD model of the object. The real image contains only color data (no depth information), so the features chosen are the edges detected in the image.

III. Methodology

As stated, our approach is based on VVS where a rendered model of the object is aligned with the object as seen in the current camera image. We now provide the notation, followed by an overview of the method and a detailed description of each component.

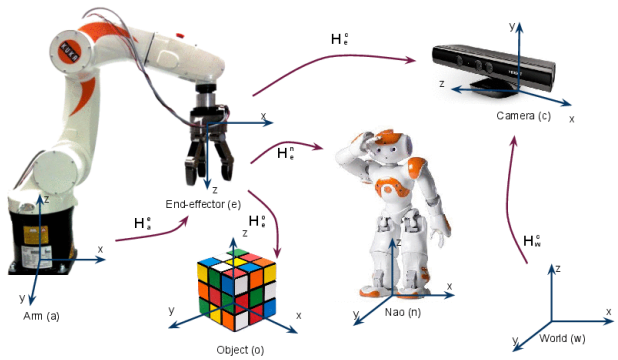

Fig. 1. Coordinate systems and some transformations between them.

The problem we deal with is the estimation of the homogeneous transformation between two coordinate frames, $\boldsymbol{H_o^e}$, relating the corresponding coordinates of a point $\boldsymbol{q_e}$ and $\boldsymbol{q_o}$ through $\boldsymbol{q_e} = \boldsymbol{H_o^e q_o}$, see Fig. 1. Vision-based pose estimation provides us with a means of obtaining the transformation $\boldsymbol{H_x^c}$ between a coordinate frame x and the coordinate frame c of the camera. By obtaining these for e and o, we can obtain the transformation $\boldsymbol{H_o^e = H_e^{c-1} H_o^c}$ between the two original frames. The robot arm is assumed to consist of several links. In general, the relationships between links form a kinematic tree, where the transformation $\boldsymbol{H_{l_i}^a}$ between the root link and link i is $\boldsymbol{H_{l_i}^a = M_{j_i(1)} M_{j_i(2)} \dots M_{j_i(n)}}$ where $\boldsymbol{M_k}$ is the transformation corresponding to joint k and j_i is the sequence of indices of the joints that separate the root link and link i in the kinematic tree. The transformation between the camera and each of the links is then: $\boldsymbol{H_{l_i}^c = H_a^c M_{j_i(1)} M_{j_i(2)} \dots M_{j_i(n)}}$ which is what we wish to estimate. $\boldsymbol{H_a^c}$ is the rigid transformation between the camera and the root of our kinematic tree, composed of a rotation $\boldsymbol{R_a^c}$ and a translation $\boldsymbol{t_a^c}$. If the kinematics and joint configuration of our robot are fully known, $\boldsymbol{M_k}$ will be known, so we only need to estimate the rotation and translation of the base. When the kinematics is known but the joint configuration is not, determining the joint transformation will be equivalent to determining some parameter ϕ_k for the joint (usually an angle). We can then write $\boldsymbol{H_{l_i}^c = H_a^c(R_a^c, t_a^c)} \prod_{s=1}^{n} \boldsymbol{M_{j_i(s)}}(\phi_{j_i(s)})$ where $\boldsymbol{R_a^c}$, $\boldsymbol{t_a^c}$ and $\boldsymbol{\phi} = \{\phi_1, \phi_2, \dots \phi_m\}$ are the parameters to estimate.

A. System overview

The outline of the system is shown in Fig. 2. To achieve the alignment, we can formalize it by either controlling the pose of the *virtual* object or moving the *virtual* camera so that the image perceived by the camera corresponds to the current camera image, denoted as *real* camera image. In this paper, we adopt the first approach achieved through rendering synthetic images by incrementally changing virtual the pose of the robot hand/arm. A rough initial estimate of the pose is given by forward kinematics. Image features are then extracted from the rendered image and matched to the features of the current image of the hand. We define

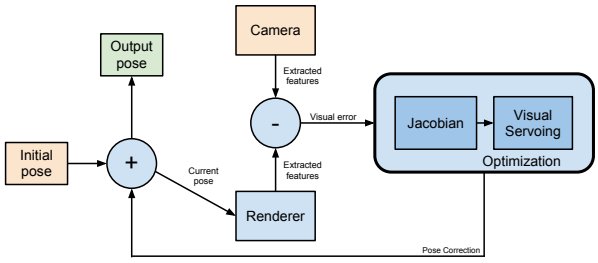

Fig. 2. Outline of the proposed model-based tracking system.

an error vector based on the differences between the image features in the rendered and current real camera image. The error vector is used for the pose estimation process using VVS formalization. Images $I_r(u, v)$ are captured with the Kinect sensor at 30fps. The sensor provides also a depth map $D_r(u, v)$.

B. Synthetic image generation

For rendering, we assume that a CAD model of the robot is available. We developed a new scenegraph engine focusing on rendering offline images and associated maps at a very high speed, using custom shaders with OpenGL. Since our matching is based solely on the edge data, the model is rendered without any texture or lighting, which allows us to obtain more than 1000 fps in modern consumer GPU hardware, for a model containing more than 100000 polygons. For each real image, several iterations of VVS need to be applied before convergence, thus fast rendering is necessary for real-time performance. Typically, 10 to 30 iterations are needed, depending on the initial offset in the pose thus requiring rendering at about 1000 fps when real images are being captured at 30 fps. Our CAD model is broken into N_{link} meshes, one for each link of the manipulator that can move independently. Each point $\boldsymbol{p_{l_i}} = [x_{l_i}, y_{l_i}, z_{l_i}, 1]^T$ in mesh i can be transformed into the camera coordinate system

using

$$\boldsymbol{p_c} = \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = \boldsymbol{H_w^c H_a^w}(\hat{\boldsymbol{R}}_{\boldsymbol{a}}^{\boldsymbol{w}}, \hat{\boldsymbol{t}}_{\boldsymbol{a}}^{\boldsymbol{w}}) \boldsymbol{H_{l_i}^a}(\hat{\phi}) \boldsymbol{p_{l_i}} \quad (1)$$

where we make it explicit that this transformation depends on the current estimation of the position and rotation of the manipulator and the joint configuration. To project the resulting point into the image plane, we use the projection matrix $\boldsymbol{P}$, which must correspond with the projection matrix for the camera model of the real camera. The point (u, v) in the image can then be obtained as:

$$\boldsymbol{p_p} = \begin{bmatrix} x_p & y_p & z_p & w_p \end{bmatrix}^T = \boldsymbol{P p_c}, \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} x_p/w_p \\ y_p/w_p \end{bmatrix} \quad (2)$$

Using this transformation we generate three different maps. $I_s(u, v)$ contains 1 where a point was rendered and 0 for the background. $D_s(u, v)$ contains the depth of the rendered point. It is also used during the rendering process for occlusion culling. Finally, $T_s(u, v)$ contains the index of the link for the corresponding pixel.

C. Image features

As mentioned before, the rendered and real images are compared using image features. In our previous work [22], we used only edges in the 2D image as a feature. Here, image edges are still used, but they are combined with new features to increase the robustness and accuracy of the system.

1) Image edges: We use a Canny operator for edge detection. We extract edges in both the real and virtual images, and the vector between an edge point in one image and the closest edge point in the other image gives us a directed error vector. For efficiency, this is implemented using the distance transform method: for each real image, a map is created which assigns to each point in the map, the position of the closest edge point. Then, for each edge point in the virtual image, the error vector can be obtained by a simple lookup in that map. Since our method assumes an initial pose estimate, edges should only be matched when their orientations are similar. To enforce that, 8 maps are generated, which record the closest edge within a certain range of orientations. Then, for each edge point in the virtual image, the lookup is performed only in the map which best corresponds to the orientation of the edge point.

2) Image depth: One of the important parts of the system is the choice of appropriate features for the raw depth information. SIFT-like 3-D features, such as the one introduced in [24] are a possibility, but they suffer from the same drawbacks as SIFT for the 2-dimensional case. Robotic surfaces are often flat, and the matching of features is an expensive operation that would need to be performed for every frame. It is also possible to use the depth information directly as a feature. In [25], the depth map is assumed to be smooth, and the difference between the depths of each point in the source and target images is used as a feature. The main drawback of this approach is that it leads to incorrect values in the edges of the object, and is extremely sensitive to small occlusions, such as the ones that can be caused by cables in robotic environments. Also, we do not have depth information for the whole scene, but only for the manipulator, which will usually only cover a small part of the depth map.

The approach we apply is to use the distance from one 3-D point obtained from the virtual image to the closest point obtained in the real image. The 3-D image is actually an edge image, in the sense that each point corresponds to what would be an edge in fronto-parallel 2-D cuts of the scene, so this method has similar advantages to the one we adopted for the 2-D information. We implement it again using a 3-D version of the distance transform, where we create, for the real image, a 3-D map of the distance from each point in space to the nearest extracted point. Then, for each depth point in the virtual image, we just need to perform a lookup for the nearest point in the 3-D map, and we obtain a directed error vector. Another practical advantage of using this method is that it is very similar to the one used for 2-D edges, so it can be easily integrated into our framework.

3) SURF features: The previous features meet the key requirements of speed and work well with textureless objects, but their main drawback is that since each point in one image is compared to the closest point in the other image, the performance degrades when the initial pose is bad. To improve the performance in such cases, we need features that can be robustly matched between the images and we chose to use SURF [26]. Since our CAD models are not textured, we cannot directly detect SURF features in the rendered image. We could generate texture maps for our CAD models, but even then, detecting SURF features for every generated virtual image would be prohibitively expensive. Instead, we enrich our CAD model with pre-detected SURF features. In an offline process, we detect SURF features in different parts of our model, and for each feature we record its 3D position within the CAD model, together with information about the viewpoint, the detection size and the feature descriptor. Then, during the pose estimation loop, SURF features are detected in each captured image, and their descriptors matched against the database of stored features. Matches that are not consistent in terms of viewpoint and detection size are discarded. The distance between the feature as detected in the real image and the projection of the recorded position into the rendered image is then used as the feature to minimize.

D. Visual Servoing

The basic idea behind visual servoing is to create an error vector which is the difference between the desired and measured values for a series of features, and then map this error directly to robot motion. Let $\boldsymbol{s}(t)$ be a vector of feature values which are measured in the image. In our case, it is constructed, at each iteration, with the distances $\boldsymbol{d}$ between the detected points in the real and synthetic images as $\boldsymbol{s}(t) = \begin{bmatrix} d_1, d_2, \ldots, d_n \end{bmatrix}^T$. Then $\dot{\boldsymbol{s}}(t)$ will be the rate of change of these distances with time as $\boldsymbol{H_a^c}(\boldsymbol{R_a^c}, \boldsymbol{t_a^c})$ is updated to improve the fit between real and synthetic images. The change in this transformation can be described

TABLE I

ESTIMATION ERRORS IN THE RETRIEVED POSE FOR KUKA ARM AND NAO ROBOT.

	KUKA arm				**NAO**			
	simulation		real data		simulation		real data	
	2-D features	2-D and 3-D features	2-D features	2-D and 3-D features	2-D features	2-D and 3-D features	2-D features	2-D and 3-D features
Translation error parallel to image plane	11.3 mm	9.8 mm	15.8 mm	12.1 mm	10.2 mm	9.7 mm	17.1 mm	11.7 mm
Translation error perpendicular to image plane	40.1 mm	9.2 mm	46.3 mm	9.7 mm	30.7 mm	9.9 mm	39.3 mm	9.6 mm
Rotation error	1.01 °	0.63 °	1.43 °	0.93 °	1.23 °	0.79 °	1.17 °	1.08 °

by a translational velocity $\boldsymbol{T}(t) = [T_x(t), T_y(t), T_z(t)]^T$ and a rotational velocity $\boldsymbol{\Omega}(t) = [\omega_x(t), \omega_y(t), \omega_z(t)]^T$, which form a velocity screw: $\dot{\boldsymbol{r}}(t) = \left[T_x, T_y, T_z, \omega_x, \omega_y, \omega_z\right]^T$. We can then define the image jacobian or interaction at a certain instant as $\boldsymbol{J}$ so that $\dot{\boldsymbol{s}} = \boldsymbol{J}\dot{\boldsymbol{r}}$ where

$$\boldsymbol{J} = \left[\frac{\partial \boldsymbol{s}}{\partial \boldsymbol{r}}\right] = \begin{bmatrix} \frac{\partial d_1}{\partial T_x} & \frac{\partial d_1}{\partial T_y} & \frac{\partial d_1}{\partial T_z} & \frac{\partial d_1}{\partial \omega_x} & \frac{\partial d_1}{\partial \omega_y} & \frac{\partial d_1}{\partial \omega_z} \\ \frac{\partial d_2}{\partial T_x} & \frac{\partial d_2}{\partial T_y} & \frac{\partial d_2}{\partial T_z} & \frac{\partial d_2}{\partial \omega_x} & \frac{\partial d_2}{\partial \omega_y} & \frac{\partial d_2}{\partial \omega_z} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial d_n}{\partial T_x} & \frac{\partial d_n}{\partial T_y} & \frac{\partial d_n}{\partial T_z} & \frac{\partial d_n}{\partial \omega_x} & \frac{\partial d_n}{\partial \omega_y} & \frac{\partial d_n}{\partial \omega_z} \end{bmatrix} \quad (3)$$

which relates the motion of the (virtual) manipulator to the variation in the features. The method used to calculate the jacobian is described in detail below.

However, what we need to be able to correct our pose estimation is the opposite, that is, we need to compute $\dot{\boldsymbol{r}}(t)$ given $\dot{\boldsymbol{s}}(t)$. When $\boldsymbol{J}$ is square and nonsingular, it is invertible, and then $\dot{\boldsymbol{r}} = \boldsymbol{J}^{-1}\dot{\boldsymbol{s}}$. This is not generally the case, so we have to compute a least squares solution, which is given by $\dot{\boldsymbol{r}} = \boldsymbol{J}^{+}\dot{\boldsymbol{s}}$ where $\boldsymbol{J}^{+}$ is the pseudoinverse of $\boldsymbol{J}$ calculated as $\boldsymbol{J}^{+} = (\boldsymbol{J}^T\boldsymbol{J})^{-1}\boldsymbol{J}^T$. The goal for our task is for all the edges in our synthetic image to match edges in the real image, so the target value for each feature is 0, and we can define the error function as $\boldsymbol{e}(\boldsymbol{s}) = \dot{\boldsymbol{s}} - \boldsymbol{0}$ which leads us to the simple proportional control law $\dot{\boldsymbol{r}} = -K\boldsymbol{J}^{+}\dot{\boldsymbol{s}}$ where K is the gain parameter.

E. Estimation of the jacobian

To estimate the jacobian we need to calculate the partial derivatives of the feature values d_i with respect to each of the components of the motion we are estimating ($\boldsymbol{R_a^c}$, $\boldsymbol{t_a^c}$ and ϕ). When features are the position of points or lines, it is possible to find analytical solutions for the derivatives. Here, however, the features are the distances from the edges of the synthetic image to the closest edge in the real image, so we approximate the derivative by calculating how a change in the motion component affects the value of the feature.

Each of the feature values d_i is the distance between a point $\boldsymbol{p_i^s(u,v)}$ in the synthetic image and the corresponding point $\boldsymbol{p_i^r(u,v)}$ in the real image. We want to find the point $\boldsymbol{p_i^{s\prime}(u,v)}$ which results from applying the small change in the motion component to $\boldsymbol{p_i^s(u,v)}$. We can use the depth map $D_s(u,v)$ to find the corresponding 3D point in camera coordinates, and then use the inverse of the matrix that we used to render the point from the model to find the point $\boldsymbol{p_i^m(x,y,z)}$ in the coordinate system of the model. Different points in the image will correspond to different links in the robot, but we can obtain the link for each point, and thus its corresponding projection matrix from map $T_s(u,v)$.

Once we have $\boldsymbol{p_i^m(x,y,z)}$, we can reproject it using the new transformation matrix which would result from applying the small change in motion component, obtaining, as we wanted, $\boldsymbol{p_i^{s\prime}(u,v)}$. We then compute the new distance d_i' to the corresponding point in the real image, and we can estimate the derivative as $(d_i' - d_i)/\epsilon$, where ϵ is the change in motion component.

IV. EXPERIMENTAL EVALUATION

We test the performance of the method with respect to the choice of 3D features. We then give a more extensive evaluation of the method's performance in different situations, demonstrating hand-eye calibration or robot pose estimation.

A. Accuracy evaluation and comparison to previous method

We first evaluated the accuracy in the pose estimation in tracking two robots: A KUKA industrial arm and a NAO humanoid robot. We performed tests both with imagery from a simulator and with real-world data obtained from a Kinect camera. The results, which include a comparison with our previous method which used only 2-D information are summarized in Table III-D. Each value is the average error over 1000 runs. We used 5 different joint configurations for each robot and 10 different initial estimates for the pose, giving the total of 50 starting conditions. Examples of initial and final position for a run are shown in Figures 3 and 4.

To evaluate the error in the real-world experiments, we needed ground truth. We chose to compare the results to how a human would manually align the input point cloud with the rendered CAD model, using the same information available to the robot. For the position error, we distinguish between errors that are parallel or perpendicular to the image plane, and we observe that the errors in the estimation of the depth of the object are greatly reduced.

B. Convergence of the method

To evaluate the robustness of the method, we estimate the maximum error in the initial pose estimation for which the method will still converge, using the KUKA industrial arm and real-world imagery. In this set of experiments,

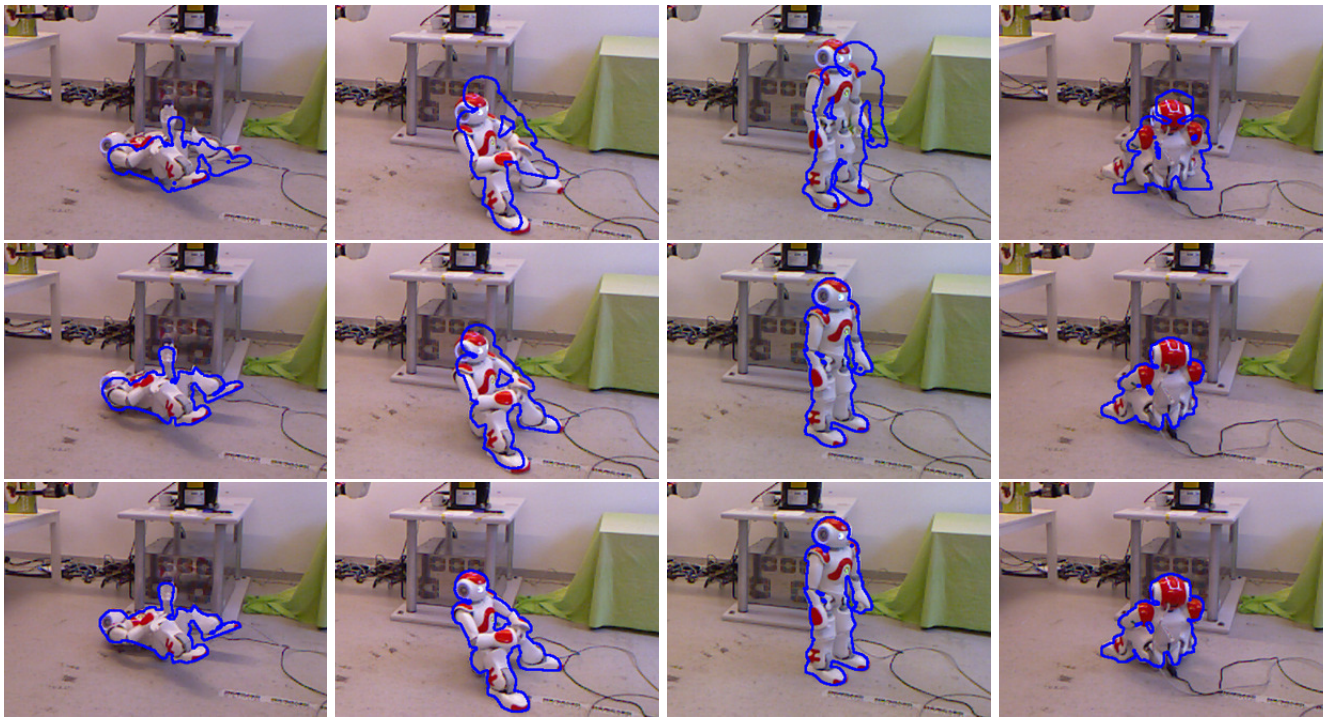

Fig. 3. A few examples of the initial (upper row) and final poses (lower row) for a Nao robot in several different configurations. The blue outline represents the current estimation of the pose. Best viewed in color.

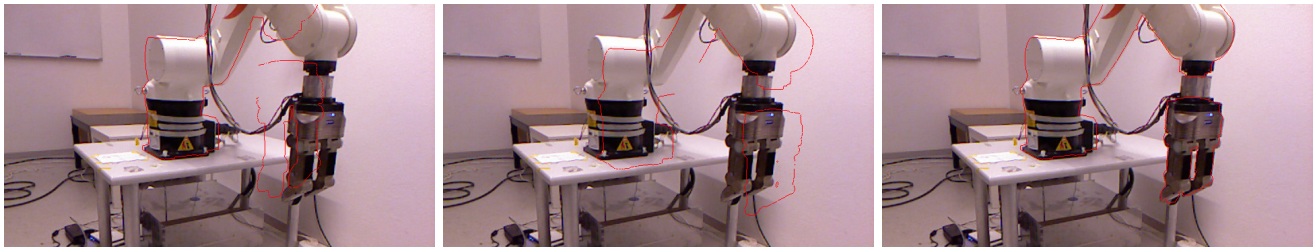

Fig. 4. Initial poses with (a) errors in the joint positions (b) errors in the transformation for the whole manipulator. (c) Converged result. Red outline represents the current estimation. Best viewed in color.

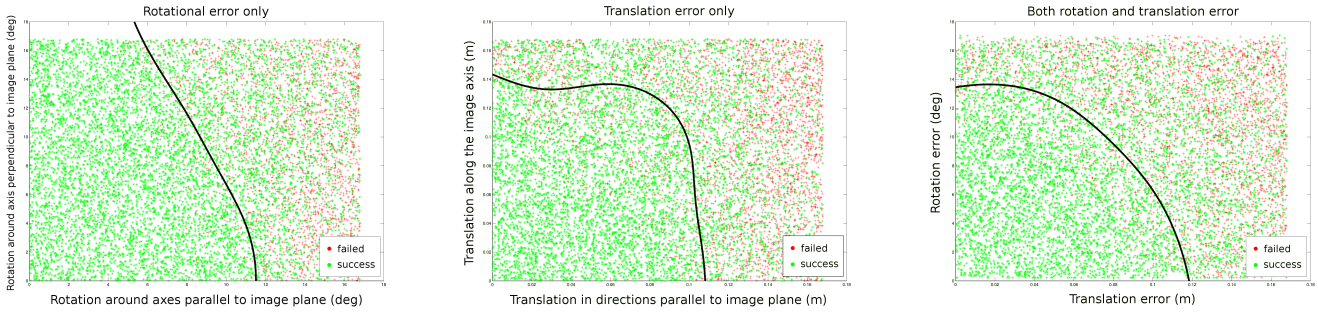

Fig. 5. Convergence results for (a) only rotational error (b) only translational error (c) both rotational and translational error. Best viewed in color.

we also assume that the joint configuration is known. We performed a total of 30000 runs of the method, using five different joint configurations for the manipulator. The results for different kinds of errors, including a decision boundary for convergence can be seen in Figure 5.

We can observe that for translational errors of less than 10 cm and rotational errors of less than 10 degrees, the method converges with high probability. Also, we can see that translational errors along the axis perpendicular to the image plane and rotational errors around that same axis, a

larger error is tolerated.

C. Estimation of joint configuration

Until now, we have assumed a known joint configuration. While this is the case in our system, it is not true for many robotic manipulators. In the following set of experiments, we assume that the transformation with respect to the base of the manipulator is known, but there is some error in the initial estimate of the joint configuration. Having the real values as provided by our system allows us to compare the results of our method with the true values.

We ran our method 10000 times for the KUKA arm using real-world images, with 5 different target (real) joint configurations, and each time introducing an error of between -5 and 5 degrees to each of the 6 joints of our arm. A total of 91% of the runs converged, and the average mean-square-error over the joints for each run was 0.83 degrees.

V. Conclusion and Future work

We have proposed an approach for vision based pose estimation of a robot hand or full body pose. The method is based on virtual visual servoing using a CAD model of the robot. The method combines 2-D image features with depth features. The method can be applied to estimate either the pose or the full configuration of a robot. We presented experimental, demonstrating the performance of the approach on both a mobile humanoid robot and a stationary manipulator.

Our experiments show that considering three-dimensional features which can be easily obtained from RGB-D images significantly improves performance when tracking robots, especially with respect to the perception of the distance from the camera to the robot. We have successfully applied the method to the tracking of a walking humanoid, as can be seen in the accompanying video. We also showed that the method can be used to refine the estimation for the joints of a robotic manipulator, where limitations in the hardware introduce uncertainties.

However, when combining both errors in the transformation for the base and in the joint configuration, the current method is stable only for limited ranges of errors. We need further studies on the relative benefits of 3D features depending on how large these errors are. Preliminary tests show that the 3D features used are complimentory. Whereas SURF features are most valuable for large errors, edges are important when errors are small. This leads to the conclusion that a system could benefit from varying the contribution of different features depending on how far you are from converging. Our plan is to continue in this direction, and gradually increase the radius of convergence, while keeping the same high accuracy.

References

[1] T. Feix, H. Schmiedmayer, J. Romero, and D. Kragic, "A comprehensive grasp taxonomy," in *In Robotics, Science and Systems: Workshop on understanding the human hand for advancing robotic manipulation*, 2009.

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

[3] C. Smith, Y. Karayiannidis, L. Nalpantidis, X. Gratal, P. Qi, D. V. Dimarogonas, and D. Kragic, "Dual arm manipulation - a survey," *Robotics and Autonomous Systems*, vol. 60, no. 10, pp. 1340–1353, 2012.

[4] H. Kato and M. Billinghurst, "Developing AR applications with ARToolkit," in *ISMAR*. IEEE Computer Society, 2004, p. 305.

[5] M. Popovic, D. Kraft, L. Bodenhagen, E. Baseski, N. Pugeault, D. Kragic, T. Asfour, and N. Krüger, "A strategy for grasping unknown objects based on co-planarity and colour information," *Robotics and Autonomous Systems*, vol. 58, no. 5, pp. 551–565, 2010.

[6] M. Irani and P. Anandan, "About direct methods," *Vision Algorithms: Theory and Practice*, pp. 267–277, 2000.

[7] M. Meilland, A. Comport, and P. Rives, "Real-time dense visual tracking under large lighting variations," in *British Machine Vision Conference, University of Dundee*, vol. 29, 2011.

[8] G. Caron, A. Dame *et al.*, "L'information mutuelle pour l'estimation visuelle directe de pose," in *Actes de la conférence RFIA 2012*, 2012.

[9] R. Rusu, G. Bradski, R. Thibaux, and J. Hsu, "Fast 3d recognition and pose using the viewpoint feature histogram," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 2155–2162.

[10] A. Aldoma, M. Vincze, N. Blodow, D. Gossow, S. Gedikli, R. Rusu, and G. Bradski, "Cad-model recognition and 6dof pose estimation using 3d cues," in *IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, 2011, pp. 585–592.

[11] K. Lai, L. Bo, X. Ren, and D. Fox, "A scalable tree-based approach for joint object and pose recognition," in *Twenty-Fifth Conference on Artificial Intelligence (AAAI)*, 2011.

[12] M. Arie-Nachimson and R. Basri, "Constructing implicit 3d shape models for pose estimation," in *IEEE 12th International Conference on Computer Vision*, 2009, pp. 1341–1348.

[13] C. Harris and M. Stephens, "A combined corner and edge detector," in *Alvey vision conference*, vol. 15. Manchester, UK, 1988, p. 50.

[14] K. Mikolajczyk and C. Schmid, "An affine invariant interest point detector," *Computer Vision—ECCV 2002*, pp. 128–142, 2002.

[15] ——, "A performance evaluation of local descriptors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 10, pp. 1615–1630, 2005.

[16] R. Haralick, D. Lee, K. Ottenburg, and M. Nolle, "Analysis and solutions of the three point perspective pose estimation problem," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 1991, pp. 592–598.

[17] R. Haralick, H. Joo, C. Lee, X. Zhuang, V. Vaidya, and M. Kim, "Pose estimation from corresponding point data," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 19, no. 6, pp. 1426–1446, 1989.

[18] D. Oberkampf, D. DeMenthon, and L. Davis, "Iterative pose estimation using coplanar points," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 1993, pp. 626–627.

[19] A. Ansar and K. Daniilidis, "Linear pose estimation from points or lines," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 5, pp. 578–589, 2003.

[20] T. Mörwald, J. Prankl, A. Richtsfeld, M. Zillich, and M. Vincze, "Blort–the blocks world robotic vision toolbox," in *Proc. ICRA Workshop Best Practice in 3D Perception and Modeling for Mobile Manipulation*, 2010.

[21] A. I. Comport, É. Marchand, M. Pressigout, and F. Chaumette, "Real-time markerless tracking for augmented reality: The virtual visual servoing framework," *IEEE Trans. Vis. Comput. Graph*, vol. 12, no. 4, pp. 615–628, 2006.

[22] X. Gratal, J. Romero, and D. Kragic, "Virtual visual servoing for real-time robot pose estimation," in *Proceedings of the 18th IFAC world congress*, 2011.

[23] D. Kragic and V. Kyrki, "Initialization and system modeling in 3-d pose tracking," in *IEEE International Conference on Pattern Recognition*, Hong Kong, 2006, pp. 643–646.

[24] P. Scovanner, S. Ali, and M. Shah, "A 3-dimensional sift descriptor and its application to action recognition," in *Proceedings of the 15th international conference on Multimedia*. ACM, 2007, pp. 357–360.

[25] C. Teuliere and E. Marchand, "Direct 3d servoing using dense depth maps," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 1741–1746.

[26] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," *Computer Vision–ECCV 2006*, pp. 404–417, 2006.

[27] S. Hutchinson, G. Hager, and P. Corke, "A tutorial on visual servo control," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 5, pp. 651–670, 1996.