ABSTRACT

In this paper we present a few different vision techniques used for robotic manipulation and grasping tasks. These tasks are performed in a domestic environment and consider manipulation of everyday objects such as a package of cereals, cups, soda cans, etc. The variety and dynamics of real world settings introduce a number of problems such as varying lighting conditions, complicated textural properties of objects, shadows, etc. In order to perform tasks under such conditions, the high level of robustness of the overall system is of key importance. Given a task at hand, different approaches of how to gain the needed robustness are discussed.

To perform a task, we have integrated a number of different vision based approaches - recognition/detection, 2D tracking for position estimation and 3D visual tracking for pose (position and orientation) estimation. In this paper, the objective is mainly on the tracking approaches and their integration to obtain a fully operational, robotic grasping system.

Keywords: Visual servoing, tracking, voting, pose estimation, manipulation, grasping

1 INTRODUCTION

Real world robotic applications pose many problems to the design of reliable and flexible systems. The tasks that we want a robot to perform, are often considered trivial from our, human point of view. As humans we have and use a diverse set of prior-knowledge while performing "simple" tasks like opening a door or fetching a soda-can from the table - first we have to detect the object, prepare the grasp and then grasp/manipulate the object. This reasoning implies that a complicated task may be designed as a series of simpler, easily performable tasks.

For many living species, not least in the case of humans, visual perception plays a key role in their behavior and the hand–eye coordination ability gives us flexibility, dexterity and robustness of movement that no machine can match yet. To locate and identify static, as well as moving, objects, to determine how to grasp and handle them, we often rely strongly on our visual sense. One of the important factors is our ability to track objects, that is, to maintain an object in the field of view for a period of time using our oculomotor system as well as head and body motions.

In this paper, we present visual techniques for robot control in domestic setting (i.e. a living room). We argue that a robust performance of robotic systems in natural, everyday environments such as, for example, our home or office, may be achieved through an integration of different techniques with different levels of complexity. The main objective of our work is visual tracking - estimating the target’s state (position, rotation, velocity) and its use for robot control. Two different techniques are presented: i) feature based 2D visual tracking, and ii) model based 3D object tracking. Given a task at hand, such as person following or robotic grasping, each of the methods is presented and some experimental results are given.

The former technique considers fusion of multiple visual cues for image based (2D) tracking. By visual cues we mean color, texture, correlation and motion. We start with a short overview of different fusion techniques and applications mainly in the machine vision area. After that, the theoretical background of the integration by voting is presented. Using mentioned visual cues, two different fusion approaches are presented and experimentally evaluated with respect to accuracy and reliability. It is demonstrated that the robustness and real-time performance of the overall system emerges from the coordination and integration between simple visual cues.

For certain manipulation and grasping tasks the robot requires an estimate of object’s position and orientation commonly referred to as pose. The latter technique considers therefore a model based tracking system. Given a wireframe model of the object, its pose relative to the camera or robot is estimated. If the object or/and the robot start to move, the system estimates the pose of the object on–line using normal displacements along visible model contours. The system is experimentally evaluated for tracking of polyhedral but textured objects against a cluttered background. Accurate tracking and pose estimation is demonstrated as well as the ability of the system to cope with partial occlusions.
Figure 1: Some of the objects we want robot to manipulate.

2 AN EXAMPLE TASK

A common task for a service robot type of applications is the ability to fetch and manipulate objects Fig. 1. Given some basic localization and navigation capabilities [5], the robot is able to freely navigate from its starting position towards, for example, dinner table, see Fig. 2.

Figure 2: A robot moving from its starting position (left) to the dinner table (right).

After the object to be fetched is recognized, approached and its pose estimated, the grasping can be performed, see Fig. 3. The robot first has to move towards the objects (approach stage) until the pose of the objects can be estimated (final alignment stage).

Figure 3: (left) The package of rice is successfully recognized by the recognition system, (right) The final grasp can now be performed.

The rest of the paper covers in more detail the vision techniques used to guide the robot during the approach (Section 3) and final alignment stage (Section 4).

3 FEATURE BASED 2D TRACKING

In order to cope with the dynamic changes in the environment, we have decided to use a number of different visual cues and integrate them in a common framework. Voting is here adopted as the underlying integration strategy [12]. Compared to the Bayesian approaches, voting requires no detailed models of the form \( p(\text{cue}|\text{object}) \) which may be difficult or even impossible to determine. A very simple or no model is used to represent this relationship giving it the advantage to operate “model–free” with respect to individual cues. In the simplest case, each estimator may be a classifier that votes for a particular attribute or against it where the level of belief (Dempster-Shafer) or degree of uncertainty (Bayesian) is completely abstracted to give a binary output. Abstracting is a useful tool for the resolution of data from multiple sources and it is one of the fundamental problems in cases of classification, where a fused outcome must reflect group consensus rather than a compromised value.

Our tracking algorithm employs the four step detect–match–update–predict loop, Fig. 3. The objective here is to track a part of an image (a region) between frames, see Fig. 5. The image position of its center is denoted with \( p = [x \ y]^T \). Hence, the state is \( x = [x \ y \ \dot{x} \ \dot{y}]^T \) where a piecewise constant white acceleration model is used [1].

Figure 4: A schematic overview of the 2D tracking system.

The cues \( c_i \) considered in the integration process are: correlation, motion, color and intensity variation (see [7] for details). The response of each of the cues are used by a voter or a fusion center, \( \delta(A) \). where weighted plurality approval voting scheme is adopted for integration. For a group of homogeneous cues, \( C = \{c_1, \ldots, c_n\} \), 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:

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

where the most appropriate action is selected according to:

\[
d = \arg\max\{\delta(a)|a \in A\} \tag{2}
\]

In (Eq. 1), the output from each cue is weighted. Four different weighting methods were evaluated: uniform
weighting, texture based weighting, one–step distance weighting and history based distance weighting. In first two cases, the weights are preset while in the last two cases the system dynamically sets the weights based on the response of each of the cues. We briefly explain each of the approaches:

**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 estimated experimentally 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: color (0.45), image differencing (0.2), correlation (0.15), intensity variation (0.2).

**3. One-step distance weighting**
Weighting factor, \( w_i \), a cue, \( c_i \), at time step \( k \) depends on the distance from the predicted image position, \( \hat{z}_k^{[k-1]} \). Initially, the distance is estimated as \( d_i = ||z_i^k - \hat{z}_k^{[k-1]}|| \) and errors are estimated as \( e_i = d_i/\sum_{i=1}^{n} d_i \). Weights are 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 strategy is: 1. For each cue, \( c_i \), examine if \( ||z_i^k - \hat{z}_k^{[k-1]}|| < d_T \) where \( i, j = 1, \ldots, n \) and \( i \neq j \). If this is true, \( a_{ij}=1 \), otherwise \( a_{ij}=0 \). Here, \( a_{ij}=1 \) means 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.

2. Build \( (n-1) \) value set for each cue: \( c_i: \{ a_{ij} \} = \{1, \ldots, n \} \) and \( i \neq j \). Find sum \( s_i = \sum_{j=1}^{n} a_{ij} \).

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

Two integration approaches were investigated 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].

The second approach uses a different action space represented by a direction and a speed. 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.

### 3.1 Response Fusion
The responses are integrated using (Eq. 1):

\[
\delta(x, k) = \sum_{i} w_i O_i(x, k)
\]

where \( O_i \) represents the response of each of the cues. However, (Eq. 2) can not be directly used since there might be several pixels with same number of votes. Therefore, this equation is slightly modified to accommodate for this:

\[
\delta'(x, k) = \begin{cases} 
1 & \text{if } \delta(x, k) = \arg\max \{ \delta(x', k)|x'| \in [x_{k-1} - 0.5x_v, x_{k-1} + 0.5x_v]\} \\
0 & \text{otherwise}
\end{cases}
\]

Finally, the new measurement \( z_k \) is given by the mean value (first moment) of \( \delta'(x, k) \), i.e., \( z_k = \delta'(x, k) \).

### 3.2 Action Fusion
After the desired action, \( a_i(k) \), for a cue is estimated, the cue produces the votes as follows:

\[
d_k = P(\text{sgn}(a_i)) \text{, speed } s_i = ||a_i||
\]

where \( P : x \rightarrow \{0, 1, \ldots, 7\} \) is a scalar function that maps the two–dimensional direction vectors 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 (Eq.1). The new measurement is then estimated by multiplying the actions from each histogram which received the maximum number of votes according to (Eq. 2):

\[
z_k = S(\arg\max_{d} HD(d)) \arg\max_{S} HS(s)
\]

where \( S: x \rightarrow \{[\pm 1] \ldots [\pm 1]\} \). 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 [13], that arbitration via vector addition can result in commands which are not satisfactory to any of the contributing cues.

Figure 5: An example of the initial and end image during 2D visual tracking.
3.3 Experimental Results

An extensive experimental evaluation is presented in [7]. Some of those results are summarized here. The proposed approaches have been evaluated through their accuracy and reliability. 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 presented through the mean square error and standard deviation in pixels. The measure of the reliability is on a yes/no basis depending on if 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.

We have evaluated the effect of different weighting techniques on the overall system performance. The results are obtained for 10 sequences and for each sequence 3 different sizes of the window of attention were used: 25×25, 35×35 and 45×45 pixels. The target undergoes arbitrary 3D motion.

**Accuracy** (Table 1) - Here, the distance measure is used as an error indicator. The overall results are presented in Table 1 for the proposed fusion approaches. The results show that the best accuracy is achieved with fixed weights using the texture based weighting and the uniform weighting. The one-step distance weighting gives a reward to a cue each time when the cue performs satisfactorily and there is no ability to determine the overall performance of the cues during the sequence. It was expected that this problem would be solved using the history based weighting but, on the other hand, temporal smoothing results in a slow weight assignment dynamics. One solution to this problem might be to change the model and instead of using all frames up to the current one, apply a temporal windowing approach. This would allow the use of the immediate history to evaluate the performance of each cue.

Comparing the performance for fusion approaches shows that action fusion approach had higher standard deviation (14 pixels for texture based weighting). 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 cases, 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.

In many cases it is, however, more important to retain the tracking at the cost of a lower accuracy. For that purpose the reliability measure is important.

**Reliability** (Table 2) - Here, the influence of choice of the weight assignment technique on the success rate of the response and action fusion approaches is discussed. As for the accuracy, the reliability was estimated for 30 test runs and the percentage of the success is presented. Ranking the results shows that the texture based weighting performed most reliably - the target was successfully tracked during 27 test runs.

Comparing the overall results, texture weighting approach resulted in both the highest accuracy and reliability. Uniform weighting, although very accurate according to the results in Table 1, performed worst in terms of reliability.

<table>
<thead>
<tr>
<th>Uniform Weights</th>
<th>Texture Weighting</th>
<th>One-step Dist. Weighting</th>
<th>Hist. Based Weighting</th>
</tr>
</thead>
<tbody>
<tr>
<td>mse std</td>
<td>mse std</td>
<td>mse std</td>
<td>mse std</td>
</tr>
<tr>
<td>RF</td>
<td>10</td>
<td>9</td>
<td>7</td>
</tr>
<tr>
<td>AF</td>
<td>9</td>
<td>14</td>
<td>8</td>
</tr>
</tbody>
</table>

Table 1: Qualitative results (pixels) for 30 sequences and all weighting techniques.

<table>
<thead>
<tr>
<th>Uniform Weights</th>
<th>Texture Weighting</th>
<th>One-step Dist. Weighting</th>
<th>Hist. Based Weighting</th>
</tr>
</thead>
<tbody>
<tr>
<td>mse std</td>
<td>mse std</td>
<td>mse std</td>
<td>mse std</td>
</tr>
<tr>
<td>RF</td>
<td>76.7 %</td>
<td>90 %</td>
<td>83.3 %</td>
</tr>
<tr>
<td>AF</td>
<td>43.3 %</td>
<td>73.5 %</td>
<td>66.7 %</td>
</tr>
</tbody>
</table>

Table 2: The influence of the weight assignment techniques on the success rate.

4 MODEL BASED 3D TRACKING

A typical model based tracking system usually involves the following steps: detection, matching, pose estimation, update and prediction of the state used to render the model of the object into the image (see Fig. 6).

![Figure 6: Block diagram of our model based tracking system.](image)

The input to the algorithm is a wire–frame model of the object. This model is used during the initialization step where the initial pose the object relative to the camera coordinate system is estimated (explained in detail in Section 5).

The main loop starts with a prediction step where the state of the object is predicted using the current state
estimate and a model. The system state vector consists of three parameters describing translation of the target, another three for orientation and an additional six for the velocities:

\[
x = [X, Y, Z, \phi, \psi, \gamma, \dot{X}, \dot{Y}, \dot{Z}, \dot{\phi}, \dot{\psi}, \dot{\gamma}]
\]  
(7)

where \( \phi, \psi \) and \( \gamma \) represent roll, pitch and yaw angles. The following piecewise constant white acceleration model is considered [1]:

\[
x_{k+1} = Fx_k + Gv_k, \quad z_k = Hx_k + w_k \tag{8}
\]

where \( v_k \) is a zero-mean white acceleration sequence, \( w_k \) is the measurement noise and

\[
F = \begin{bmatrix} t_{0,6} & \Delta T t_{0,6} \\ 0 & I_{6,6} \end{bmatrix}, \quad G = \begin{bmatrix} \Delta^2 T t_{0,6} \\ \Delta T t_{0,6} \end{bmatrix}, \quad H = [t_{0,6} | \theta] \tag{9}
\]

When a new 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 (projection and rendering step). 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 (detection step).

After that, a search is performed for the maximum discontinuity (nearby edge) in the intensity gradient along the normal direction to the edge. In each point along a visible edge, the perpendicular distance to the nearby edge is determined using a one-dimensional search. The search starts at the projected model point and the traversal continues simultaneously in opposite search directions until the first local maximum is found (matching).

After the normal displacements are available, the method proposed in [2] 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 [7]. A few images from a tracking sequence are shown in Fig. 7.

Finally, the calculated pose is input to the update step where an \( \alpha - \beta \) filter is used according to [1]:

\[
\hat{x}_{k+1|k} = F\hat{x}_k, \quad \hat{z}_{k+1|k} = H\hat{x}_{k+1|k}
\]

\[
\hat{x}_{k+1|k+1} = \hat{x}_{k+1|k} + W[z_{k+1} - \hat{z}_{k+1|k}]
\]

(10)

Here, the pose of the target is used as measurement rather than image features, as commonly used in the literature (see, for example, [15], [14]). An approach similar to the one presented here is taken in [11]. This approach simplifies the structure of the filter which facilitates a computationally more efficient implementation. In particular, the dimension of the matrix \( H \) does not depend on the number of matched features in each frame but it remains constant during the tracking sequence.

5 FROM 2D TO 3D

We have decided to integrate both appearance based and geometrical models to bridge the gap between 2D and 3D. Many similar systems use manual pose initialization where the correspondence between the model and object features is given by the user, (see [4] and [2]). Although there are systems where this step is performed automatically [14], [8] 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 and therefore not suited for matching approaches based on, for example, line features [10], [6], [11].

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 [9] 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.

![Figure 8: The small image shows the training image used to estimate the nearest pose of the object for the current image. Left) the initial pose overlaid on the current image, and right) the final pose obtained by local refinement method.](image-url)

6 CONCLUSIONS

We have presented 2D, image based and 3D, pose based tracking systems used during robotic manipulation tasks. The former systems use a voting based integration framework where correlation, motion, color and intensity variation are used in order achieve robustness. The latter system uses a wireframe model of the object in order to estimate its position and orientation relative to the robot so that the final grasping stage can easily be performed. In addition, both appearance based and model based approach are integrated to bridge the gap between the 2D and 3D approaches and automatically initialize the model based tracking system.

The future work will evaluate the system for a variety of grasping tasks.
Figure 7: first row) An example of tracking a package of raisins: a fairly textured object against a textured background. The estimated pose of the object is overlaid in white. During this experiment a 6mm lens was used and the object was at the distance of approximately 50cm from the camera, and second row) A moving camera and a static object show the ability of the system to cope with significant depth changes and perspective effects.

Figure 9: Training image used to estimate the initial pose (far left) followed by the intermediate images of the fitting step.

REFERENCES


