

Visual Servoing Meets the Real World

Danica Kragic & Henrik I. Christensen
Centre for Autonomous Systems
Royal Institute of Technology
{danik,hic}@nada.kth.se

July 5, 2002

1 Introduction

Visual servoing is by now a mature research field in the sense that the basic paradigms largely have been defined as for example seen in the tutorial by [1]. Over the last few years a variety of visual servoing refinements have been published. It is, however, characteristic that relatively few truly operational systems have been reported. Given that the basic methods are available, why is it that so few *systems* are reported in the literature. Some examples have been reported. Good examples include the work on vision for highway driving by [2] and the work on vision based assembly by [3] and [4], and the work on vision guided welding by [5]. It is characteristic for these applications that they are implemented in setting where the scenario and its constituent objects are well defined and can be model at great detail prior to the execution of the task.

1.1 Issues in servoing

For use of visual servoing in a general setting such as a regular domestic environment it is not only necessary to perform the basic servoing based on a set of well defined features, the full process involves:

1. Recognition of the object to interact with
2. Detection of the features to be used for servoing
3. Initial alignment of model to features (pose estimation or initial data association)
4. Servoing using the defined features

In addition it must be considered that most visual servoing techniques have a limited domain of operation, i.e. some techniques are well suited for point based estimation, while others are well suited for detailed alignment at close range. For many realistic tasks there is consequently a need to decompose the task into several visual servoing stages to allow robust operation. I.e. something like

1. Point based visual servoing to achieve initial alignment
2. Approximate alignment using 2.5D or homography based servoing
3. Model based servoing to achieve high quality alignment

Depending on the task at hand these tasks might be executed in parallel (i.e. in car driving the coarse alignment is used for longer term planning and model estimation, while the model based techniques is used for short term correction – model based correction), or in sequence (initial alignment, that can be used for initialization of the intermediate model, which in term provides an estimate for model initialization). In this paper we will in particular study the manipulation of objects in a domestic setting and consequently the sequential model will be used.

1.2 Outline

In the next section (2) the basic processes involved in end-to-end visual servoing are outlined. To achieve robustness it is necessary to utilize a mixture of object models (object models, appearance models, geometric models) and fusion of multiple cues to compensate for variations in illumination, viewpoint, and surface texture, this is discussed in section 3. The use of the presented methodology is illustrated for the task of manipulating a variety of objects in a living room setting as reported in section 4. Finally a few observations and issues for future research are discussed in section 5.

2 Processes in Vision based Grasping

2.1 Detection/Recognition

The object to be manipulated is first recognized using the view-based SVM (support vector machine) system presented in [6]. For execution of a task such as “pickup coke can on the dinner table” the robot will navigate to the living room, and position itself with the camera directed towards the dinner table. The recognition step will then deliver regions in the image that may represent the object of interest. The recognition step delivers the image position and approximate size of the image region occupied by the object. This information is required by the tracking system to allow initialization through use of a *window of attention*. Initially the recognition step is carried out at a distance of 2-3 meters to allow use of a field of view that is likely to be large enough include the object of interest. Once a valid image region has been identified the position and scale information is delivered to the image based servoing step.

2.2 Object Approach

While approaching the object (getting within reach), we want to keep it 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*, Fig. 1. The objective here is to track a part of an image (a region) between frames. The image position of its center is denoted with $\mathbf{p} = [x \ y]^T$. Hence, the state is $\mathbf{x} = [x \ y \ \dot{x} \ \dot{y}]^T$ where a piecewise constant white acceleration model is used [7]:

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

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

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

For prediction and estimation, the $\alpha - \beta$ filter is used, [7]:

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

with

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

where α and β are determined using steady-state analysis. The tracking of the object is

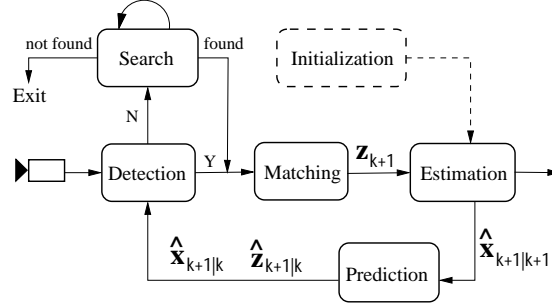


Figure 1: A schematic overview of the tracking system.

used to direct the arm so as to ensure that the TPC is pointing at the object while it is being approached. The standard image Jacobian is being used for specification of the control of the arm. Once the object occupies more than, an object, specified portion of the image the servoing is terminated and control is handed over to the coarse alignment control for pre-grasping and gripper - object alignment.

2.3 Coarse alignment

The basic idea is the use of a reference position for the camera. It is assumed to be known how the object can be grasped from the reference position (see Fig. 2).

Using a stored image taken from the reference position, the manipulator should be moved in such a way that the current camera view is gradually changed to match the stored reference view. Accomplishing this for general scenes is difficult, but a robust system can be made under the assumption that the objects are piecewise planar [8], i.e. a homography models the transformation between the current and the reference view. The scheme implemented requires three major components – initialization, servoing and tracking. Each component exploits the planarity of the tracked object. For initialization, a wide baseline matching algorithm [8] is employed to establish point correspondences between the current and the reference image. The point correspondences enable the computation of the homography H relating the two views. Assuming



Figure 2: The left image shows the gripper in a reference position while the right image shows the camera in another position and situation. Servoing is then used to move the camera to the reference position relative to the object.

known internal camera parameters, the homography matrix is used in the “2.5D” servoing algorithm of Malis et al [9]. Finally, as initialization is computationally expensive, matches are established in consecutive images using tracking. This is accomplished by making a prediction of the new homography H' relating the current and the reference images, given the arm odometry between frames and the homography H relating the reference image and the previous image [8]. H' is then used in a guided search for new correspondences between the current image and the reference image [10].

2.4 Grasp alignment

Although suitable for a number of tasks, previous approaches lack 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 to, for example, allow the alignment of the robot arm with the object or to generate a feasible grasp and execute a grasp of the object. 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- and appearance-based representations may be employed [11].

A recent study of human visually guided grasps in situations similar to that typically used in visual servoing control, [12] 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. These computations are more complex than those typically carried out in visual servoing systems and permit humans to operate in large range of environments.

We have therefore decided to integrate both appearance based and geometrical models to solve different steps of a manipulation task. Many similar systems use manual pose initialization where the correspondence between the model and object features is given by the user, (see [13] and [5]). Although there are systems where this step is performed automatically [14], [15] proposed approaches are time consuming and not appealing for real-time applications. One additional problem, in our case, is that the objects to be manipulated by the robot are highly textured (see Fig. 3) and therefore not suited for matching approaches based on, for example, line features [16], [17], [18].

After the object has been recognized and its position in the image is known, an



Figure 3: Some of the objects we want robot to manipulate.

appearance based method is employed to estimate its initial pose. The method we have implemented has been initially proposed in [19] where just three pose parameters have been estimated and used to move a robotic arm to a predefined 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 different number of degrees of freedom.

Compared to the system proposed in [20], where the network has been entirely trained on simulated images, we use real images for training where no particular background was considered. As pointed out in [20], the illumination conditions (as well as the background) strongly affect the performance of their system and these can not be easily obtained with simulated images. In addition, the idea of projecting just the wire-frame model to obtain training images can not be employed in our case due to the objects' texture. The system proposed in [16] 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 find with high certainty.

Compared to the system proposed in [20], where the network has been entirely trained on simulated images, we use real images for training where no particular background was considered. As pointed out in [20], the illumination conditions (as well as the background) strongly affect the performance of their system and these can not be easily obtained with simulated images. In addition, the idea of projecting just the wire-frame model to obtain training images can not be employed in our case due to the objects' texture. The system proposed in [16] 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 find with high certainty.

Our model based tracking system is presented in Fig. 4. Here, a wire-frame model of the object is used to estimate the pose and the velocity of the object in each frame. The model is used during the *initialization* step where the initial pose 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 (velocity, acceleration) 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 systems 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.

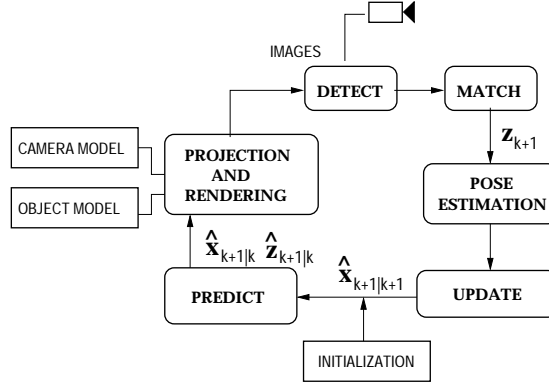


Figure 4: Block diagram of the model based tracking system.

2.4.1 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:

$$\mathbf{x} = [X, Y, Z, \phi, \psi, \gamma, \dot{X}, \dot{Y}, \dot{Z}, \dot{\phi}, \dot{\psi}, \dot{\gamma}] \quad (4)$$

where ϕ , ψ and γ represent roll, pitch and yaw angles [21]. The following piecewise constant white acceleration model is considered [7]:

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{F}\mathbf{x}_k + \mathbf{G}\mathbf{v}_k \\ \mathbf{z}_k &= \mathbf{H}\mathbf{x}_k + \mathbf{w}_k \end{aligned} \quad (5)$$

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

$$\mathbf{F} = \begin{bmatrix} \mathbf{I}_{6 \times 6} & \Delta T \mathbf{I}_{6 \times 6} \\ \mathbf{0} & \mathbf{I}_{6 \times 6} \end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix} \frac{\Delta T^2}{2} \mathbf{I}_{6 \times 6} \\ \Delta T \mathbf{I}_{6 \times 6} \end{bmatrix}, \quad \mathbf{H} = [\mathbf{I}_{6 \times 6} \mid \mathbf{0}] \quad (6)$$

For prediction and update, the $\alpha - \beta$ filter is used:

$$\begin{aligned} \hat{\mathbf{x}}_{k+1|k} &= \mathbf{F}_k \hat{\mathbf{x}}_k \\ \hat{\mathbf{z}}_{k+1|k} &= \mathbf{H} \hat{\mathbf{x}}_{k+1|k} \\ \hat{\mathbf{x}}_{k+1|k+1} &= \hat{\mathbf{x}}_{k+1|k} + \mathbf{W}[\mathbf{z}_{k+1} - \hat{\mathbf{z}}_{k+1|k}] \end{aligned} \quad (7)$$

Here, the pose of the target is used as measurement rather than image features, as commonly used in the literature (see, for example, [22], [14]). An approach similar to

the one presented here is taken in [18]. 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.

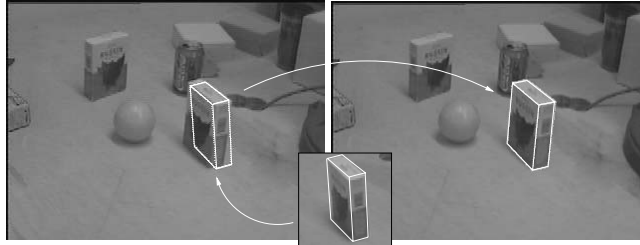


Figure 5: On the left is the initial pose estimated using PCA approach. On the right is the pose obtained by local refinement method.

2.4.2 Initial Pose Estimation

Initialization step uses the ideas proposed in [19]. During training, each image is projected as a point to the eigen-space and the corresponding pose of the object is stored with each point. For each object, we have used 96 training images (8 rotations for each angle on 4 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 pixels color images. The training procedure takes about 3 minutes 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 Fig. 5. 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 compensate for it. 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 Fig. 6. 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. In addition,

2.4.3 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



Figure 6: Training image used to estimate the initial pose (far left) followed by the intermediate images of the fitting step.

object onto the image plane. For each visible edge, a number of image points is 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 [23]. After that, 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, see Fig. 7.

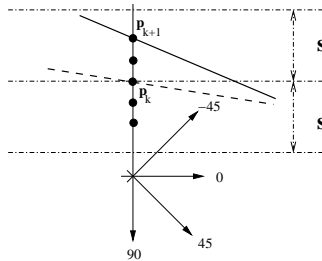


Figure 7: Determining normal displacements for points on a contour in consecutive frames.

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 depends on the distance of the objects from the camera.

After the normal displacements are available, the method proposed in [5] is used. Lie group and Lie algebra formalism are used as the basis for representing the motion of a rigid body and pose estimation. Implementation details can be found in [24].

2.5 Grasping

The ability of a system to generate both a *feasible* and a *stable* grasp adds to its robustness. By a *feasible* grasp, a kinematically feasible grasp is considered, that is, given the pose of the object the estimated configuration of the manipulator and the end-effector are kinematically valid. By a *stable* grasp, a grasp for which the object will not twist or slip relative to the end-effector.

We have integrated our model-based vision system with a model-based grasp planning and visualization system called GraspIt! (see [25] and [26] for more details). To

plan a grasp, grasp planners need to know the pose of objects in a workspace. For that purpose a vision system may be employed which, in addition, may be used to determine how a robot hand should approach, grasp, and move an object to a new configuration. A synergistic integration of a grasp simulator and a model based visual system is described. These work in concert to i) find an object’s pose, ii) generate grasps and movement trajectories, and iii) visually monitor the execution of a servoing task [27].

An example grasp is presented in Fig. 8. It demonstrates a successfully planned and executed stable grasp of an L-shaped object. After the first image of the scene is acquired, the estimated pose of the object is sent to GraspIt, which aids the user in selecting an appropriate grasp. After a satisfactory grasp is planned, it is executed by controlling both the robot arm and the robot hand. As can be seen in the figures, the grasping is performed in two steps. The first step sends the arm to the vicinity of the object, i.e., to the pose form which the grasp will be performed. After the arm is positioned, the hand may be closed.



Figure 8: Grasping of the L-shaped object: The pose of the object is estimated in the camera coordinate frame (the estimated pose is overlaid in black). The pose is transformed to the manipulator coordinate system using the off-line estimated camera/robot transformation. The estimated pose is propagated to GraspIt (first row, right) where a stable grasp is planned (the measures of the quality of the grasp are shown in the lower left corner of the image: $e=0.317$ and $v=1.194$). The planned grasp is used to position the manipulator and the hand.

3 Robustness

To achieve robustness in realistic situations it is necessary to deploy multiple features and integrate these into a joint representation. To fusion of multiple cues here deploys a voting strategy to enable use of weak model for the integration of cues.

3.1 Multiple Cues

3.1.1 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 the c_i ’s 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, disparity. Mathematically, a cue is formalized as a mapping from an action space, \mathbf{A} , to the interval $[0,1]$:

$$c : \mathbf{A} \rightarrow [0, 1] \quad (8)$$

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 [28], [29], we define the following voting scheme:

Definition 3.1 - Weighted Plurality Approval Voting For a group of homogeneous cues, $\mathbf{C} = \{c_1, \dots, c_n\}$, 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:

$$\delta(a) = \sum_{i=1}^n w_i O_{c_i}(a) \quad (9)$$

where the most appropriate action is selected according to:

$$a' = \operatorname{argmax}\{\delta(a) | a \in \mathbf{A}\} \quad (10)$$

3.1.2 Visual Cues

The cues considered in the integration process are:

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

$$SSD(u, v) = \sum_n \sum_m [I(u + m, v + n) - T(m, n)]^2 \quad (11)$$

where $I(u, v)$ and $T(u, v)$ represent the grey level values of the image and the template, respectively.

Color - It has been shown in [30] 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 component since 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 image is segmented into regions of motions and regions of inactivity. This is estimated using image differencing:

$$M[(u, v), k] = \mathcal{H}[|I[(u, v), k] - I[(u, v), k - 1]| - \Gamma] \quad (12)$$

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

$$\mathcal{H}(x) = \begin{cases} 0 & : x \leq 0 \\ x & : x > 0 \end{cases} \quad (13)$$

Intensity Variation - In each frame, the following is estimated for all $m \times m$ (details about m are given in Section 3.1.4) regions inside the tracked window:

$$\sigma^2 = \frac{1}{m^2} \sum_u \sum_v [I(u, v) - \bar{I}(u, v)]^2 \quad (14)$$

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 was rich in texture large variation is expected. The level of texture is evaluated as proposed in [31].

3.1.3 Weighting

In (Eq. 9) 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 determined either i) *a-priori* and kept constant during tracking, or ii) estimated during tracking based on cue's success to estimate the final result or based on how much it is in agreement with other cues. In our previous work the following methods were evaluated [24]:

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 - 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

$$d_i = \|\mathbf{z}_k^i - \hat{\mathbf{z}}_{k|k-1}\| \quad (15)$$

and errors are estimated as

$$e_i = \frac{d_i}{\sum_{i=1}^n d_i}. \quad (16)$$

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

4. History-based distance weighting - 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 the agreement with other cues. Weights are then simply proportional to this value:

$$w_i = \frac{S_i}{\sum_{i=1}^n S_i} \text{ with } \sum_{i=1}^n w_i = 1 \quad (17)$$

3.1.4 Implementation

We have investigated two approaches where voting is used for: i) *response fusion*, and ii) *action fusion*. The first approach makes the use of “raw” responses from the employed visual cues in the image which also represents the action space, \mathbf{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 Fig. 9. 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, \mathbf{A} , which is now the velocity space.

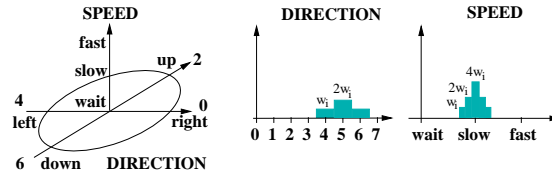


Figure 9: A schematic overview of the *action fusion* approach: the desired direction is (*down and left*) with a (*slow*) speed. The use of bins represents a *neighborhood voting scheme* which ensures that slight differences between different cues do not result in an unstable classification.

Initialization

According to Fig. 1, a tracking sequence should be initiated by *detecting* the target object. If a recognition module is not available, another strategy can be used. In [32] 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 ideas, 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 has either i) left the field of view, or ii) it 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.

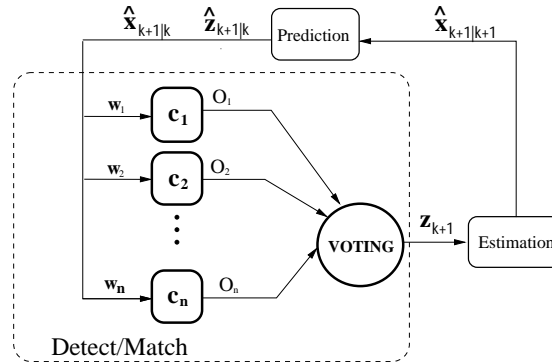


Figure 10: A schematic overview of the *response fusion* approach.

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 Fig. 10. Here, \mathbf{x} represents a position:

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

$$\begin{aligned} 0 \leq O_{color}(\mathbf{x}, k) \leq 255 \quad \text{with} \\ \mathbf{x} \in [\hat{\mathbf{z}}_{k|k-1} - 0.5\mathbf{x}_w, \hat{\mathbf{z}}_{k|k-1} + 0.5\mathbf{x}_w] \end{aligned} \quad (18)$$

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

Motion- Using (Eq. 12) and (Eq. 13) with $\Gamma = 10$, image is segmented into regions of motion and inactivity:

$$\begin{aligned} 0 \leq O_{motion}(\mathbf{x}, k) \leq 255 - \Gamma \quad \text{with} \\ \mathbf{x} \in [\hat{\mathbf{z}}_{k|k-1} - 0.5\mathbf{x}_w, \hat{\mathbf{z}}_{k|k-1} + 0.5\mathbf{x}_w] \end{aligned} \quad (19)$$

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

$$\begin{aligned} O_{SSD}(\mathbf{x}, k) = 255e^{(-\frac{\mathbf{x}^2}{2\sigma^2})} \quad \text{with } \sigma = 5 \\ \mathbf{x} \in [\mathbf{z}_{SSD} - 0.5\mathbf{x}_w, \mathbf{z}_{SSD} + 0.5\mathbf{x}_w], \\ \bar{\mathbf{x}} \in [-0.5\mathbf{x}_w, 0.5\mathbf{x}_w] \end{aligned} \quad (20)$$

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

$$\begin{aligned} 0 \leq O_{var}(\mathbf{x}, k) \leq 255 \quad \text{with} \\ \mathbf{x} \in [\hat{\mathbf{z}}_{k|k-1} - 0.5\mathbf{x}_w, \hat{\mathbf{z}}_{k|k-1} + 0.5\mathbf{x}_w] \end{aligned} \quad (21)$$

Response Fusion

The estimated responses are integrated using (Eq. 9):

$$\delta(\mathbf{x}, k) = \sum_i^n w_i O_i(\mathbf{x}, k) \quad (22)$$

However, (Eq. 10) can not be directly used for selection, as there might be several pixels with same number of votes. Therefore, this equation is slightly modified:

$$\delta'(\mathbf{x}, k) = \begin{cases} 1 : & \text{if } \delta(\mathbf{x}, k) \text{ is } \operatorname{argmax}\{\delta(\mathbf{x}', k) | \mathbf{x}' \\ & \in [\hat{\mathbf{z}}_{k|k-1} - 0.5\mathbf{x}_w, \hat{\mathbf{z}}_{k|k-1} + 0.5\mathbf{x}_w]\} \\ 0 : & \text{otherwise} \end{cases} \quad (23)$$

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

Action Fusion Approach

Here, the action space is defined by a direction d and speed s , see Fig. 9. Both the direction and the speed are represented by histograms of discrete values where the direction is represented by eight values, see Fig. 11:

$$\begin{aligned} & \text{LD} \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \text{L} \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \text{LU} \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \text{U} \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \\ & \text{RU} \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \text{R} \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \text{RD} \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \text{D} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \end{aligned} \quad (24)$$

with L-left, R-right, D-down and U-up. 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, and 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 Fig. 9 a *neighborhood voting scheme* is used to ensure that slight differences between different cues do not result in an unstable classification. (Eq. 3) is modified so that:

$$\mathbf{H} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ and } \mathbf{W} = \begin{bmatrix} \alpha\Delta T & 0 & \beta & 0 \\ 0 & \alpha\Delta T & 0 & \beta \end{bmatrix}^T \quad (25)$$

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

Color - The response of the color cue is first estimated according to (Eq. 18) followed by:

$$\mathbf{a}_{color}(k) = \frac{\sum_{\mathbf{x}} O_{color}(\mathbf{x}, k)\mathbf{x}(k)}{\sum_{\mathbf{x}} \mathbf{x}(k)} - \hat{\mathbf{p}}_{k|k-1} \quad (26)$$

$$\text{with } \mathbf{x} \in [\hat{\mathbf{p}}_{k|k-1} - 0.5\mathbf{x}_w, \hat{\mathbf{p}}_{k|k-1} + 0.5\mathbf{x}_w]$$

where $\mathbf{a}_{color}(k)$ represents the desired action and $\hat{\mathbf{p}}_{k|k-1}$ is the predicted position of the tracked region. 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:

$$\mathbf{a}_{SSD}(k) = \operatorname{argmin}_{\mathbf{x}} (SSD(\mathbf{x}, k)) - \hat{\mathbf{p}}_{k|k-1} \quad (27)$$

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:

$$\begin{aligned} \text{direction } d_i &= \mathcal{P}(\operatorname{sgn}(\mathbf{a}_i)), \\ \text{speed } s_i &= \|\mathbf{a}_i\| \end{aligned} \quad (28)$$

where $\mathcal{P} : \mathbf{x} \rightarrow \{0, 1, \dots, 7\}$ is a scalar function that maps the two-dimensional direction vectors (see (Eq. 24)) 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 a weight, w_i , are used to update the direction and speed of the histograms according to Fig. 9 and (Eq.9). The new measurement is then estimated by multiplying the actions from each histogram which received the maximum number of votes according to (Eq. 10):

$$\mathbf{z}_k = \mathcal{S}(\operatorname{argmax}_d HD(d)) \operatorname{argmax}_s HS(s) \quad (29)$$

where $\mathcal{S} : x \rightarrow \{[\begin{smallmatrix} -1 \\ 0 \end{smallmatrix}], \dots, [\begin{smallmatrix} -1 \\ 1 \end{smallmatrix}]\}$. The update and prediction steps are then performed using (Eq. 25) and (Eq. 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 it has been pointed out in [29], that arbitration via vector addition can result in commands which are not satisfactory to any of the contributing cues.

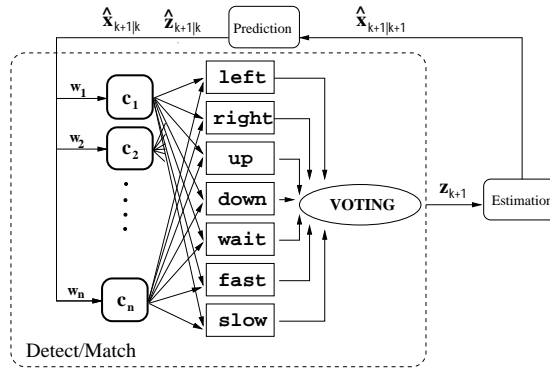


Figure 11: A schematic overview of the *action fusion* approach.

4 Experiments

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 predefined 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 including several objects. The robot is instructed to pick up a package of rice having an arbitrary placement. Here, Distributed Control Architecture [33] is used for integration of the different methods into a fully operational system. The system includes i) recognition, ii) image based tracking, iii) initial pose estimation, iv) hand alignment for grasping, v) grasp execution and vi) delivery of object. The details of the system implementation are unfortunately beyond this paper. The results of a mission with the integrated system are outlined below.

The sequence in Fig. 12 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 recognizes it and locates in the scene. After that, the robot moves closer to the table keeping the rice package centered

in the image. Finally, the gripper is aligned with the object and grasping is performed. The details about the alignment can be found in [34].



Figure 12: 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.

5 Summary

A key problem of deployment of manipulation in realistic setting is integration of a multitude of methods into a unified framework that manages the different phases of grasping from i) identification, ii) approach, iii) pre-grasping, and iv) grasp execution. In this paper we have presented a system in which all of these aspects are integrated into an operational system. Each of the participating processes have been designed to accommodate handling of a large variety of situations. No single visual servoing technique is well suited for operation of a large range variations, consequently techniques with varying operation characteristics have been combined to provide an end-to-end system that can operate in situations with realistic complexity.

At present the system is tailored to the task in terms of sequential execution of the different steps in the servoing process. To achieve smooth transition between the different phases it is naturally of interest to provide a control framework for integrated modelling of the control, which will be a key problem to be studied as part of future work. In addition generality across a range of different application domains will be studied.

Acknowledgment

This research has been carried out by the Intelligent Service Robot team at the Centre for Autonomous Systems. The work has involved P. Jensfelt, A. Oreback, D. Tell, M. Strandberg, A. Miller, D. Roobaert, L. Petersson and M. Zillich. Their contribution to the integration of the overall system and numerous discussions is gratefully acknowledged.

The research has been sponsored by the Swedish Foundation for Strategic Research, this support is gratefully acknowledged.

References

- [1] S. Hutchinson, G. Hager, and P. Corke, "A tutorial on visual servo control," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 5, pp. 651–670, 1996.
- [2] E. Dickmanns, "Vehicles capable of dynamic vision: a new breed of technical beings?," *Artificial Intelligence*, vol. 103, pp. 49–76, August 1998.
- [3] H. Kollnig and H. Nagel, "3D pose estimation by directly matching polyhedral models to gray value gradients," *International Journal of Computer Vision*, vol. 23, no. 3, pp. 282–302, 1997.
- [4] G. Hirzinger, M. Fischer, B. Brunner, R. Koeppel, M. Otter, M. Gerbenstein, and I. Schäfer, "Advanced in robotics: The DLR experience," *International Journal of Robotics Research*, vol. 18, pp. 1064–1087, November 1999.
- [5] T. Drummond and R. Cipolla, "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, 2000.
- [6] D. Roobaert, *Pedagogical Support Vector Learning: A Pure Learning Approach to Object Recognition*. PhD thesis, Computational Vision and Active Perception Laboratory (CVAP), Royal Institute of Technology, Stockholm, Sweden, May 2001.
- [7] Y. Bar-Shalom and Y. Li, *Estimation and Tracking: Principles, techniques and software*. Artech House, 1993.
- [8] D. Tell, *Wide baseline matching with applications to visual servoing*. PhD thesis, Computational Vision and Active Perception Laboratory (CVAP), Royal Institute of Technology, Stockholm, Sweden, 2002.
- [9] E. Malis, F. Chaumette, and S. Boudet, "Positioning a coarse-calibrated camera with respect to an unknown object by 2-1/2-d visual servoing," in *ICRA*, pp. 1352–1359, 1998.
- [10] P. Pritchett and A. Zisserman, "Wide baseline stereo matching," in *ICCV*, pp. 863–869, 1998.
- [11] L. Bretzner, *Multi-scale feature tracking and motion estimation*. PhD thesis, CVAP, NADA, Royal Institute of Technology, KTH, 1999.
- [12] Y. Hu, R. Eagleson, and M. Goodale, "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, 1999.
- [13] N. Giordana, P. Bouthemy, F. Chaumette, and F. Spindler, "Two-dimensional model-based tracking of complex shapes for visual servoing tasks," in *Robust vision for vision-based control of motion* (M. Vincze and G. Hager, eds.), pp. 67–77, IEEE Press, 2000.
- [14] V. Gengenbach, H.-H. Nagel, M. Tonko, and K. Schäfer, "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, 1996.
- [15] D. Lowe, *Perceptual Organisation and Visual Recognition*. Robotics: Vision, Manipulation and Sensors, Dordrecht, NL: Kluwer Academic Publishers, 1985. ISBN 0-89838-172-X.
- [16] M. Vincze, M. Ayromlou, and W. Kubinger, "An integrating framework for robust real-time 3D object tracking," in *Proceedings of the First International Conference on Computer Vision Systems, ICVS'99*, pp. 135–150, 1999.
- [17] D. Koller, K. Daniilidis, and H. Nagel, "Model-based object tracking in monocular image sequences of road traffic scenes," *International Journal of Computer Vision*, vol. 10, no. 3, pp. 257–281, 1993.
- [18] P. Wunsch and G. Hirzinger, "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, 1997.

- [19] S. Nayar, S. Nene, and H. Murase, "Subspace methods for robot vision," *IEEE Transactions on Robotics and Automation*, vol. 12, pp. 750–758, October 1996.
- [20] P. Wunsch, S. Winkler, and G. Hirzinger, "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*, vol. 3, (Albuquerque, New Mexico), pp. 3232–3237, April 1997.
- [21] J. Craig, *Introduction to Robotics: Mechanics and Control*. Addison Wesley Publishing Company, 1989.
- [22] E. Dickmanns and V. Graefe, "Dynamic monocular machine vision," *Machine Vision nad Applications*, vol. 1, pp. 223–240, 1988.
- [23] J. Foley, A. van Dam, S. Feiner, and J. Hughes, eds., *Computer graphics - principles and practice*. Addison-Wesley Publishing Company, 1990.
- [24] D. Kragic, *Visual Servoing for Manipulation: Robustness and Integration Issues*. PhD thesis, Computational Vision and Active Perception Laboratory (CVAP), Royal Institute of Technology, Stockholm, Sweden, 2001.
- [25] A. Miller and P. Allen, "Examples of 3D grasp quality computations," in *Proceedings of the of the 1999 IEEE International Conference on Robotics and Automation*, pp. 1240–1246, 1999.
- [26] A. T. Miller and P. Allen, "Grasplit!: A versatile simulator for grasping analysis," in *To appear in Proceedings of the of the 2000 ASME International Mechanical Engineering Congress and Exposition*, 2000.
- [27] D. Kragić, A. Miller, and P. Allen, "Realtime tracking meets online grasp planning," in *IEEE International Conference on Robotics and Automation, ICRA'01*, 2001.
- [28] D. Blough and G. Sullivan, "Voting using predispositions," *IEEE Transactions on reliability*, vol. 43, no. 4, pp. 604–616, 1994.
- [29] J. Rosenblatt and C. Thorpe, "Combining multiple goals in a behavior-based architecture," in *IEEE International Conference on Intelligent Robots and Systems, IROS'95*, vol. 1, pp. 136–141, 1995.
- [30] H. Christensen, D. Kragic, and F. Sandberg, "Vision for interaction," in *Intelligent Robotic Systems* (G. Hager and H. Christensen, eds.), Lecture notes in Computer Science, Springer, 2001. to be published.
- [31] J. Shi and C. Tomasi, "Good features to track," in *Proceedings of the IEEE Computer Vision and Pattern Recognition, CVPR'94*, pp. 593–600, 1994.
- [32] K. Toyama and G. Hager, "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, 1996.
- [33] L. Petersson, D. Austin, and H. Christensen, "DCA: A Distributed Control Architecture for Robotics," in *Proc. IEEE International Conference on Intelligent Robots and Systems IROS'2001*, vol. 4, (Maui, Hawaii), pp. 2361–2368, October 2001.
- [34] D. Tell, *Wide baseline matching with applications to visual servoing*. Phd thesis, Computational Vision and Active Perception Laboratory (CVAP), Royal Institute of Technology, Stockholm, Sweden, march 2002.