Virtual Visual Servoing for Real-Time Robot Pose Estimation

Xavi Gratal * Javier Romero ** Danica Kragic ***

* *e-mail: gratal@gmail.com*
** *e-mail: jrgn@kth.se*
*** *e-mail: dani@kth.se*

Abstract:
We propose a system for markerless pose estimation and tracking of a robot manipulator. By tracking the manipulator, we can obtain an accurate estimate of its position and orientation necessary in many object grasping and manipulation tasks. Tracking the manipulator allows also for better collision avoidance. The method is based on the notion of virtual visual servoing. We also propose the use of distance transform in the control loop, which makes the performance independent of the feature search window.

1. INTRODUCTION

The ability of robot systems to grasp and manipulate objects has become one of the most important areas in robotics. Most of the state-of-the-art systems use visual/camera input as it offers rich information about the state of the environment such as number of objects, their positions, categorical information, etc. For the accurate control of a robotic manipulator using visual information, it is necessary to know the position and orientation (*pose*) of the manipulator with respect to the camera and/or the object it is supposed to manipulate. A rough estimation of the pose can be obtained from the kinematics of the manipulator and calibration camera-robot. However, sometimes a kinematic model may not be accurate or available at all. Also, in some cases it is desirable to have a very accurate estimation of the pose.

Classically, vision based control — *visual servoing* (Kragic and Christensen [2001]) has been used to align the robot hand with the object prior to applying a grasp to it. The most common solution to this problem is to place fiducial markers on the robotic manipulator. For cases in which the manipulator has a big planar surface, ART tags (Kato and Billinghurst [2004]) are a common choice, see Figure 1(a). However, more dexterous manipulators resembling the human hand may not have a surface to which such a marker can be attached. In these cases one possibility is to rigidly attach an easily trackable object to the wrist, like the red sphere in Figure 1(b). However, the usage of markers can limit substantially the mobility of the manipulator, since the markers have to remain visible in the camera(s). Moreover, the control of the manipulator's end-effector depends on the calibration accuracy of the transformation that links the marker to the end-effector's coordinate system. In other words, if that transformation changes, the system needs to be recalibrated.

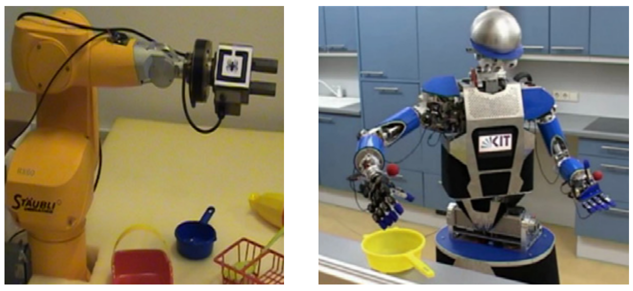

(a) ART marker (b) Red spherical markers

Fig. 1. Different markers on robotic hands. Images extracted from our previous work Popovic et al. [2010].

We propose an approach that alleviates this problem by tracking the manipulator itself instead of a marker placed on it. By tracking the manipulator, we obtain an accurate estimate of its pose and we do not need to worry about the visibility constraints as in the case of markers. Tracking the manipulator allows us to avoid collisions of the end-effector in a more precise way than by tracking just a marker.

The first contribution of this paper is a real-time model-based tracking system that alleviates the need for special markers and is also applicable to complex shapes such as the three fingered Schunk Dexterous Hand, see Fig. 3. We show how through the process of rendering the whole model, occlusions between different parts of the manipulator can be properly dealt with. The second contribution is related to the use of visual features. Instead of searching for edges in the neighbourhood of the edge in the virtual image, we use the distance transform, which makes the performance independent of the search window.

2. RELATED WORK

Approaches for tracking objects of complex geometry can be divided in two groups: appearance-based and model-based methods. The first is based on comparing camera

[1] This work is supported by EU through the project GRASP, IST-FP7-IP-215821 ,Swedish Foundation for Strategic Research and UKF-Unity through Knowledge Fund.The authors are with the Computational Vision and Active Perception Lab, Centre for Autonomous Systems, CSC-KTH, Stockholm, Sweden.

images with a huge database of stored images with annotated poses (Lepetit et al. [2004]). The second relies on the use of a geometrical model of the object and perform tracking based on optical flow (Drummond and Cipolla [2002]). There have also been examples that integrate both of these approaches (Kyrki and Kragic [2005]).

The initialization of the tracking process represents an important problem by itself. This can be done by first localizing the object in the image and then using a global pose estimation step. In our previous work, we have demonstrated how the initialization can be done for objects in general settings (Kragic and Kyrki [2006]). In the case of a manipulator, the initialization step can be performed by comparing the renderization of its model with the initial image where the pose for the model is given through the forward kinematics. Thus, for the initial rendering of the 3D CAD model, the position and orientation are assumed to be known. The renderization or the synthetic image of the manipulator is then compared to the camera image, through a process based on image based control though virtual visual servoing (VVS) (Comport et al. [2005]).

In this paper, we thus focus on improving the pose estimation of the manipulator by VVS. The system is also applicable to the estimation of the pose of each finger of the robot hand with respect to the wrist; however, for the purposes of this paper, we consider the finger joint angles and opening of the robot hand to be constant.

The problem of estimating the full pose of the manipulator can be seen as an instance of the general object pose estimation problem. An important aspect to take into account is the type of visual features that can be used for tracking. The use of different types of interest points such as corners is common (Lepetit et al. [2004]). While this approach works well for textured objects, it is not suitable for our robotic manipulator, which is non-textured and has highly specular surfaces. Some model-based methods have been already used for tracking of a robotic manipulator. The efforts by Comport et al. [2006], Sorribes et al. [2010] and our previous work Comport et al. [2005] use 3D edges or ellipses as the input to a VVS loop.

In our recent work Romero et al. [2010], we demonstrate how pose estimation can be performed for a more complex object such as a human hand. The major contribution of the work in Romero et al. [2010] is the use of a discriminative machine learning approach for obtaining a real-time tracking performance of an object with 27 degrees of freedom. The problem considered in this paper has a lower dimensionality and a more accurate guess of the initial pose. This makes it possible to adopt a real-time generative approach that renders the last pose of the object in each frame and estimates the new pose through a process of error minimization.

3. METHODOLOGY

The pose of an object is denoted by $\mathbf{X}(\mathbf{R}, \mathbf{t})$ where $\mathbf{t} \in \mathcal{R}(3)$, $\mathbf{R} \in \mathbf{SO}(3)$. The set of all poses that the robot's end–effector can attain is denoted with $\mathcal{T}_G \subseteq \mathbf{SE}(3) = \mathcal{R}(3) \times \mathbf{SO}(3)$. A positioning task in general can then be represented by a function

$$\mathbf{e} : \mathcal{S} \rightarrow \mathcal{R}(n) \tag{1}$$

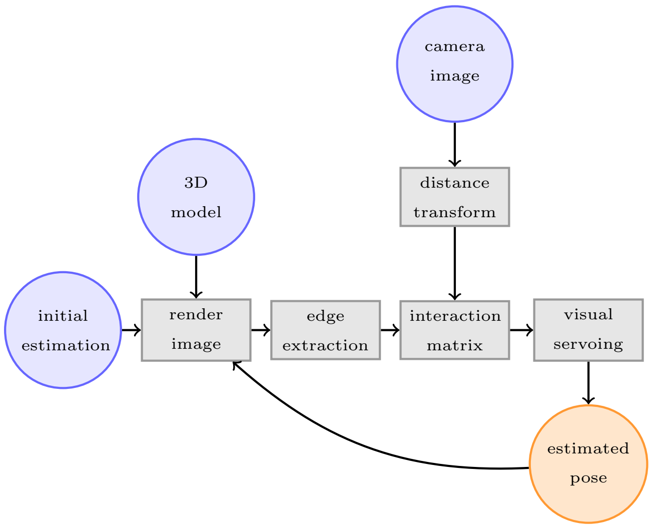


Fig. 2. Outline of the proposed model-based tracking system.

where $\mathcal{S}$, in the case of position based visual servoing, may represent the task space of the end–effector, $\mathcal{T}_G$. In this case, $\mathbf{e}$ is referred to as the *kinematic error function*, Hutchinson et al. [1996]. The dimension n of $\mathcal{R}$ in (Eq. 1) depends on the number of degrees of freedom (DOF) of the robot constrained by $\mathbf{e}$ with $n \leq m$ (m denotes DOF of the manipulator). In the case of image based visual servoing, $\mathcal{S}$ represents *image feature parameter space*, $\mathcal{F}$ and $n \leq k$ where k is the dimension of $\mathcal{F}$.

Pose estimation considers a computation of a rotation matrix (orientation) and a translation vector of the object (position), $\mathbf{X}(\mathbf{R}, \mathbf{t})$:

$$\mathbf{X} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & T_X \\ r_{21} & r_{22} & r_{23} & T_Y \\ r_{31} & r_{32} & r_{33} & T_Z \end{bmatrix} \tag{2}$$

The equations used to describe the projection of a three–dimensional model point $\mathbf{Q}$ into homogeneous coordinates of the image point $[x\ y]^T$ are:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \mathbf{R}\,[\mathbf{Q} - \mathbf{t}] \quad \text{with} \quad \begin{bmatrix} wx \\ wy \\ w \end{bmatrix} = \mathbf{P} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{3}$$

where $\mathbf{P}$ is a 3x3 matrix representing the internal camera parameters matrix including focal length and aspect ratio of the pixels, w is the scaling factor, $\mathbf{R}$ and $\mathbf{t}$ represent the rotation matrix and translation vector, respectively.

Our approach to pose estimation and tracking is based on virtual visual servoing where a rendered model of the hand is aligned with the current image of the hand. The outline of the system is presented in Fig. 2. In order to achieve the alignment, we can either control the position of the object to bring it to the desired pose or move the *virtual* camera so that the image perceived by the camera corresponds to the current camera image, denoted as *real* camera image in the rest of the paper. In this paper, we adopt the first approach where we render synthetic images by incrementally changing the pose of the tracked object. In the first iteration, the position is given based on the forward kinematics. Then, we extract visual features from the rendered image, and compare them with the features extracted from the current camera image. The details about the features that are extracted are given in the next section.

Based on the extracted features, we define a difference or an error vector between the desired values for the features and the current ones. Based on this error vector, we can estimate the incremental change in pose in each iteration following the classical image based servoing control loop. This process continues until the difference vector is smaller than a certain threshold. Each of the steps is explained in more detail in the following subsections. Our current implementation and experimental evaluation is performed for the Schunk Dexterous hand, shown in Fig. 3.

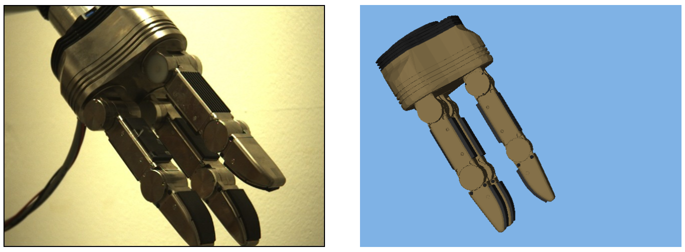


Fig. 3. The Schunk Dexterous Hand and a renderization of our CAD model.

3.1 Virtual image generation

As we mentioned before, we use a realistic 3D model of the object as input for our system. This adds the challenge of having to render this model at a high frame rate, since our system runs in real time, and several visual servoing iterations must be performed for every frame that we obtain from the cameras.

To render the image, we use a projection matrix $\boldsymbol{P}$, which corresponds to the internal parameters of the real camera, and a modelview matrix $\boldsymbol{M}$, which consists of a rotation matrix and a translation vector. The modelview matrix is then estimated in the visual servoing loop.

One of the most common CAD formats for objects such as robotic hands are Inventor files. There are a number of rendering engines which can deal with such models, but none of them had the performance and flexibility that we needed. For that we developed a new scenegraph engine, specific for this application, which focused on rendering offline images at a high speed. It was developed directly over OpenGL. With this engine, we can obtain about 1000 frames per second, and is not, at the moment, the bottleneck of the system.

We render the image without texture or lighting, since we are only interested in silhouette of the model. We also save the depth map produced by the rendering process, which will be useful later for the estimation of the jacobian.

3.2 Features

As mentioned before, the virtual and the real image will be compared in terms of visual features. The choice of those features is crutial because it can affect drastically the speed and the reliability of the system. The most important characteristics of such a feature are the following:

- Speed: The feature will be computed once per frame on the real image and once per iteration (several per frame) on the virtual image. Therefore the feature computation could become the system's performance bottleneck. Since the virtual image features are computed much more often, features with an asymmetric computational time can be beneficial.
- Robustness towards illumination changes: Our robotic hand model, as many other models, does not have a proper model for the specularity and reflectance of the hand surface. Consequently, the selected feature should not depend on the specular reflections or the reflected color on the hand surface. The main characteristic that the feature should reflect is shape.
- Directed error: Some formulations of the error computation in visual servoing depend on the direction which minimizes the error. Therefore, it is preferable to have features where the error direction can be extracted, so that it is possible to know how this error will change with respect to motion of the object.
- Robustness for non-textured models: Robot manipulators are usually plain colored or metalic; therefore features which rely on texture should be avoided.

Harris corners (Harris and Stephens [1988]) or SIFT (Lowe [1999]) are features which can be extracted fast and reliably. However, they depend heavily on texture and are therefore not suitable for our application. In Comport et al. [2006] the distance between point and a line, and between ellipse and a line is used. Those features by themselves do not provide a directed error. But more importantly, they are not suitable for complex models with curved lines. Moreover, the creation of the virtual model is either manual or involves detecting linear or ellipsoidal patches, which may be non-trivial depending on the complexity of the model.

Chamfer distance (Butt and Maragos [1998]) can be considered as a generalisation of the features used in Comport et al. [2006]. Instead of measuring distance between shapes and points, it directly measures the distance between points and their closest match. The sets of points $\mathcal{U}, \mathcal{V}$ considered are usually edges extracted with a Sobel or Canny operator from images U, V. While standard chamfer distance measures the distance between two whole shapes (as the average of the point-to-point distances), we will focus on the individual distances:

$$d_{cham}(u) = \min_{v \in \mathcal{V}} ||u - v||, u \in \mathcal{U} \tag{4}$$

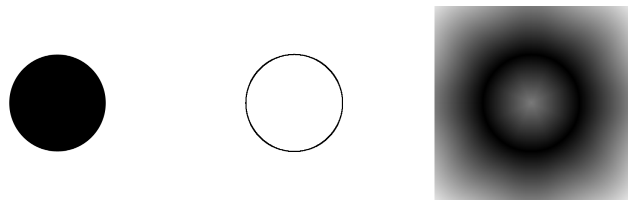
This is a good feature candidate because its computation is fast and it is robust to illumination and color changes. In its original design it does not provide directed error, but that characteristic can be easily added with negligible computational cost:

$$o_{cham}(u) = \angle(u - v), u \in \mathcal{U}, v = \arg\min_{v \in \mathcal{V}} ||u - v|| \tag{5}$$

We want to compute the similarities between one real image and multiple virtual images for each frame. Therefore it makes sense to precompute the distance from each pixel from the image to the closest edge for the real image.That is the so called distance transform Borgefors [1986]:

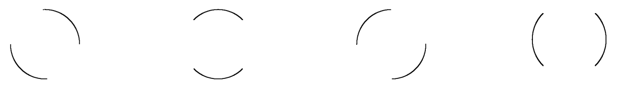
$$d_{transf}(p) = \min_{u \in \mathcal{U}} ||p - u||, p \in U \tag{6}$$

$$d_{cham}(u) = d_{transf}(v), v \in \mathcal{V} \tag{7}$$



(a) Image (b) Edge detection (c) Dist. Transform

Fig. 4. Image, edge detection and distance transform of a circle



(a) $0° - 90°$ (b) $45° - 135°$ (c) $90° - 180°$ (d) $135° - 225°$

Fig. 5. Edge detection divided into 4 overlapping ranges

Once we compute d_{transf}, we can compute $d_{cham}(v)$ multiple times for different images rather fast ($\mathcal{O}(n)$ with the number of edge points). Therefore, we can compute the distance transform d_{transf} of the real image and then check how similar are the virtual images in linear time with the number of edge points of each virtual image. The simplest form of this feature has a problem; any edge, no matter its shape or orientation, will have a low distance value if positioned in a zone with high density of edges. This can be alleviated by matching edges only if they have similar orientation. For that purpose we use the horizontal and vertical edge images $\mathcal{U}_h, \mathcal{U}_v$ to divide the edge image $\mathcal{U}$ into 8 channels of different overlapping orientation ranges (see Figure 5 for an example with four channels). This method is similar to the 3D distance transform described in Liu et al. [2010], but in our case we want to discriminate, rather than use some weighted metric for edges with different orientations, so we calculate the distance transform over overlapping channels, effectively rejecting edges with a different orientation.

$$\forall u \in \mathcal{U}_c : \lfloor \frac{\arctan(\frac{u_v}{u_h})N}{2\pi} \rfloor = c, N = 8 \tag{8}$$

$$d_{transf}(p) = \min_{u \in \mathcal{U}_c} ||p - u||, p \in U_c \tag{9}$$

$$d_{cham}(u) = d_{transf}(v), v \in \mathcal{V}_c, u \in \mathcal{U}_c \tag{10}$$

This means that edges with different orientations do not affect the distance transform of each other, and therefore they do not get "attracted" by those edges.

Interior edges are more sensitive to illumination changes, shadows, reflections and model imperfections than exterior edges. That is why we decided to extract only the silhouette of the virtual model and try to localize that silhouette in the real image. This avoided the interior edges of the model getting trapped in local minima on the real image, resulting in a more robust matching. It also makes the computation of the chamfer distance faster. However, it can be argued that it makes the virtual model representation ambiguous due to the lack of details in the interior of the silhouette. During our tests this did not represent a problem since the initial position is close enough to the real pose in order to disambiguate between similar silhouettes.

3.3 Visual servoing

Once the features have been extracted, we can use a classical visual servoing approach to calculate the correction to the pose, Hutchinson et al. [1996]. Vision-based control systems can use two different approaches: *Position-based* control uses image data to extract a series of 3D features, and control is performed in the 3D Cartesian space. On the other hand, in *image-based* control, the image features are directly used to control the robot motion. In our case, since the features we are using are distances between edges in an image, for which we have no depth information, we will be using an *image-based* approach.

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 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 between the edges in the real and synthetic images:

$$\boldsymbol{s}(t) = [d_1, d_2, \ldots, d_n]^T \tag{11}$$

then $\dot{\boldsymbol{s}}(t)$ will be the rate of change of these distances.

The movement of the manipulator (in this case, the virtual manipulator) can be described by a translational velocity $\boldsymbol{T}(t) = [T_x(t), T_y(t), T_z(t)]^T$ and a rotational velocity $\boldsymbol{\Omega}(t) = [\omega_x(t), \omega_y(t), \omega_z(t)]^T$. Together, they form a velocity screw:

$$\dot{\boldsymbol{r}}(t) = [T_x, T_y, T_z, \omega_x, \omega_y, \omega_z]^T \tag{12}$$

We can then define the image jacobian or interaction at a certain instant as $\boldsymbol{J}$ so that:

$$\dot{\boldsymbol{s}} = \boldsymbol{J}\dot{\boldsymbol{r}} \tag{13}$$

$$\boldsymbol{J} = \left[\frac{\partial \boldsymbol{s}}{\partial \boldsymbol{r}}\right] = \begin{bmatrix} \frac{\partial d_1}{\partial T_x} & \frac{\partial d_1}{\partial T_y} & \frac{\partial d_1}{\partial T_z} & \frac{\partial d_1}{\partial \omega_x} & \frac{\partial d_1}{\partial \omega_y} & \frac{\partial d_1}{\partial \omega_z} \\ \frac{\partial d_2}{\partial T_x} & \frac{\partial d_2}{\partial T_y} & \frac{\partial d_2}{\partial T_z} & \frac{\partial d_2}{\partial \omega_x} & \frac{\partial d_2}{\partial \omega_y} & \frac{\partial d_2}{\partial \omega_z} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial d_n}{\partial T_x} & \frac{\partial d_n}{\partial T_y} & \frac{\partial d_n}{\partial T_z} & \frac{\partial d_n}{\partial \omega_x} & \frac{\partial d_n}{\partial \omega_y} & \frac{\partial d_n}{\partial \omega_z} \end{bmatrix} \tag{14}$$

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, we need to compute $\dot{\boldsymbol{r}}(t)$ given $\dot{\boldsymbol{s}}(t)$ in order to correct our pose estimation.

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

$$\dot{\boldsymbol{r}} = \boldsymbol{J}^{+}\dot{\boldsymbol{s}} \tag{15}$$

where $\boldsymbol{J}^{+}$ is the pseudoinverse of $\boldsymbol{J}$, which can be calculated as:

$$\boldsymbol{J}^{+} = (\boldsymbol{J}^{T}\boldsymbol{J})^{-1}\boldsymbol{J}^{T}. \tag{16}$$

The goal for our task is to have all the edges in our synthetic image match edges in the real image, so the target value for each of the features is 0. Then, we can define the error function as the following, which leads us to the simple proportional control law where K is the gain parameter:

$$\boldsymbol{e}(\boldsymbol{s}) = \boldsymbol{0} - \dot{\boldsymbol{s}} \tag{17}$$
$$\dot{\boldsymbol{r}} = -K\boldsymbol{J}^{+}\boldsymbol{s} \tag{18}$$

3.4 Estimation of the jacobian

To estimate the jacobian, we need to calculate the partial derivatives of the feature values with respect to each of the motion components. When features are the position of points or lines, it is possible to find an analytical solution for the derivatives.

In this case, 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 small change in the relevant direction affects the value of the feature.

As said before, we obtained a depth map while rendering the model. This depth map allow us to obtain the 3D point corresponding to each of the edges. If $D(\boldsymbol{u})$ is the distance to an edge for a projected point $\boldsymbol{u}$ we can calculate the derivative for a model point $\boldsymbol{U}$ with respect to T_x as:

$$\frac{D\big(\boldsymbol{PM}(\boldsymbol{U}+\epsilon\boldsymbol{x})\big) - D\big(\boldsymbol{PMU}\big)}{\epsilon} \tag{19}$$

where ϵ is an arbitrary small number and $\boldsymbol{x}$ is a unitary vector in the x direction. A similar process is applied to each of the motion components.

4. EXPERIMENTAL EVALUATION

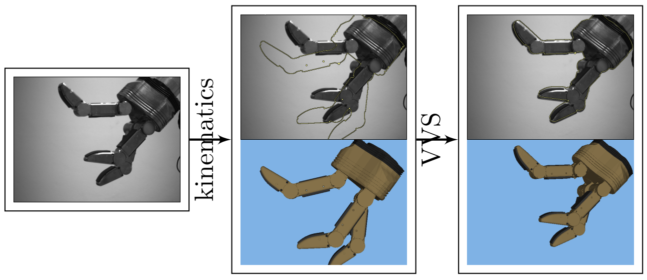


Fig. 6. Alignment procedure; the kinematics give a first estimation of the manipulator pose, and it is refined by VVS

The goal of the experimental evaluation is to test the performance of the system according to a number of aspects such as the applicability of virtual visual servoing, convergence speed, robustness of the system wrt a cluttered background, etc.

The first step in our experimental evaluation was to test the system in a simulated setup using synthetic images only. The synthetic images were created from the same CAD model we use for the virtual images in the real system. This ensures that the visual input to the VVS framework matches the model perfectly. In this way, we test the performance of the VVS framework, disregarding problems like visual differences between real and virtual manipulator and image noise.

Since the ground truth in the case of a simulated setup is available, we perturbed the final pose with errors inn both translation and orientation. This corresponds to the errors from forward kinematics. As long as these errors were kept small, the method converged, and the maximum estimation error was 0.4 mm for the position in the direction which is perpendicular to the optical axis. In the direction that corresponds to the depth in the image and thus has less impact to the appearance of the image, the errow was 4 mm. Finally the error in rotation was 0.5°. We used these experiments to fine-tune the parameters of the system working with real camera images.

The evaluation on real camera images adds difficulties such as model imperfections, noise in the edge detection, edges variability depending on illumination changes, etc. We used the hand mounted on a KUKA industrial arm, as shown in Figure 7. This allowed us to have a good initial estimation of the pose, that we could then refine with our method. We used a camera that could provide 640x480 images at 15 fps.

It was at this point that we introduced the separation of the distance transform in several channels depending on the orientation of the edges. This was not necessary for the simulated setup, but the real images contained many edges not corresponding to the contour of the manipulator, which ‘attracted’ the edges of the synthetic image. This was alleviated by introducing the orientation channels, so that edges were only ‘attracted’ by edges with similar orientation.

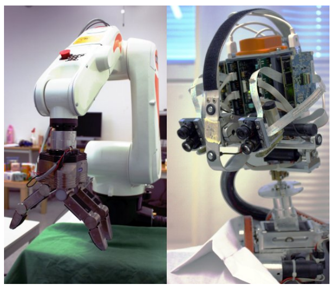


Fig. 7. Our setup with a KUKA 6DOF industrial arm with the Schunk Dexterous Hand and a humanoid robot head.

Due to the absence of markers in the hand, we could not provide ground truth data for our real experiments, and therefore the evaluation has been done in qualitative terms. In these qualitative tests we read the initial pose from the kinematic chain and camera-robot calibration, and used it as the starting point for the system. The system converges in general, but sometimes to local minima. When the convergence was successful, we also used the system for tracking. We did this by using the pose estimated

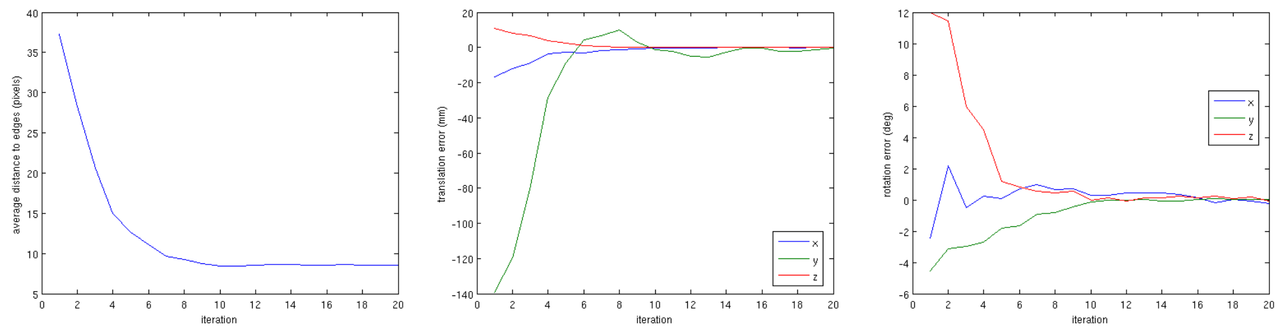


Fig. 8. Evolution of the error in the image space and corrections in translation and rotation.

for the previous frame as initial pose, and then moving the robot at different speeds. The system performed well for small movements of the hand.

The result for one of the runs of the system is shown in Fig. 6. As we can see, the initial pose obtained from calibration is quite off, and the system still manages to converge. The final result, however, is not perfect, and there are some edges which are not exactly matched. This is due, in part, to imprecisions in the joint configuration of the hand, which is read from the motors, and then used when rendering the virtual images. The evolution and correction of the error is shown in Fig. 8.

5. FUTURE WORK

We proposed a system for markerless pose estimation and tracking of a robot manipulator. By tracking the manipulator, we can obtain an accurate estimate of its pose and we do not need to worry about the visibility constraints as in case of markers. Tracking the manipulator allows us to avoid collisions of the end-effector in a more precise way compared to tracking a marker. We also propose the use of distance transform in the control loop, which makes the performance independent of the feature search window.

The system shows good performance in a realistic robotic setup considering a state-of-the-art robot hand. We have discussed how a wrong forward kinematics can affect the performance of the system. Our idea to add the joint values to the parameters to be estimated in the control loop and deal with the problem of wron initialization and tracking. Another future line of work is to adapt this method to use it with stereo imagery. By using pairs of images coming from two calibrated cameras, it will be possible to increase the robustness, by matching the edges to the images obtained in both cameras. Also, by taking into account the depth map which can be obtained from such a setup, it will be possible to match only edges that are at similar depths, in a way that is similar to how orientation is handled in the current method.

REFERENCES

G. Borgefors. Distance transformations in digital images. *Computer Vision, Graphics and Image Processing*, 34: 344–371, 1986.

M. A. Butt and P. Maragos. Optimum design of chamfer distance transforms. *IEEE Transactions on Image Processing*, 7(10):1477–1484, 1998.

A. I. Comport, D. Kragic, E. Marchand, and F. Chaumette. Robust real-time visual tracking: Comparison, theoretical analysis and performance evaluation. pages 2841 – 2846, apr. 2005.

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*, 12(4):615–628, 2006.

T. Drummond and R. Cipolla. Real-time visual tracking of complex structures. *IEEE Trans. PAMI*, 24(7):932–946, 2002.

C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey Vision Conf*, pages 189–192, 1988.

S. Hutchinson, G.D. Hager, and P.I. Corke. A tutorial on visual servo control. *IEEE Transactions on Robotics and Automation*, 12(5):651–670, 1996.

H. Kato and M. Billinghurst. Developing AR applications with ARToolkit. In *ISMAR*, page 305, 2004.

D. Kragic and H. I. Christensen. Cue integration for visual servoing. *IEEE Transactions on Robotics and Automation*, 17(1):18–27, February 2001.

D. Kragic and V. Kyrki. Initialization and system modeling in 3-d pose tracking. In *ICPR*, pages 643–646, Hong Kong, 2006.

V. Kyrki and D. Kragic. Integration of model-based and model-free cues for visual object tracking in 3d. In *ICRA*, pages 1566–1572, 2005.

V. Lepetit, J. Pilet, and P. Fua. Point matching as a classification problem for fast and robust object pose estimation. In *CVPR*, pages II: 244–250, 2004.

M. Liu, O. Tuzel, A. Veeraraghavan, R. Chellappa, A. Agrawal, and H. Okuda. Pose estimation in heavy clutter using a multi-flash camera. In *ICRA*, pages 2028–2035. IEEE, 2010.

D. G. Lowe. Object recognition from local scale-invariant features. In *ICCV*, pages 1150–1157, 1999.

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 coplanarity and colour information. *Robotics and Autonomous Systems*, 58(5):551–565, 2010.

J. Romero, H. Kjellström, and D. Kragic. Hands in action: real-time 3D reconstruction of hands in interaction with objects. In *ICRA*, pages 458–463. IEEE, 2010.

J. J. Sorribes, M. Prats, and A. Morales. Visual tracking of a jaw gripper based on articulated 3D models for grasping. In *ICRA*, pages 2302–2307. IEEE, 2010.