D. Kragic
H. I. Christensen
Centre for Autonomous Systems
Computational Vision and Active Perception
Royal Institute of Technology
SE–100 44 Stockholm, Sweden
danik@nada.kth.se
hic@nada.kth.se

Robust Visual Servoing

Abstract

For service robots operating in domestic environments, it is not enough to consider only control level robustness; it is equally important to consider how image information that serves as input to the control process can be used so as to achieve robust and efficient control.

In this paper we present an effort towards the development of robust visual techniques used to guide robots in various tasks. Given a task at hand, we argue that different levels of complexity should be considered; this also defines the choice of the visual technique used to provide the necessary feedback information. We concentrate on visual feedback estimation where we investigate both two- and three-dimensional techniques.

In the former case, we are interested in providing coarse information about the object position/velocity in the image plane. In particular, a set of simple visual features (cues) is employed in an integrated framework where voting is used for fusing the responses from individual cues. The experimental evaluation shows the system performance for three different cases of camera–robot configurations most common for robotic systems.

For cases where the robot is supposed to grasp the object, a two-dimensional position estimate is often not enough. Complete pose (position and orientation) of the object may be required. Therefore, we present a model-based system where a wire-frame model of the object is used to estimate its pose. Since a number of similar systems have been proposed in the literature, we concentrate on the particular part of the system usually neglected—automatic pose initialization. Finally, we show how a number of existing approaches can successfully be integrated in a system that is able to recognize and grasp fairly textured, everyday objects. One of the examples presented in the experimental section shows a mobile robot performing tasks in a real-word environment—a living room.

KEY WORDS—**AUTHOR: PLEASE PROVIDE**

The International Journal of Robotics Research
Vol. 22, No. 10, October 2003, pp. xxx-xxx,
©2003 Sage Publications

1. Introduction

Robotics is gradually expanding its application domain beyond manufacturing. In manufacturing settings, it is possible to engineer the environment so as to simplify detection and handling of objects. In an industrial context, this is often achieved through careful selection of background color and lighting. As the application domain is expanded, the use of engineering to simplify the perception problem is becoming more difficult. It is no longer possible to assume a given setup of lighting and a homogeneous background. Recent progress in *service robotics* shows a need to equip robots with facilities for operation in everyday settings where the design of computer vision methods to facilitate robust operation and to enable interaction with objects has to be reconsidered.

In this paper we consider the use of computational vision for manipulation of objects in everyday settings. The process of manipulation of objects involves all aspects of recognition/detection, servoing to the object, alignment and grasping. Each of these processes have typically been considered independently or in relatively simple environments (Hutchinson, Hager, and Corke 1996). Through careful consideration of the task constraints and combination of multiple methods it is, however, possible to provide a system that exhibits robustness in realistic settings.

The paper starts with a motivation that argues for integration of visual processes in Section 2. A key competence for robotic grasping is recognition/detection of objects as outlined in Section 3. Once the object has been detected, the reaching phase requires use of a coarse servoing strategy which can be based on "simple" visual features. Unfortunately, "simple" visual features suffer from a limited generality requiring therefore an integration of multiple cues to achieve robustness as described in Section 4. Once the object has been recognized and an approximate alignment has been achieved, it is possible to use model-based methods for accurate interaction with the object as outlined in Section 5. Finally, all of these components are integrated into a complete system and used for a series of experiments. The results from these experiments are presented and discussed in Section 6. The overall approach and the associated results are discussed in Section 7.

2. Motivation

Robotic manipulation in an everyday setting is typically carried out in the context of a task such as "please, fetch the cup from the dinner table in the living room" or "please, fetch the rice package from the shelf" (see Figure 1). To execute such a manipulation task, we use a mobile platform with a manipulator on the top. The robot is equipped with facilities for automatic localization and navigation (Petersson et al. 2002), allowing it to arrive at a position in the vicinity of the dinner table. From there the robot is required to carry out the following.

1. *Recognize* a cup on the table (discussed in Section 3).
2. *Transportation*—servo to the vicinity of the cup, potentially involving both platform and manipulator motion (discussed in Section 4).
3. Estimate the pose (position and orientation) of the object (Section 5).
4. *Alignment*—servo to the object to allow grasping (Section 5).
5. Pick up the object and drive away.

In a typical scenario, the distance between the object and the on-board camera may vary significantly. This implies a significant uncertainty in terms of scale (i.e., the size of the object in the image). Based on the actual scale of the object in the image, a hierarchical strategy for visual servoing may be used. For distant objects ($\geq$ 1 m), there is no point in attempting to perform a full pose estimation, as the size of the object in the image typically will not provide the necessary information. A coarse position estimate is therefore adequate for an initial alignment.

A visual servoing task in general includes some form of (i) *positioning*, such as aligning the robot/gripper with the target, and (ii) *tracking*, updating the position/pose of the image features/object. Typically, image information is used to measure the error between some current and reference/desired location. Image information used to perform the task is either (i) two-dimensional, using image plane coordinates, or (ii) three-dimensional, retrieving the pose parameters of objects with respect to the camera/world/robot coordinate system. So, the robot is controlled using image information as either two- or three-dimensional, which classifies the visual servoing approaches as (i) image based, (ii) position based or (iii) hybrid visual servoing (2.5D servoing) (Kragic and Christensen 2002).

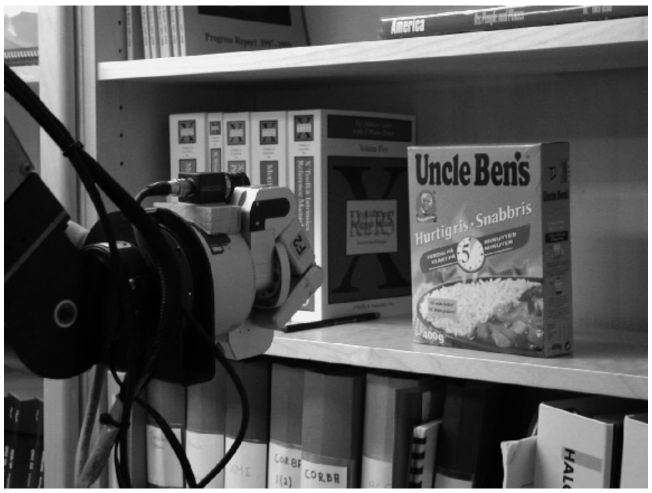


Fig. 1. An example of a robot task: fetching a rice package from a shelf.

If the object is far from the camera, due to the resolution/size of the object in the image, it is advantageous to use an image-based strategy to perform the initial alignment. Image-based servoing using crude image features such as center of mass is well suited for coarse alignment. A problem here is that visual cues such as color, edges, corners or fiducial marks in general are highly sensitive to variation in illumination, etc., and no single one of them will provide robust frame by frame estimates that allow continuous tracking. Through their integration it is however possible to achieve an increased robustness, as described in Section 4.

For the accurate alignment of the gripper with the object, we can utilize (i) a binocular camera pair, (ii) a monocular image in combination with a geometric model, or (iii) one camera in combination with a "structure-from-motion" approach. As the object has been recognized and its approximate image position is known, it is possible to utilize this information for efficient pose estimation using a wire-frame model. Model-based fitting is highly sensitive to surface texture and background noise. The availability of a good initial estimate does, however, allow robust estimation of pose even in the presence of significant noise. Once a pose estimate is available, it is possible to use 2.5D or position-based servoing for the alignment of the gripper with the objects. This is even possible in the presence of significant perspective effects, which typically is the case when operating close to objects. Model-based tracking is in general highly sensitive to surface texture and initialization, but through its integration into a system context these problems are reduced to a minimum as outlined in Section 5.

Finally, as the gripper is approaching the object it is essential to use the earlier mentioned geometric model for grasp planning, an issue not discussed in detail here. For an introduction, consult Bicchi and Kumar (2000) and Shimoga (1996).

3. Detection and Pose Estimation

Given a view of the scene, the robot should be able to find/recognize the object to be manipulated or give us an answer such as “I-did-not-find-the-cup”. This task is very complicated and hard to generalize with respect to cluttered environments and types of objects. Depending on the implementation, the recognition process may provide the partial/full pose of the object which can in return be used for generating the robot control sequence. Object recognition is a long standing research issue in the field of computational vision (Edelman 1999). We consider limited aspects of it, but demonstrate recognition in real environments, a problem that has received limited attention so far.

3.1. Appearance-Based Method

The object to be manipulated is recognized using the view-based Support Vector Machine (SVM) system presented in Roobaert, Zillich, and Eklundh (2001). The objective here is to detect and recognize everyday household objects in a scene.

The recognition step delivers the image position and approximate size of the image region occupied by the object (see Figure 2). This information is in our case used (i) to track the part of the image, the *window of attention*, occupied by the object while the robot approaches it, or (ii) as the input to the pose estimation algorithm.

3.2. Feature-Based Method

In the case of a moving object, its appearance may change significantly between frames. For this reason, we have also exploited a feature or a cue-based method. Here, color and motion were used in an integrated framework where voting was used as an underlying integration strategy. This is discussed in more detail in the next section.

4. Transportation: Coarse Visual Servoing

While approaching the object, we want to keep in the field of view, or even centered, of the image. This implies that we have to estimate the position/velocity of the object in the image and use this information to control the mobile platform.

Our tracking algorithm employs the four-step *detect–match–update–predict loop* (Figure 3). The objective here is to track a part of an image (a region) between frames. The image position of its center is denoted by $\mathbf{p} = [x\ y]^{\mathrm{T}}$. Hence, the state is $\mathbf{x} = [\ x\ y\ \dot{x}\ \dot{y}\ ]^{\mathrm{T}}$ where a piecewise constant white acceleration model is used (Bar-Shalom and Li 1993):

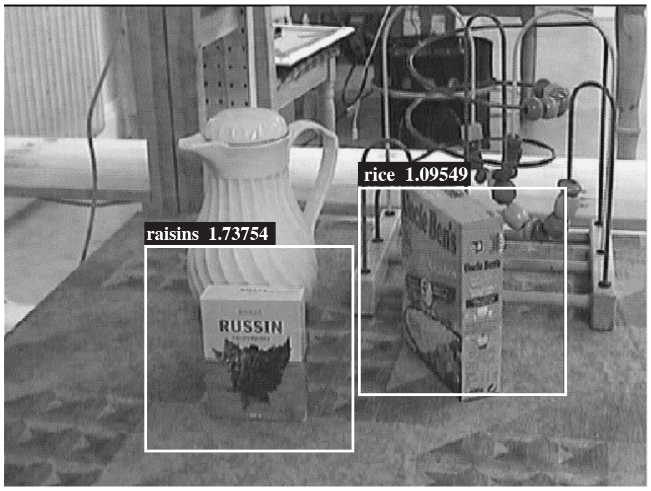


Fig. 2. An example where the recognition system successfully recognizes two of the objects. Above each rectangle there is a value representing the recognition confidence which is proportional to the distance of the object to the hyperplane of the classifier. In order to make confidence measures comparable across all the classifiers, the distances are normalized with the margin of the hyperplane of the classifier. If the confidence is greater than 1, a positive detection/recognition is assumed. If the confidence is less than 1, the classification is considered uncertain.

$$\begin{aligned}\mathbf{x}_{k+1} &= \mathbf{F}\mathbf{x}_k + \mathbf{G}\mathbf{v}_k \\ \mathbf{z}_k &= \mathbf{H}\mathbf{x}_k + \mathbf{w}_k.\end{aligned} \tag{1}$$

Here, $\mathbf{v}_k$ is a zero-mean white acceleration sequence, $\mathbf{w}_k$ is measurement noise and

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & \Delta T & 0 \\ 0 & 1 & 0 & \Delta T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix} \frac{\Delta T^2}{2} & 0 \\ 0 & \frac{\Delta T^2}{2} \\ \Delta T & 0 \\ 0 & \Delta T \end{bmatrix}, \tag{2}$$

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}.$$

For prediction and estimation, the α–β filter is used (Bar-Shalom and Li 1993)

$$\begin{aligned}\hat{\mathbf{x}}_{k+1|k} &= \mathbf{F}_k\hat{\mathbf{x}}_k \\ \hat{\mathbf{z}}_{k+1|k} &= \mathbf{H}\hat{\mathbf{x}}_{k+1|k} \\ \hat{\mathbf{x}}_{k+1|k+1} &= \hat{\mathbf{x}}_{k+1|k} + \mathbf{W}[\mathbf{z}_{k+1} - \hat{\mathbf{z}}_{k+1|k}]\end{aligned} \tag{3}$$

with

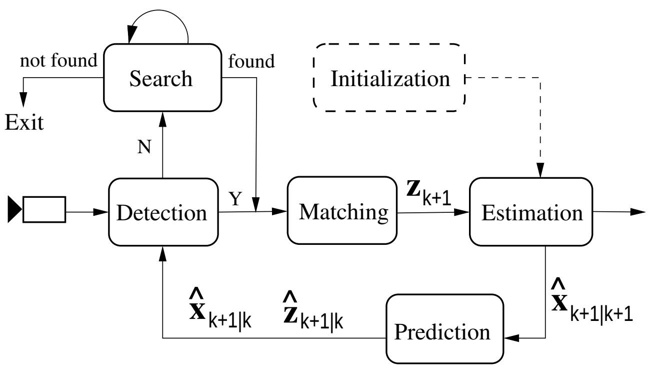


Fig. 3. A schematic overview of the tracking system.

$$\mathbf{W} = \begin{bmatrix} \alpha & 0 & \dfrac{\beta}{\Delta T} & 0 \\ 0 & \alpha & 0 & \dfrac{\beta}{\Delta T} \end{bmatrix}^{\mathrm{T}}$$

where α and β are determined using steady-state analysis.

4.1. Voting

Voting, in general, may be viewed as a method to deal with n input data objects, c_i, having associated votes/weights w_i (n input data–vote pairs (c_i, w_i)) and producing the output data–vote pair (y, v) where y may be one of c_i or some mixed item. Hence, voting combines information from a number of sources and produces outputs which reflect the consensus of the information.

The reliability of the results depends on the information carried by the inputs and, as we will see, their number. Although there are many voting schemes proposed in the literature, mean, majority and plurality voting are the most common ones. In terms of voting, a visual cue may be motion, color or disparity. Mathematically, a cue is formalized as a mapping from an action space, **A**, to the interval [0,1]:

$$c \ : \ \mathbf{A} \rightarrow [0, 1]. \tag{4}$$

This mapping assigns a *vote* or a preference to each action $a \in \mathbf{A}$, which may, in the context of tracking, be considered as the position of the target. These votes are used by a *voter* or a *fusion center*, $\delta(\mathbf{A})$. Based on the ideas proposed in Blough and Sullivan (1994) and Rosenblatt and Thorpe (1995), we define the following voting scheme:

DEFINITION 1. Weighted Plurality Approval Voting. For a group of homogeneous cues, $\mathbf{C} = \{c_1, \ldots, c_n\}$, where n is the number of cues and O_{c_i} is the output of a cue i, a weighted plurality approval scheme is defined as

$$\delta(a) = \sum_{i=1}^{n} w_i \ O_{c_i}(a) \tag{5}$$

where the most appropriate action is selected according to

$$a' = \operatorname{argmax}\{\delta(a) | a \in \mathbf{A}\}. \tag{6}$$

4.2. Visual Cues

The cues considered in the integration process are as follows.

Correlation. The standard sum of squared differences (SSD) similarity metric is used and the position of the target is found as that giving the lowest dissimilarity score

$$SSD(u, v) = \sum_n \sum_m [I(u+m, v+n) - T(m, n)]^2, \tag{7}$$

where $I(u, v)$ and $T(u, v)$ represent the gray-level values of the image and the template, respectively. To compensate for changes in the appearance of the tracked region, the template is updated on a 25 frames cycle.

Color. It has been shown in Christensen, Kragic, and Sandberg (2001) that efficient and robust results can be achieved using the chromatic color space. Chromatic colors, known as “pure” colors without brightness, are obtained by normalizing each of the components by the total sum. Color is represented by r and g components since the blue component is both the noisiest channel and it is redundant after the normalization.

Motion. Motion detection is based on computation of the temporal derivative and the image is segmented into regions of motions and regions of inactivity. This is estimated using image differencing

$$M[(u, v), k] = \mathcal{H}[|I[(u, v), k] - I[(u, v), k-1]| - \Gamma] \tag{8}$$

where Γ is a fixed threshold and $\mathcal{H}$ is defined as

$$\mathcal{H}(x) = \begin{cases} 0 & : \ x \leq 0 \\ x & : \ x > 0 \end{cases}. \tag{9}$$

Intensity Variation. In each frame, the following is estimated for all $m \times m$ (details about m are given in Section 4.4.2) regions inside the tracked window

$$\sigma^2 = \frac{1}{m^2} \sum_u \sum_v \left[I(u, v) - \bar{I}(u, v)\right]^2 \tag{10}$$

where $\bar{I}(u, v)$ is the mean intensity value estimated for the window. For example, for a mainly uniform region, low variation is expected during tracking. On the other hand, if the region is rich in texture, large variation is expected. The level of texture is evaluated as proposed in Shi and Tomasi (1994).

4.3. Weighting

In eq. (5) it is defined that the outputs from individual cues should be weighted by w_i. Consequently, the reliability of a cue should be estimated and its weight determined based on its ability to track the target. The reliability can be either (i) determined a priori and kept constant during tracking, or (ii) estimated during tracking based on the cue's success in estimating the final result or based on how much it is in agreement with other cues. In our previous work, the following methods were evaluated (Kragic 2001):

1. **Uniform weights**. Outputs of all cues are weighted equally, $w_i = 1/n$, where n is the number of cues.

2. **Texture-based weighting**. Weights are preset and depend on the spatial content of the region. For a highly textured region, we use: color (0.25), image differencing (0.3), correlation (0.25), intensity variation (0.2). For uniform regions, the weights are: color (0.45), image differencing (0.2), correlation (0.15), intensity variation (0.2). The weights were determined experimentally.

3. **One-step distance weighting**. The weighting factor, w_i, of a cue, c_i, at time step k depends on the distance from the predicted image position, $\hat{\mathbf{z}}_{k|k-1}$. Initially, the distance is estimated as

$$d_i = ||\mathbf{z}_k^i - \hat{\mathbf{z}}_{k|k-1}|| \tag{11}$$

and errors are estimated as

$$e_i = \frac{d_i}{\sum_{i=1}^n d_i}. \tag{12}$$

Weights are then inversely proportional to the error with $\sum_{i=1}^n w_i = 1$.

4. **History-based distance weighting**. The weighting factor of a cue depends on its overall performance during the tracking sequence. The performance is evaluated by observing how many times the cue was in an agreement with the rest of the cues. The following strategy is used.

   (a) For each cue, c_i, examine if $||\mathbf{z}_k^i - \mathbf{z}_k^j|| < d_T$ where $i, j = 1, \dots, n$ and $i \neq j$. If this is true, $a_{ij} = 1$, otherwise $a_{ij} = 0$. Here, $a_{ij} = 1$ means that there is an agreement between the outputs of cues i and j at that voting cycle and d_T represents a distance threshold which is set in advance.

   (b) Build the $(n-1)$ value set for each cue, c_i : $\{a_{ij} | j = 1, \dots, n \text{ and } i \neq j\}$ and estimate sum $s_i = \sum_{j=1}^n a_{ij}$.

   (c) The accumulated values during N tracking cycles, $S_i = \sum_{k=1}^N s_i^k$, indicate how many times a cue, c_i, was in agreement with other cues. Weights are then simply proportional to this value:

$$w_i = \frac{S_i}{\sum_{i=1}^n S_i} \quad \text{with} \quad \sum_{i=1}^n w_i = 1. \tag{13}$$

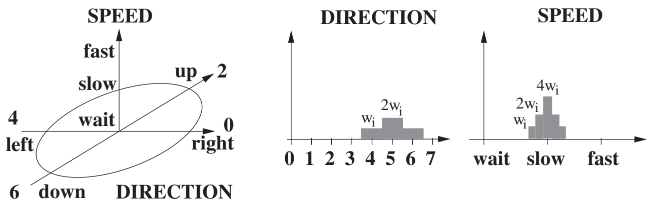


Fig. 4. A schematic overview of the *action fusion* approach; the desired direction is (*down and left*) with a (*slow*) speed. The use of bins represents a *neighborhood voting scheme* which ensures that slight differences between different cues do not result in an unstable classification.

4.4. Implementation

We have investigated two approaches where voting is used for (i) *response fusion*, and (ii) *action fusion*. The first approach makes use of "raw" responses from the employed visual cues in the image, which also represents the action space, **A**. Here, the response is represented either by a binary function (yes/no) answer, or in the interval [0,1] (these values are scaled between [0,255] to allow visual monitoring). The second approach uses a different action space represented by a *direction* and a *speed* (see Figure 4). Compared to the first approach, where the position of the tracked region is estimated, this approach can be viewed as estimating its velocity. Again, each cue votes for different actions from the action space, **A**, which is now the velocity space.

4.4.1. Initialization

According to Figure 3, a tracking sequence should be initiated by *detecting* the target object. If a recognition module is not available, another strategy can be used. In Toyama and Hager (1996) it is proposed that *selectors* should be employed which are defined as heuristics that selects regions possibly occupied by the target. When the system does not have definite state information about the target, it should actively search the state space to find it. Based on this idea, color and image differences (or foreground motion) may be used to detect the target in the first image. Again, if a recognition module is not available, these two cues may also be used in cases where the target either (i) has left the field of view, or (ii) was occluded for a few frames. Our system searches the whole image for the target and once the target enters the image, tracking is regained.

4.4.2. Response Fusion Approach

After the target is located, a template is initialized which is used by the correlation cue. In each frame, a color image of the scene is acquired. Inside the window of attention the response of each cue, denoted O_i, is evaluated (see Figure 5). Here, $\mathbf{x}$ represents a position:

Color. During tracking, all pixels whose color falls in the pre-trained color cluster are given a value between [0,255] depending on the distance from the center of the cluster

$$\begin{aligned} &0 \le O_{color}(\mathbf{x}, k) \le 255 \quad \text{with} \\ &\mathbf{x} \in [\hat{\mathbf{z}}_{k|k-1} - 0.5\mathbf{x}_w,\ \hat{\mathbf{z}}_{k|k-1} + 0.5\mathbf{x}_w] \end{aligned} \tag{14}$$

where $\mathbf{x}_w$ is the size of the window of attention.

Motion. Using eqs. (8) and (9) with $\Gamma = 10$, the image is segmented into regions of motion and inactivity:

$$\begin{aligned} &0 \le O_{motion}(\mathbf{x}, k) \le 255 - \Gamma \quad \text{with} \\ &\mathbf{x} \in [\hat{\mathbf{z}}_{k|k-1} - 0.5\mathbf{x}_w,\ \hat{\mathbf{z}}_{k|k-1} + 0.5\mathbf{x}_w]. \end{aligned} \tag{15}$$

Correlation. Since the correlation cue produces a single position estimate, the output is given by

$$\begin{aligned} &O_{SSD}(\mathbf{x}, k) = 255 e^{(-\frac{\bar{\mathbf{x}}^2}{2\sigma^2})} \quad \text{with} \quad \sigma = 5 \\ &\mathbf{x} \in [\mathbf{z}_{SSD} - 0.5\mathbf{x}_w,\ \mathbf{z}_{SSD} + 0.5\mathbf{x}_w], \\ &\bar{\mathbf{x}} \in [-0.5\mathbf{x}_w,\ 0.5\mathbf{x}_w] \end{aligned} \tag{16}$$

where the maximum of the Gaussian is centered at the peak of the SSD surface. The size of the search area depends on the estimated velocity of the region.

Intensity variation. The response of this cue is estimated according to eq. (10). If a low variation is expected, all pixels inside an $m \times m$ region are given values (255-σ). If a large variation is expected, pixels are assigned a σ value directly. The size $m \times m$ of the subregions which are assigned the same value depends on the size of the window of attention with $n = 0.2\mathbf{x}_w$. Hence, for a 30 × 30 pixel window of attention, $m = 6$. The result is presented as follows:

$$\begin{aligned} &0 \le O_{var}(\mathbf{x}, k) \le 255 \quad \text{with} \\ &\mathbf{x} \in [\hat{\mathbf{z}}_{k|k-1} - 0.5\mathbf{x}_w,\ \hat{\mathbf{z}}_{k|k-1} + 0.5\mathbf{x}_w]. \end{aligned} \tag{17}$$

Response Fusion. The estimated responses are integrated using eq. (5):

$$\delta(\mathbf{x}, k) = \sum_i^n w_i O_i(\mathbf{x}, k). \tag{18}$$

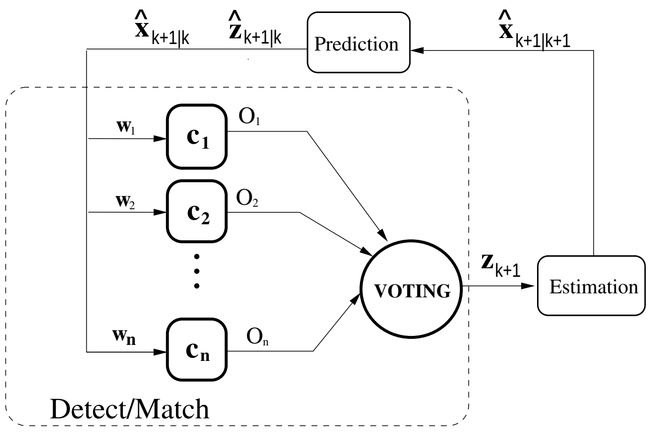


Fig. 5. A schematic overview of the *response fusion* approach.

However, eq. (6) cannot be directly used for selection, as there might be several pixels with same number of votes. Therefore, this equation is slightly modified:

$$\delta'(\mathbf{x}, k) = \begin{cases} 1: & \text{if } \delta(\mathbf{x}, k) \text{ is } \operatorname{argmax}\{\delta(\mathbf{x}', k) | \mathbf{x}' \\ & \in [\hat{\mathbf{z}}_{k|k-1} - 0.5\mathbf{x}_w, \hat{\mathbf{z}}_{k|k-1} + 0.5\mathbf{x}_w]\} \\ 0: & \text{otherwise} \end{cases}. \tag{19}$$

Finally, the new measurement $\mathbf{z}_k$ is given by the mean value (first moment) of $\delta'(\mathbf{x}, k)$, i.e., $\mathbf{z}_k = \bar{\delta}'(\mathbf{x}, k)$.

4.4.3. Action Fusion Approach

Here, the action space is defined by a direction d and speed s (see Figure 4). Both the direction and the speed are represented by histograms of discrete values where the direction is represented by eight values (see Figure 6):

$$\begin{aligned} &\text{LD}\begin{bmatrix}-1\\1\end{bmatrix}, \text{L}\begin{bmatrix}-1\\0\end{bmatrix}, \text{LU}\begin{bmatrix}-1\\-1\end{bmatrix}, \text{U}\begin{bmatrix}0\\-1\end{bmatrix}, \\ &\text{RU}\begin{bmatrix}1\\-1\end{bmatrix}, \text{R}\begin{bmatrix}1\\0\end{bmatrix}, \text{RD}\begin{bmatrix}1\\1\end{bmatrix}, \text{D}\begin{bmatrix}0\\1\end{bmatrix}, \end{aligned} \tag{20}$$

with L, R, D, and U denoting left, right, down, and up, respectively. Speed is represented by 20 values with 0.5 pixel interval, which means that the maximum allowed displacement between successive frames is 10 pixels (this is easily made adaptive based on the estimated velocity). There are two reasons for choosing just eight values for the direction: (i) if the update rate is high or the inter-frame motion is slow, this approach will still give a reasonable accuracy and hence, a smooth performance; (ii) by keeping the voting space rather small there is a higher chance that the cues will vote for the same action. Accordingly, each cue will vote for a desired direction and a desired speed. As presented in Figure 4, a

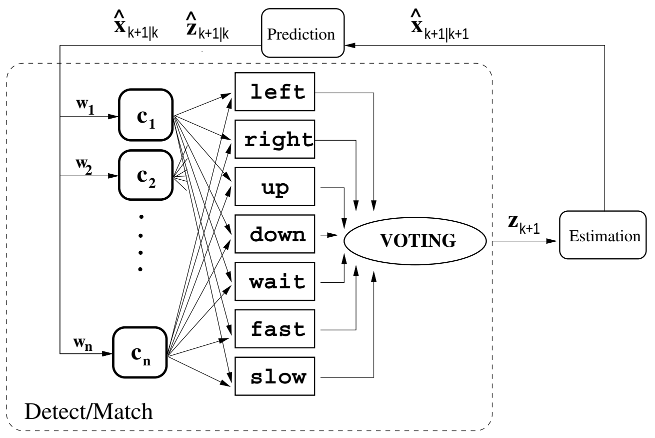


Fig. 6. A schematic overview of the *action fusion* approach.

neighborhood voting scheme is used to ensure that slight differences between different cues do not result in an unstable classification. Equation (3) is modified so that

$$\mathbf{H} = \begin{bmatrix} 0\,0\,1\,0 \\ 0\,0\,0\,1 \end{bmatrix} \text{ and } \mathbf{W} = \begin{bmatrix} \alpha\Delta T & 0 & \beta & 0 \\ 0 & \alpha\Delta T & 0 & \beta \end{bmatrix}^{\mathrm{T}}. \tag{21}$$

In each frame, the following is estimated for each cue:

Color. The response of the color cue is first estimated according to eq. (14) followed by

$$\mathbf{a}_{color}(k) = \frac{\sum_{\mathbf{x}} O_{color}(\mathbf{x}, k)\mathbf{x}(k)}{\sum_{\mathbf{x}} \mathbf{x}(k)} - \hat{\mathbf{p}}_{k|k-1} \quad \text{with} \quad \mathbf{x} \in [\hat{\mathbf{p}}_{k|k-1} - 0.5\mathbf{x}_w,\ \hat{\mathbf{p}}_{k|k-1} + 0.5\mathbf{x}_w] \tag{22}$$

where $\mathbf{a}_{color}(k)$ represents the desired action, and $\hat{\mathbf{p}}_{k|k-1}$ is the predicted position of the tracked region. The same approach is used to obtain $\mathbf{a}_{motion}(k)$ and $\mathbf{a}_{var}(k)$.

Correlation. The minimum of the SSD surface is used as

$$\mathbf{a}_{SSD}(k) = \underset{\mathbf{x}}{\operatorname{argmin}}(SSD(\mathbf{x}, k)) - \hat{\mathbf{p}}_{k|k-1} \tag{23}$$

where the size of the search area depends on the estimated velocity of the tracked region.

Action Fusion. After the desired action, $\mathbf{a}_i(k)$, for a cue is estimated, the cue produces the votes as follows:

$$\begin{aligned} &\text{direction } d_i = \mathcal{P}(\operatorname{sgn}(\mathbf{a}_i)), \\ &\text{speed } s_i = ||\mathbf{a}_i||. \end{aligned} \tag{24}$$

Here, $\mathcal{P} : \mathbf{x} \rightarrow \{0, 1, \ldots, 7\}$ is a scalar function that maps the two-dimensional direction vectors (see eq. (20)) to one-dimensional values representing the bins of the direction histogram. Now, the estimated direction, d_i, and the speed, s_i, of a cue, c_i, with weight, w_i, are used to update the direction and speed of the histograms according to Figure 4 and eq. (5). The new measurement is then estimated by multiplying the actions from each histogram which received the maximum number of votes according to eq. (6)

$$\mathbf{z}_k = \mathcal{S}(\underset{d}{\operatorname{argmax}}\, HD(d))\ \underset{s}{\operatorname{argmax}}\, HS(s) \tag{25}$$

where $\mathcal{S} : x \rightarrow \{\left[\begin{smallmatrix} -1 \\ 0 \end{smallmatrix}\right], \ldots, \left[\begin{smallmatrix} -1 \\ 1 \end{smallmatrix}\right]\}$. The update and prediction steps are then performed using eqs. (21) and (3). The reason for choosing this particular representation instead of simply using a weighted sum of first moments of the responses of all cues is, as has been pointed out in Rosenblatt and Thorpe (1995), that arbitration via vector addition can result in commands which are not satisfactory to any of the contributing cues.

4.5. Examples

The proposed methods have been investigated in detail in Kragic (2001). Here, we present one of the experiments where we evaluated the performance of the voting approaches as well as the performance of individual cues with respect to three sensor–object configurations typically used in visual servoing systems: (i) static sensor/moving object ("stand-alone camera system"); (ii) moving sensor/static object ("eye-in-hand camera" servoing toward a static object); (iii) moving sensor/moving object (camera system on a mobile platform or eye-in-hand camera servoing toward a moving object). The results are discussed through *accuracy* and *reliability* measures. The accuracy is expressed using an error measure which is a distance between the ground truth (chosen manually using a reference point on the object) and the currently estimated position of the reference point. The results are summarized through the mean square error and standard deviation in pixels. The measure of the reliability is on a yes/no basis depending on whether a cue (or the fused system) successfully tracks the target during a single experiment. The tracking is successful if the object is kept inside the window of attention during the entire test sequence.

The two fusion approaches, as well as the individual cues, have been tested with respect to the ability to cope with occlusions of the target and to regain tracking after the target has left the field of view for a number of frames. The results are presented for correlation, color and image differences since the intensity variation cue cannot be used alone for tracking.

Accuracy (Table 1). As can be seen from the table, the best accuracy is achieved using the *response fusion* approach. Although the *mse* is similar for the *action fusion* approach in cases of *static sensor/moving object* and *moving sensor/static object* configurations, *std* is higher. The reason for this is the choice of the underlying voting space. For example, if the color cue shows a stable performance for a number of frames, its weight will be high compared to the other cues (or it might have been set to a high value from the beginning). In some

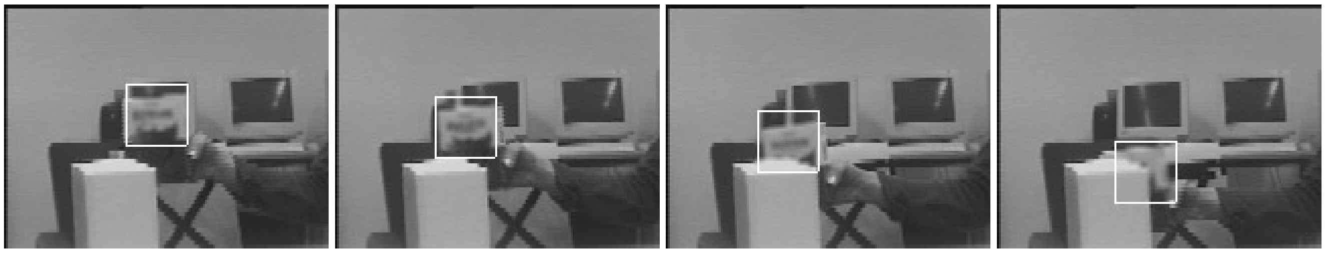


Fig. 7. Example images from a raisin package tracking.

Table 1. Qualitative Results for Various Sensor–Object Configurations (in Pixels)

	Static Sensor/ Moving Object		Moving Sensor/ Static Object		Moving Sensor/ Moving Object	
	mse	std	mse	std	mse	std
RF	7	7	4	3	9	10
AF	7	9	4	10	13	25
Color	15	16	10	6	10	14
Diff	23	26	failed	failed	failed	failed
SSD	25	27	12	13	17	21

cases, as in the case of a box of raisins presented in Figure 7, two colors are used at the same time. When an occlusion occurs, the position of the center of the mass of the color blob will change fast (and sometimes in different directions) which results in abrupt changes in both direction and speed. The other method, *response fusion*, on the other hand, does not suffer from this which results in a lower standard deviation value.

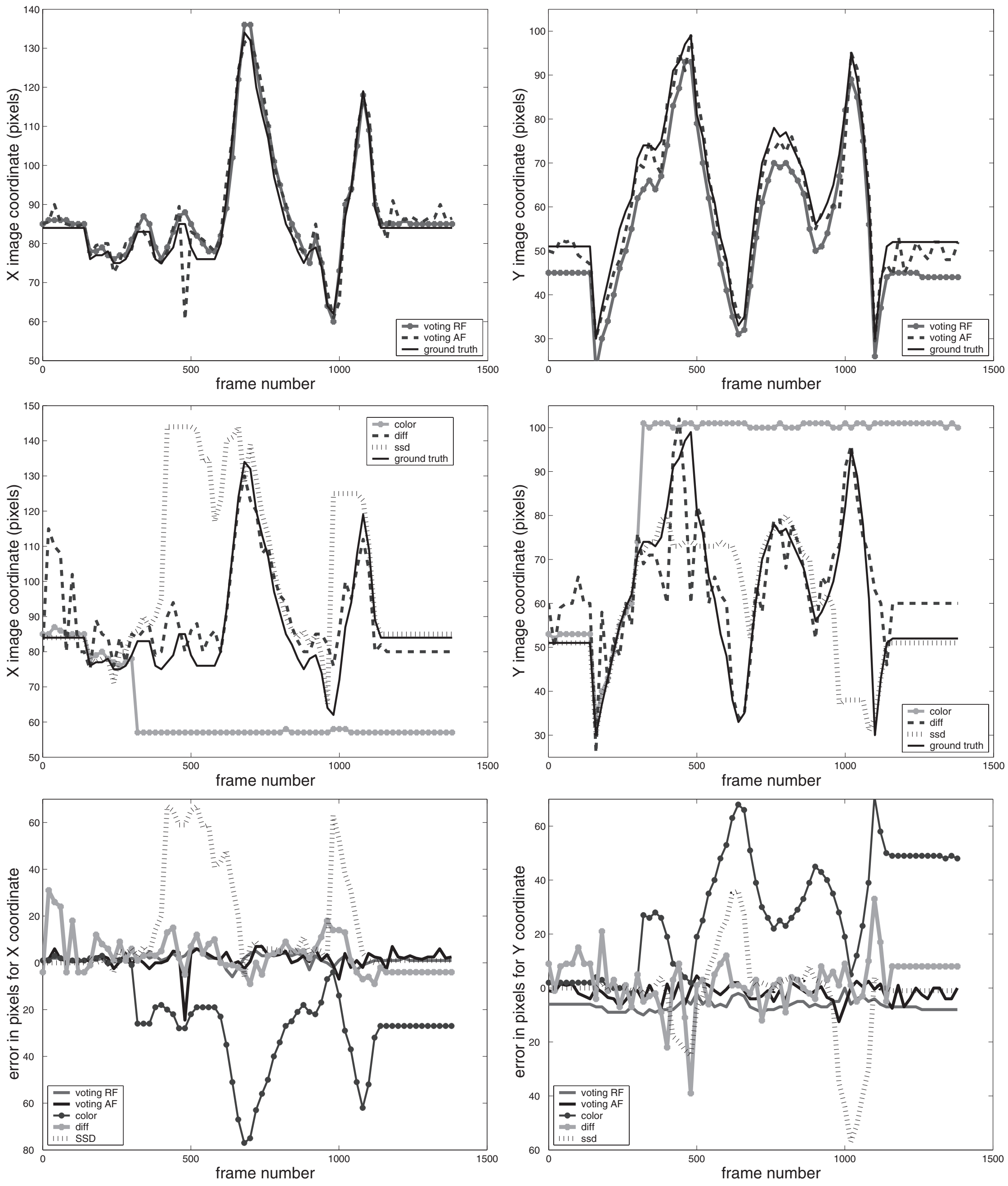
The comparison of the performance of the fusion approaches and the performance of the individual cues shows the necessity for fusion. Image differences alone cannot be used in cases of *moving sensor/static object* and *moving sensor/moving object* configurations since there is no ability to differ between the object and the background. As mentioned earlier, during most of the sequences the target undergoes 3D motion which results in scale changes and rotations not modeled by SSD. It is obvious that these factors will affect this cue significantly, resulting in a large error as demonstrated in the table. This problem may be solved by using a better model (affine, quadratic; see Hager and Toyama 1996). It can also be seen that the color cue performed best of the individual cues. In the case of *moving sensor/static object*, after the tracking is initialized the color cue "sticks" to the object during the sequence and, since the background varies a little, the best accuracy is achieved compared to other configurations. During the other two configurations, the background will change containing also the same color as the target. This distracts the color tracker, resulting in increased error. The error is larger in the case of *static sensor/moving object* compared to *moving sensor/moving object* since in the test sequences the background included the target's color more often.

Reliability (Table 2). The reliability is expressed through a number of successful runs where the accuracy is obtained using the *texture-based weighting*. In Table 2, the obtained reliability results are ranked showing that color performs most reliably compared to other individual cues. In certain cases, especially when the influence of the background is not significant, this cue will perform satisfactorily. However, it will easily become distracted if the background takes a large portion of the window of attention and includes the target's color. Image differencing will depend on the size of the moving target with respect to the size of the window of attention and variations in lighting. In structured environments, however, this cue may perform well and may be considered in cases of a single moving target where the size of the target is small compared to the size of the image (or window of attention).

Figure 8 shows tracking accuracy for the proposed fusion approaches and for each of the cues individually. The plots and the table show the deviation from the ground truth value (in pixels). The target is a package of raisins. During this sequence, a number of occlusions occur (as demonstrated in the images), but the plots demonstrate a stable performance of the fusion approaches during the whole sequence. The color cue is, however, "fooled" by the box which is the same color as the target. The plots demonstrate how this cue fails around frame 300 and never regains tracking after that. These two

Table 2. Success Rate for Individual Cues and Fusion Approaches

	Number of Successes	Number of Failures	%
RF voting	27	3	90
AF voting	22	8	73.3
Color	18	12	60
SSD	12	18	40
Diff.	7	23	23.3



	RF voting		AF voting		Color		Diff		SSD	
	x	*y*	*x*	*y*	*x*	*y*	*x*	*y*	*x*	*y*
mean	1.5	−6.4	1.5	−1.7	−24	28	3.2	2.5	14.9	-2.4
std	2	1.9	4	3	19.4	20.8	8.3	9.5	22	16

Fig. 8. The comparison between the ground truth, voting approaches and individual cues in case of occlusions (first two rows). Third row shows error plots for all approaches. The table represents the mean and standard deviation from the ground truth (in pixels). Some of the images from the tracking sequence are shown in Figure 7.

examples clearly demonstrate that tracking by fusion is more superior than any of the individual cues.

5. Model-Based Visual Servoing

Although suitable for a number of tasks, the previous approach lacks the ability to estimate position and orientation (pose) of the object. In terms of manipulation, it is usually required to accurately estimate the pose of the object, for example, to allow the alignment of the robot arm with the object or to generate a feasible pose for grasping. Using prior knowledge about the object, a special representation can further increase the robustness of the tracking system. Along with commonly used CAD models (wire-frame models), view-based and appearance-based representations may be employed (Bretzner 1999).

A recent study of human visually guided grasps in situations similar to that typically used in visual servoing control has shown that the human visuomotor system takes into account the three-dimensional geometric features rather than the two-dimensional projected image of the target objects to plan and control the required movements (Hu, Eagleson, and Goodale 1999). These computations are more complex than those typically carried out in visual servoing systems and permit humans to operate in a large range of environments.

We have decided to integrate appearance-based and geometrical models in our model-based tracking system. Many similar systems use manual pose initialization where the correspondence between the model and object features is given by the user (Giordana et al. 2000; Drummond and Cipolla 2000). Although there are systems where this step is performed automatically, the approaches are time-consuming and not appealing for real-time applications (Gengenbach et al. 1996; Lowe 1985). One additional problem, in our case, is that the objects to be manipulated by the robot are highly textured (see Figure 9) and therefore not suited for matching approaches based on, for example, line features (Koller, Daniilidis, and Nagel 1993; Vincze, Ayromlou, and Kubinger 1999; Wunsch and Hirzinger 1997).

After the object has been recognized and its position in the image is known, an appearance-based method is employed to estimate its initial pose. The method we have implemented has been initially proposed in Nayar, Nene, and Murase (1996) where just three pose parameters have been estimated and used to move a robotic arm to a pre-defined pose with respect to the object. Compared to our approach, where the pose is expressed relative to the camera coordinate system, they express the pose relative to the current arm configuration, making the approach unsuitable for robots with a different number of degrees of freedom.

Compared to the system proposed in Wunsch, Winkler, and Hirzinger (1997), where the network has been entirely trained on simulated images, we use real images for training

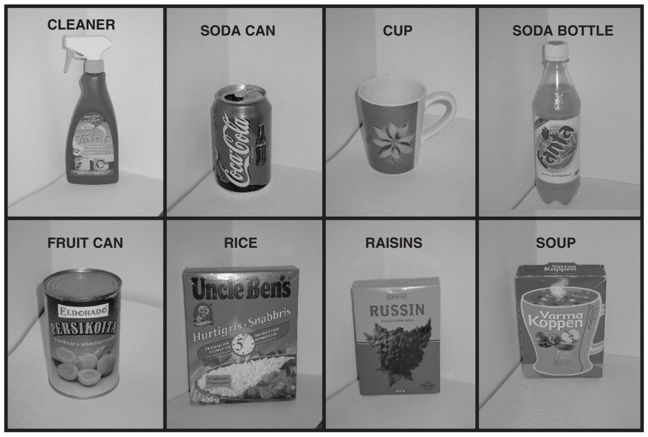


Fig. 9. Some of the objects we want the robot to manipulate.

where no particular background was considered. As pointed out in Wunsch, Winkler, and Hirzinger (1997), the illumination conditions (as well as the background) strongly affect the performance of their system and these cannot be easily obtained with simulated images. In addition, the idea of projecting just the wire-frame model to obtain training images cannot be employed in our case due to the texture of the objects. The system proposed in Vincze, Ayromlou, and Kubinger (1999) also employs a feature-based approach where lines, corners and circles are used to provide the initial pose estimate. However, this initialization approach is not applicable in our case since, due to the geometry and textural properties, these features are not easy to extract with high certainty.

Our model-based tracking system is presented in Figure 10. During the *initialization* step, the initial pose of the object relative to the camera coordinate system is estimated. The main loop starts with a *prediction* step where the state of the object is predicted using the current pose estimate and a motion model. The visible parts of the object are then projected into the image (*projection and rendering* step). After the *detection* step, where a number of features are extracted in the vicinity of the projected ones, these new features are *matched* to the projected ones and used to estimate the new pose of the object. Finally, the calculated pose is input to the *update* step. The system has the ability to cope with partial occlusions of the object, and to successfully track the object even in the case of significant rotational motion.

5.1. Prediction and Update

The system state vector consists of three parameters describing translation of the target, another three for orientation and an additional six for the velocities:

$$\mathbf{x} = \left[X, Y, Z, \phi, \psi, \gamma, \dot{X}, \dot{Y}, \dot{Z}, \dot{\phi}, \dot{\psi}, \dot{\gamma}\right]. \qquad (26)$$

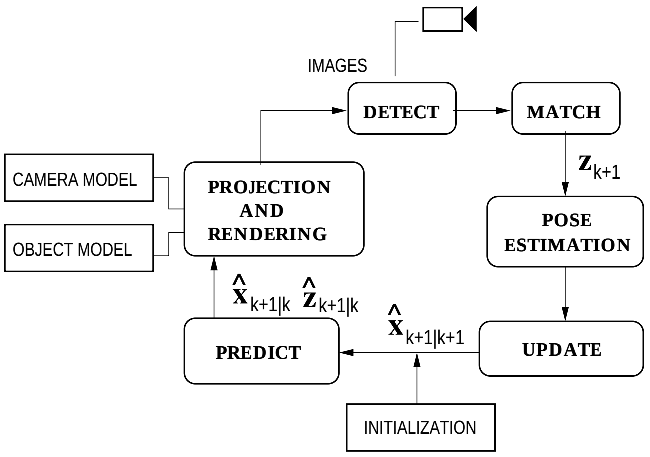


Fig. 10. Block diagram of the model-based tracking system.

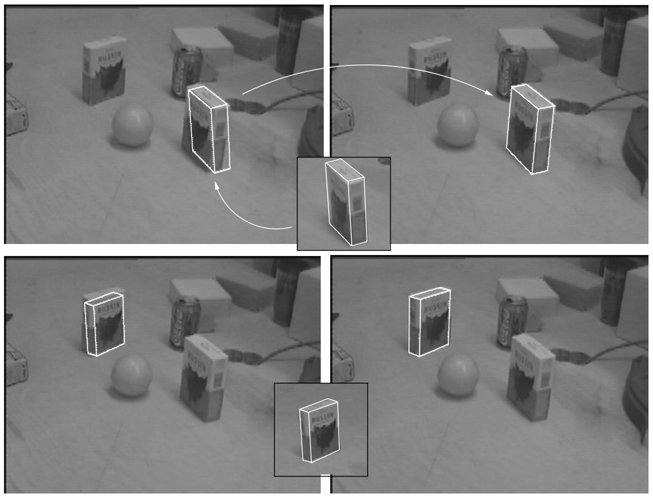


Fig. 11. On the left is the initial pose estimated using the PCA approach. On the right is the pose obtained by the local refinement method.

Here, ϕ, ψ, and γ represent roll, pitch, and yaw angles, respectively (Craig 1989). The following piecewise constant white acceleration model is considered (Bar-Shalom and Li 1993)

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{F}\mathbf{x}_k + \mathbf{G}\mathbf{v}_k \\ \mathbf{z}_k &= \mathbf{H}\mathbf{x}_k + \mathbf{w}_k \end{aligned} \tag{27}$$

where $\mathbf{v}_k$ is a zero-mean white acceleration sequence, $\mathbf{w}_k$ is the measurement noise and

$$\mathbf{F} = \begin{bmatrix} \mathbf{I}_{6\times6} & \Delta T\mathbf{I}_{6\times6} \\ \mathbf{0} & \mathbf{I}_{6\times6} \end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix} \frac{\Delta T^2}{2}\mathbf{I}_{6\times6} \\ \Delta T\mathbf{I}_{6\times6} \end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix} \mathbf{I}_{6\times6} & | & \mathbf{0} \end{bmatrix}. \tag{28}$$

For prediction and update, the α–β filter is used:

$$\begin{aligned} \hat{\mathbf{x}}_{k+1|k} &= \mathbf{F}_k\hat{\mathbf{x}}_k \\ \hat{\mathbf{z}}_{k+1|k} &= \mathbf{H}\hat{\mathbf{x}}_{k+1|k} \\ \hat{\mathbf{x}}_{k+1|k+1} &= \hat{\mathbf{x}}_{k+1|k} + \mathbf{W}[\mathbf{z}_{k+1} - \hat{\mathbf{z}}_{k+1|k}]. \end{aligned} \tag{29}$$

Here, the pose of the target is used as measurement rather than image features, as commonly used in the literature; see, for example, Dickmanns and Graefe (1988) and Gengenbach et al. (1996). An approach similar to that presented here is taken in Wunsch and Hirzinger (1997). This approach simplifies the structure of the filter which facilitates a computationally more efficient implementation. In particular, the dimension of the matrix $\mathbf{H}$ does not depend on the number of matched features in each frame but it remains constant during the tracking sequence.

5.2. Initial Pose Estimation

The initialization step uses the ideas proposed in Nayar, Nene, and Murase (1996). During training, each image is projected as a point to the eigenspace and the corresponding pose of the object is stored with each point. For each object, we have used 96 training images (eight rotations for each angle on four different depths). One of the reasons for choosing this low number of training images is the workspace of the PUMA560 robot used. Namely, the workspace of the robot is quite limited and for our applications this discretization was satisfactory. To enhance the robustness with respect to variations in intensity, all images are normalized. At this stage, the size of the training samples is 100×100 pixel color images. The training procedure takes about 3 min on a Pentium III 550 running Linux.

Given an input image, it is first projected to the eigenspace. The corresponding parameters are found as the closest point on the pose manifold. Now, the wire-frame model of the object can be easily overlaid on the image. Since a low number of images is used in the training process, pose parameters will not accurately correspond to the input image. Therefore, a local refinement method is used for the final fitting (see Figure 11). The details are given in the next section.

During the training step, it is assumed that the object is approximately centered in the image. During task execution, the object can occupy an arbitrary part of the image. Since the recognition step delivers the image position of the object, it is easy to estimate the offset of the object from the image center and to compensate for it. In this way, the pose of the object relative to the camera frame can also be arbitrary.

An example of the pose initialization is presented in Figure 12. Here, the pose of the object in the training image (far left) was $X = -69.3$, $Y = 97.0$, $Z = 838.9$, $\phi = 21.0$, $\psi = 8.3$ and $\gamma = -3.3$. After the fitting step, the pose was $X = 55.9$, $Y = 97.3$, $Z = 899.0$, $\phi = 6.3$, $\psi = 14.0$ and $\gamma = 1.7$ (far right), showing the ability of the system to cope with significant differences in pose parameters.

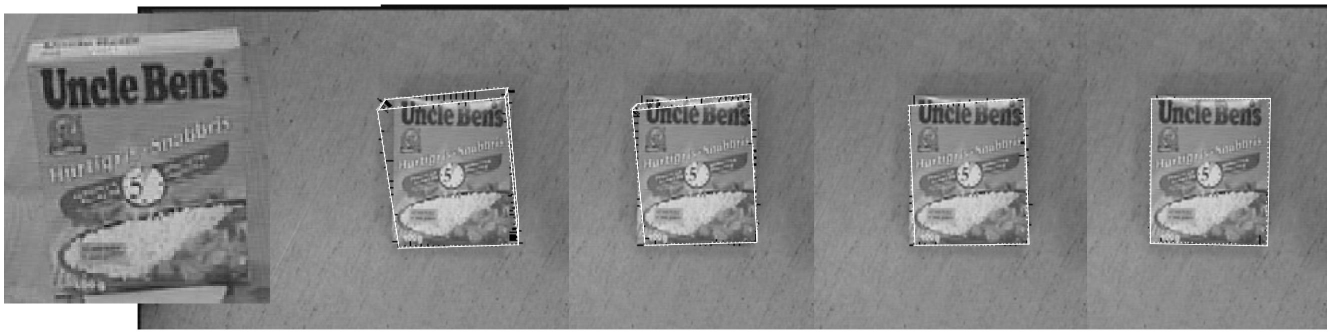


Fig. 12. Training image used to estimate the initial pose (far left) followed by the intermediate images of the fitting step.

5.3. Detection and Matching

When the estimate of the object's pose is available, the visibility of each edge feature is determined and internal camera parameters are used to project the model of the object onto the image plane. For each visible edge, a number of image points are generated along the edge. So-called tracking nodes are assigned at regular intervals in image coordinates along the edge direction. The discretization is performed using the Bresenham algorithm (Foley et al. 1990). After this, a search is performed for the maximum discontinuity (nearby edge) in the intensity gradient along the normal direction to the edge. The edge normal is approximated with four directions: $\{-45, 0, 45, 90\}$ degrees.

In each point $\mathbf{p}_i$ along a visible edge, the perpendicular distance $d_i^{\perp}$ to the nearby edge is determined using a one-dimensional search. The search region is denoted by $\{S_i^j, j \in [-s, s]\}$. The search starts at the projected model point $\mathbf{p}_i$ and the traversal continues simultaneously in opposite search directions until the first local maximum is found. The size of the search region s is adaptive and inversely depends on the distance of the objects from the camera.

After the normal displacements are available, the method proposed in Drummond and Cipolla (2000) is used. Lie group and Lie algebra formalism are used as the basis for representing the motion of a rigid body and pose estimation. The method is also related to the work presented in Harris (1992) and Thompson et al. (2001). Implementation details can be found in Kragic (2001).

5.4. Servoing Based on Projected Models

Once the pose of the object is available, any of the servoing approaches can be employed. In this section, we show how a model-based tracking system can be used for both image- and position-based visual servoing.

5.4.1. Position-Based Servoing

Let us assume the following scenario. The task is to align the end-effector with respect to an object and maintain the constant pose when/if the object starts to move. It is assumed here that a model-based tracking algorithm is available and one stand-alone camera (not attached to the robot) is used during the execution of the tasks (see Figure 13).

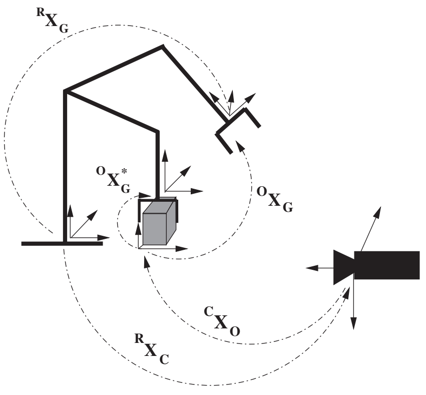


Fig. 13. Relevant coordinate frames and their relationships for the "align-and-track" task.

Here, $^{O}\mathbf{X}_G^*$ represents the desired pose between the object and the end-effector while $^{O}\mathbf{X}_G$ represents the current (or initial) pose between them. To perform the task using the position-based servoing approach, the transformation between the camera and the robot coordinate frames, $^{C}\mathbf{X}_R$, has to be known. The pose of the end-effector with respect to the robot base system, $^{R}\mathbf{X}_G$, is known from the robot kinematics. The model-based visual tracking system estimates the pose of the object relative to the camera coordinate system, $^{C}\mathbf{X}_O$.

Let us assume that the manipulator is controlled in the end-effector frame. According to Figure 13, if $^{O}\mathbf{X}_G = {}^{O}\mathbf{X}_G^*$ then $^{R}\mathbf{X}_G = {}^{R}\mathbf{X}_G^*$. The error function to be minimized may then be defined as the difference between the current and the desired end-effector pose:

$$\begin{aligned} \Delta\, {}^{R}\mathbf{t}_G &= {}^{R}\mathbf{t}_G - {}^{R}\mathbf{t}_G^* \\ \Delta\, {}^{R}\theta_G &= {}^{R}\theta_G - {}^{R}\theta_G^*. \end{aligned} \tag{30}$$

Here, ${}^{R}\mathbf{t}_G$ and ${}^{R}\theta_G$ are known from the forward kinematics equations and ${}^{R}\mathbf{t}^*_G$ and ${}^{R}\theta^*_G$ have to be estimated. The homogeneous transformation between the robot and desired end-effector frame is given by

$$ {}^{R}\mathbf{X}^*_G = {}^{R}\mathbf{X}_C \, {}^{C}\mathbf{X}_O \, {}^{O}\mathbf{X}^*_G. \tag{31}$$

The pose between the camera and the robot is estimated offline and the pose of the object relative to the camera frame is estimated using the model-based tracking system presented in Section 3. Expanding the transformations in eq. (31) we obtain

$$ {}^{R}\mathbf{t}^*_G = {}^{R}\mathbf{R}_C \, {}^{C}\hat{\mathbf{R}}_O \, {}^{O}\mathbf{t}^*_G + {}^{R}\mathbf{R}_C \, {}^{C}\hat{\mathbf{t}}_O + {}^{R}\mathbf{t}_C \tag{32}$$

where ${}^{C}\hat{\mathbf{R}}_O$ and ${}^{C}\hat{\mathbf{t}}_O$ represent predicted values obtained from the tracking algorithm. A similar expression can be obtained for the change in rotation by using the addition of angular velocities (see Figure 13; Craig 1989):

$$ {}^{R}\Omega^*_G = {}^{R}\Omega_C + {}^{R}\mathbf{R}_C \, {}^{C}\hat{\Omega}_O + {}^{R}\mathbf{R}_C \, {}^{C}\hat{\mathbf{R}}_O \, {}^{O}\Omega^*_G. \tag{33}$$

Assuming that ${}^{R}\mathbf{R}_C$ and ${}^{C}\hat{\mathbf{R}}_O$ are slowly-varying functions of time, integration of ${}^{R}\Omega^*_G$ gives (Wilson, Williams Hulls, and Bell 1996):

$$ {}^{R}\theta^*_G \approx {}^{R}\theta_C + {}^{R}\mathbf{R}_C \, {}^{C}\hat{\theta}_O + {}^{R}\mathbf{R}_C \, {}^{C}\hat{\mathbf{R}}_O \, {}^{O}\theta^*_G. \tag{34}$$

Substituting eqs. (32) and (34) into eq. (30) yields

$$\begin{aligned} \Delta\, {}^{R}\mathbf{t}_G &= {}^{R}\mathbf{t}_G - {}^{R}\mathbf{t}_C - {}^{R}\mathbf{R}_C \, {}^{C}\hat{\mathbf{t}}_O - {}^{R}\mathbf{R}_C \, {}^{C}\hat{\mathbf{R}}_O \, {}^{O}\mathbf{t}^*_G \\ \Delta\, {}^{R}\theta_G &\approx {}^{R}\theta_G - {}^{R}\theta_C - {}^{R}\mathbf{R}_C \, {}^{C}\hat{\theta}_O - {}^{R}\mathbf{R}_C \, {}^{C}\hat{\mathbf{R}}_O \, {}^{O}\theta^*_G \end{aligned} \tag{35}$$

which represents the error to be minimized:

$$ \mathbf{e} = \begin{bmatrix} \Delta\, {}^{R}\mathbf{t}_G \\ \Delta\, {}^{R}\theta_G \end{bmatrix}. \tag{36}$$

After the error function is defined, a simple proportional control law is used to drive the error to zero. The velocity screw of the robot is defined as[1]:

$$ \dot{\mathbf{q}} \approx \mathbf{K}\mathbf{e}. \tag{37}$$

Using the estimate of the object's pose and defining the error function in terms of pose, all six degrees of freedom of the robot are controlled as shown in Figure 14.

5.4.2. Image-Based Servoing

In many cases the change in the pose of the object is significant and certain features are likely to become occluded. An example of such motion is presented in Figure 15. A model-based tracking system allow us to use image positions of features (for example, end-points of an object's edges) even if they are not currently visible due to the pose of the object. An image-based approach can therefore successfully be used throughout the whole servoing sequence. In addition, since a larger number of features is available as well as the depth of the object, the feedback from one camera is sufficient for performing the task.

In our example, the error function, $\mathbf{e}(\mathbf{f})$, is defined which is to be regulated to zero. The vector of the final (desired) image point feature positions is denoted by $\mathbf{f}^*$ and the vector of their current positions by $\mathbf{f}^c$. The task consists of moving the robot such that the Euclidian norm of the error vector $(\mathbf{f}^* - \mathbf{f}^c)$ decreases. Hence, we may constrain the image velocity of each point to exponentially reach its goal position with time. The desired behavior is then

$$ \dot{\mathbf{f}} = \mathbf{K}\,\mathbf{e}(\mathbf{f}) = \mathbf{K}\,(\mathbf{f}^* - \mathbf{f}^c) \tag{38}$$

where $\mathbf{K}$ is a diagonal gain matrix that controls the convergence rate of the visual servoing. From Hutchinson, Hager, and Corke (1996), a robot velocity screw may be computed using

$$ {}^{G}\dot{\mathbf{q}} = \mathbf{K}\,\mathbf{J}^{\dagger}(\mathbf{q})\,\mathbf{e}(\mathbf{f}) \tag{39}$$

where $\mathbf{J}^{\dagger}$ is a (pseudo-)inverse of the image Jacobian.

6. Example Tasks

In this section we show additional examples of how the presented visual systems were used for robotic tasks.

6.1. Mobile Robot Grasping

Here, we consider the problem of real manipulation in a realistic environment—a living room. Similarly to the previous example, we assume that a number of pre-defined grasps is given and suitable grasp is generated depending on the current pose of the object. The experiment shows a XR4000 platform with a hand-mounted camera. The task is to approach the dinner table where there are several objects. The robot is instructed to pick up a package of rice having an arbitrary placement. Here, distributed control architecture (Petersson, Austin, and Christensen 2001) is used for integration of the different methods into a fully operational system. To perform the task, object recognition, voting-based two-dimensional tracking and model-based three-dimensional tracking are used. The details of the system implementation are reported in Petersson et al. (2002). The results of a mission with the integrated system are outlined below.

The sequence in Figure 16 shows the robot starting at the far end of the living room, moving towards a point where a good view of the dinner table can be obtained. After the robot is instructed to pick up the rice package, it locates in the scene using the system presented in Section 3.1. After that, the robot

1. It is straightforward to estimate the desired velocity screw in the end-effector coordinate frame.

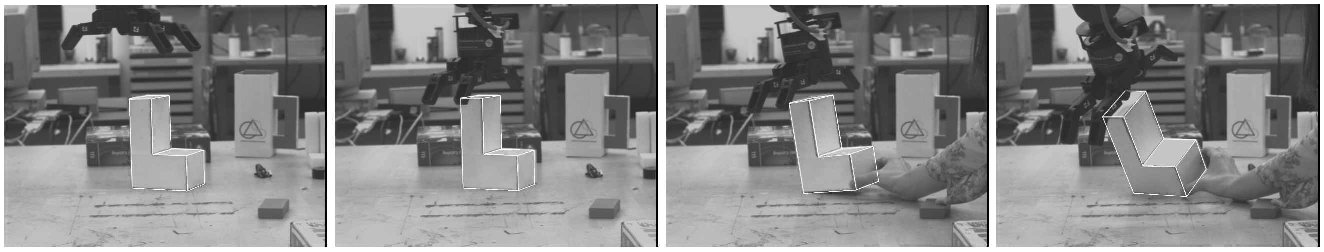


Fig. 14. A sequence of a six-degree-of-freedom visual control. From an arbitrary starting position (upper left), the end-effector is controlled to a pre-defined reference position with respect to the target object (upper right). When the object starts moving, the visual system tracks the pose of the object. The robot is then controlled in a position-based framework to remain a constant pose between the gripper and the object frame.

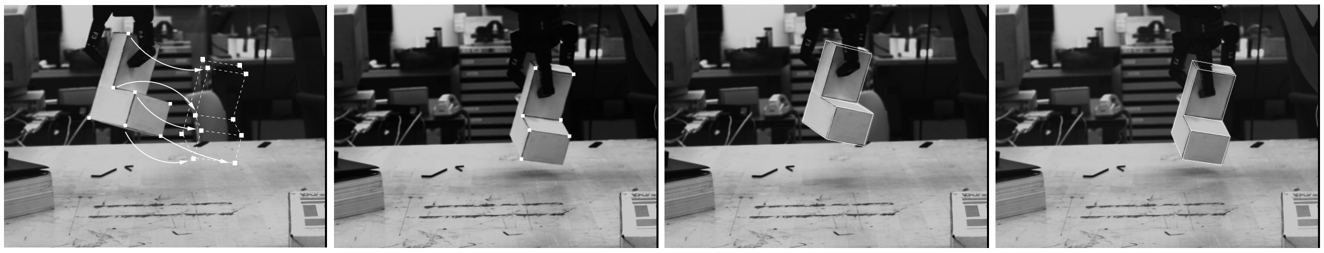


Fig. 15. Start pose, destination pose and two intermediate poses in an image-based visual servo approach.

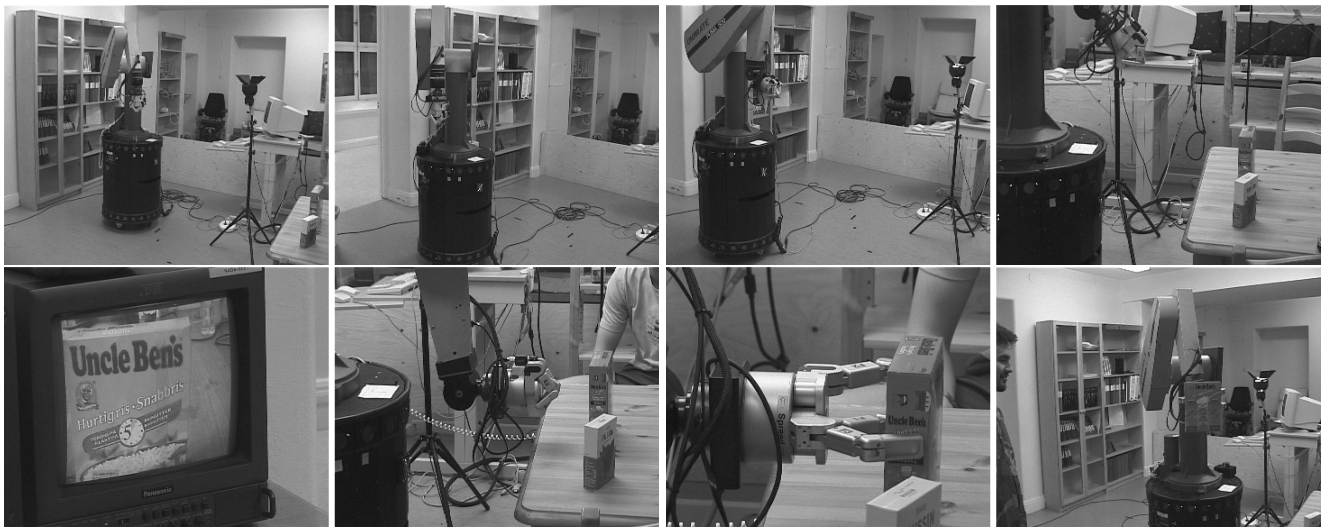


Fig. 16. An experiment where the robot moves up to the table, recognizes the rice box, approaches it, picks it up and hands it over to the user.

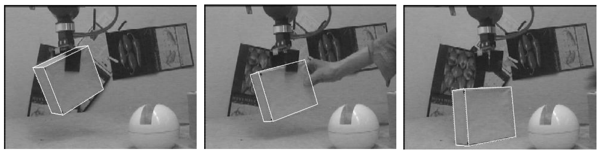


Fig. 17. Removing the object from the gripper; the object is tracked during the whole sequence. The pose of the object is used to estimate the change in object/gripper transformation. The results are presented in Figure 19.

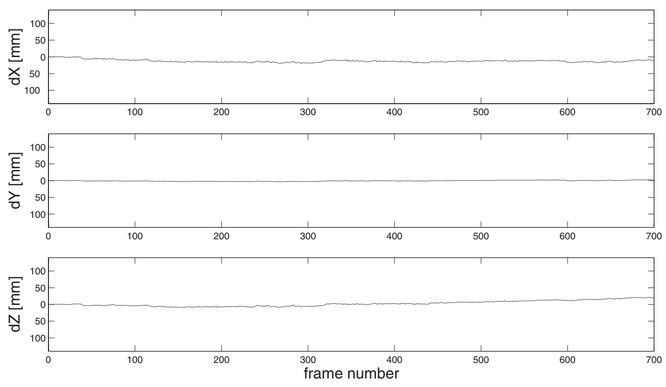


Fig. 18. Change in translation between the gripper and object frames during a stable grasp. The change in dX, dY and dZ is very small and mostly less than 10 mm.

moves closer to the table keeping the rice package centered in the image using the two-dimensional tracking system presented in Section 4. Finally, the gripper is aligned with the object and grasping is performed. The details about the alignment can be found in Tell (2002).

6.2. Model-Based Tracking for Slippage Detection

The ability of a robotic system to generate both a *feasible* and a *stable* grasp adds to its robustness. By a *feasible* grasp, a kinematically feasible grasp is considered; by a *stable* grasp, a grasp for which the object will not twist or slip relative to the end-effector. Here, the latter issue is considered. Aside from picking up the object, a task for the robot may also be to place the object at some desired position in the workspace. If that is the case, after the object is grasped, a *task monitoring* step may be initiated. The basic idea is that, even if the planed grasp was considered stable, when the manipulator starts to move the object may start to slide. If the grasp is stable, the relative transformation between the manipulator (gripper) and the object frames should be constant. Since tactile sensing is not available to us at this stage, our vision system is used to track the object held by the robot and estimate its pose during the placement task. The estimated pose of the object is then used to estimate the change between the object and hand coordinate frames. For a stable grasp, this change should be ideally zero or very small.

In Figures 18 and 19, the variation in the hand/object transformation is presented, showing two cases obtained during a stable grasp and a grasp where the object slid from the hand. Comparing the figures, we see a significant difference in the transformation plots. The most significant change is observed for dX and dZ, which is for dX approximately 100 mm and for dZ almost 150 mm. These changes are caused by the object being removed from the gripper in Figure 19 (see Figure 17).

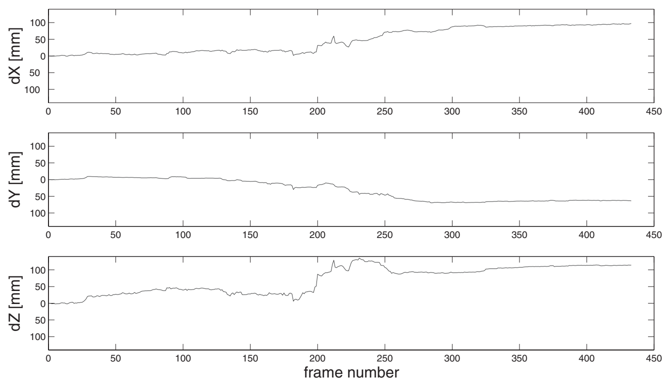


Fig. 19. Change in translation between the gripper and object frames when the object was removed from the gripper. Compared to the plot for a stable grasp (Figure 18), the change is approximately 100 mm for the dX component and 150 mm for the dZ component.

7. Conclusion

Due to the real-world variability, it is not enough to consider only control level robustness. It is equally important to consider how image information that serves as input to the control process can be used so as to achieve robust and efficient control. In visual servoing, providing a robust visual feedback is crucial to the performance of the overall system. It is well known that no single visual feature is robust to variations in geometry, illumination, camera motion and coarse calibration. In addition, no single visual servoing technique can easily be tailored to cope with large-scale variations. For the construction of realistic systems, there is consequently a need to formulate methods for integration of a range of different servoing techniques. The techniques differ in terms of the underlying control space (position/image/hybrid) and type of visual features used to provide the necessary measurements (2D/3D). We argue that each of these techniques must carefully consider how models, multiple cues and fusion can be utilized to provide the necessary robustness.

In this paper, we have taken a first step towards the design of such a system by presenting a range of different vision-based techniques for object detection and tracking. Particular emphasis has been put on the use of computationally tractable methods in terms of real-time operation in realistic settings. The efficiency has in particular been achieved through careful consideration of task level constraints.

Through adaptation of a task oriented framework where the utility of different cues are considered together with methods for integration, it is possible to achieve desired robustness. Task constraints here allow dynamic selection of cues (in terms of weights), integration and associated control methodology so as to allow visual servoing over a wide range of work conditions. Such an approach is, for example, needed in the case of mobile manipulation of objects in service robotics applications. The use of the developed methodology has been demonstrated with a number of examples to illustrate the system's performance.

An open problem in the presented methodology is the optimal balance between different visual servoing approaches as well as the amount and type of information shared between them. An integral part of the future research will consider these issues.

Acknowledgment

This research has been sponsored by the Swedish Foundation for Strategic Research through the Centre for Autonomous Systems. The funding is gratefully acknowledged.

References

Bar-Shalom, Y., and Li, Y. 1993. *Estimation and Tracking: Principles, Techniques and Software*. Artech House, Boston, MA.

Bicchi, A., and Kumar, V. 2000. Robotic grasping and contact: A review. In *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA'00*, pp. 348–353.

Blough, D. M., and Sullivan, G. F. 1994. Voting using predispositions. *IEEE Transactions on Reliability* 43(4):604–616.

Bretzner, L. 1999. *Multi-scale feature tracking and motion estimation*. Ph.D. Thesis, CVAP, NADA, Royal Institute of Technology, KTH.

Christensen, H. I., Kragic, D., and Sandberg, F. 2001. Vision for interaction. In G. Hager, H. I. Christensen, H. Bunke, and R. Klein, eds., *Sensor Based Intelligent Robots*, Lecture Notes in Computer Science Vol. 2238. Springer-Verlag, Berlin, pp. 51–73.

Craig, J. J. 1989. *Introduction to Robotics: Mechanics and Control*. Addison Wesley, Reading, MA.

Dickmanns, E., and Graefe, V. 1988. Dynamic monocular machine vision. *Machine Vision and Applications* 1:223–240.

Drummond, T. W., and Cipolla, R. 2000. Real-time tracking of multiple articulated structures in multiple views. In *Proceedings of the 6th European Conference on Computer Vision, ECCV'00*, Vol. 2, pp. 20–36.

Edelman, S., ed. 1999. *Representation and Recognition in Vision*. MIT Press, Cambridge, MA.

Foley, J. D., van Dam, A., Feiner, S. K., and Hughes, J. F., eds. 1990. *Computer Graphics—Principles and Practice*. Addison-Wesley, Reading, MA.

Gengenbach, V., Nagel, H.-H., Tonko, M., and Schäfer, K. 1996. Automatic dismantling integrating optical flow into a machine-vision controlled robot system. In *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA'96*, Vol. 2, pp. 1320–1325.

Giordana, N., Bouthemy, P., Chaumette, F., and Spindler, F. 2000. Two-dimensional model-based tracking of complex shapes for visual servoing tasks. In M. Vincze and G. Hager, eds., *Robust Vision for Vision-Based Control of Motion*, IEEE Press, Piscataway, NJ, pp. 67–77.

Hager, G. D., and Toyama, K. 1996. The XVision system: A general-purpose substrate for portable real-time vision applications. *Computer Vision and Image Understanding* 69(1):23–37.

Harris, C. 1992. Tracking with rigid models. In A. Blake and A. Yuille, eds., *Active Vision*, MIT Press, Cambridge, MA, pp. 59–73.

Hu, Y., Eagleson, R., and Goodale, M. A. 1999. Human visual servoing for reaching and grasping: The role of 3D geometric features. In *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA'99*, Vol. 3, pp. 3209–3216.

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

Koller, D., Daniilidis, K., and Nagel, H. H. 1993. Model-based object tracking in monocular image sequences of road traffic scenes. *International Journal of Computer Vision* 10(3):257–281.

Kragic, D. 2001. *Visual Servoing for Manipulation: Robustness and Integration Issues*. Ph.D. Thesis, Computational Vision and Active Perception Laboratory (CVAP), Royal Institute of Technology, Stockholm, Sweden.

Kragic, D., and Christensen, H. I. January 2002. Survey on visual servoing for manipulation. Technical report, ISRN KTH/NA/P–02/01–SE, Computational Vision and Active Perception Laboratory, Royal Institute of Technology, Stockholm, Sweden. See http://www.nada.kth.se/cvap/cvaplop/lop-cvap.html.

Lowe, D. 1985. *Perceptual Organization and Visual Recognition*. Robotics: Vision, Manipulation and Sensors. Kluwer Academic, Dordrecht.

Nayar, S. K., Nene, S. A., and Murase, H. 1996. Subspace methods for robot vision. *IEEE Transactions on Robotics and Automation* 12(5):750–758.

Petersson, L., Austin, D., and Christensen, H. I. 2001. DCA: A Distributed Control Architecture for Robotics. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems IROS'2001*, October, Maui, Hawaii, Vol. 4, pp. 2361–2368.

Petersson, L., Jensfelt, P., Tell, D., Strandberg, M., Kragic, D., and Christensen, H. 2002. Systems integration for real-world manipulation tasks. *IEEE International Conference on Robotics and Automation, ICRA 2002*, Vol. 3, pp. 2500–2505.

Roobaert, D., Zillich, M., and Eklundh, J.-O. 2001. A pure learning approach to background-invariant object recognition using pedagocical support vector learning. In *Proceedings of IEEE Computer Vision and Pattern Recognition, CVPR'01*, December, Kauai, Hawaii, Vol. 2, pp. 351–357.

Rosenblatt, J. K., and Thorpe, C. 1995. Combining multiple goals in a behavior-based architecture. In *IEEE International Conference on Intelligent Robots and Systems, IROS'95*, Vol. 1, pp. 136–141.

Shi, J., and Tomasi, C. 1994. Good features to track. In *Proceedings of the IEEE Computer Vision and Pattern Recognition, CVPR'94*, pp. 593–600.

Shimoga, K. B. 1996. Robot grasp synthesis algorithms: A survey. *International Journal of Robotics Research* 15(3):230–266.

Tell, D. 2002. *Wide baseline matching with applications to visual servoing*. Ph.D. Thesis, Computational Vision and Active Perception Laboratory (CVAP), Royal Institute of Technology, Stockholm, Sweden.

Thompson, R. L., Reid, I. D., Munoz, L. A., and Murray, D. W. 2001. Providing synthetic views for teleoperation using visual pose tracking in multiple cameras. *IEEE Systems, Man, and Cybernetics, Part A* 31(1):43–54.

Toyama, K., and Hager, G. 1996. Incremental focus of attention for robust visual tracking. In *Proceedings of the Computer Society Conference on Computer Vision and Pattern Recognition, CVPR'96*, pp. 189–195.

Vincze, M., Ayromlou, M., and Kubinger, W. 1999. An integrating framework for robust real-time 3D object tracking. In *Proceedings of the 1st International Conference on Computer Vision Systems, ICVS'99*, pp. 135–150.

Wilson, W., Williams Hulls, C. C., and Bell, G. S. 1996. Relative end-effector control using Cartesian position based visual servoing. *IEEE Transactions on Robotics and Automation* 12(5):684–696.

Wunsch, P., and Hirzinger, G. 1997. Real-time visual tracking of 3D objects with dynamic handling of occlusion. In *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA'97*, Vol. 2, pp. 2868–2873.

Wunsch, P., Winkler, S., and Hirzinger, G. 1997. Real-Time pose estimation of 3D objects from camera images using neural networks. In *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA'97*, April, Albuquerque, NM, Vol. 3, pp. 3232–3237.