INSTICC Press, Setubal, Portugal, pages 169–176

# ACTIVE SENSING STRATEGIES FOR ROBOTIC PLATFORMS, WITH AN APPLICATION IN VISION-BASED GRIPPING

Benjamin Deutsch*, Frank Deinzer*, Matthias Zobel*
*Chair for Pattern Recognition, University of Erlangen-Nuremberg*
*91058 Erlangen, Germany*
{*deutsch, deinzer, zobel*}*@informatik.uni-erlangen.de*

Joachim Denzler
*Chair for Computer Vision, University of Passau*
*94030 Passau, Germany*
*denzler@fmi.uni-passau.de*

Key words: Service Robotics, Object Tracking, Zoom Planning, Object Recognition, Grip Planning

Abstract: We present a vision-based robotic system which uses a combination of several active sensing strategies to grip a free-standing small target object with an initially unknown position and orientation. The object position is determined and maintained with a probabilistic visual tracking system. The cameras on the robot contain a motorized zoom lens, allowing the focal lengths of the cameras to be adjusted during the approach. Our system uses an entropy-based approach to find the optimal zoom levels for reducing the uncertainty in the position estimation in real-time. The object can only be gripped efficiently from a few distinct directions, requiring the robot to first determine the pose of the object in a classification step, and then decide on the correct angle of approach in a grip planning step. The optimal angle is trained and selected using reinforcement learning, requiring no user-supplied knowledge about the object. The system is evaluated by comparing the experimental results to ground-truth information.

## 1 INTRODUCTION

This paper focuses on the task of visual tracking, classification and gripping of a free-standing object by a robot. Typically, vision-based robotic gripping applications (Smith and Papanikolopoulos, 1996), use a passive, non-adaptive vision system. We present a system which combines several active sensing strategies to improve the sensor input available to the robot, and allow the robot to choose the best approach direction. A quantitative evaluation of the robot's performance is achieved by comparing ground-truth information with experimental results.

Our robot (see figure 1) consists of a platform with a holonomic movement system, a stereo camera system mounted on a pan-tilt-unit on top of the platform, and a lift-like gripper on the front of the platform. The object tracking subsystem uses the stereo head to estimate the 3D position of the object relative to the robot; this information is used to continuously adapt the robot's movement and guide it to the target. Additionally, the two cameras possess motorized zoom lenses, allowing the cameras' zoom levels to be adjusted during tracking. Instead of a simple reactive approach (Tordoff and Murray, 2001), or a trained model-dependent behavior (Paletta and Pinz, 2000), the tracking system automatically calculates the optimal zoom level with an entropy-based information theoretical approach (Denzler and Brown, 2002).

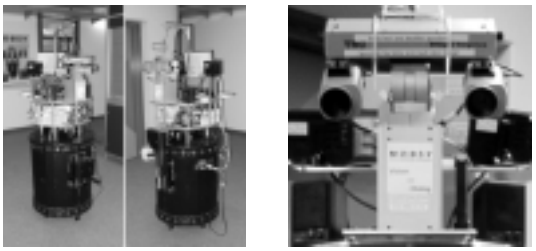

Figure 1: (left) Our robot, showing the stereo head and the gripper (holding a plastic bottle). (right) The stereo head.

Since the objects used may not be gripped from every angle (see figure 2 for examples of object orientation), the robot needs to detect the relative orientation of the object and adjust its approach accordingly. The object classifier generates both class and pose information, of which only the latter is used in this system.

The gripping planner uses the pose information to calculate the optimal gripping direction and the movement the robot must perform to approach the object

*This work was partly funded by the German Research Foundation (DFG) under grant SFB 603/TP B2. Only the authors are responsible for the content.

(a) (b) (c)

Figure 2: (a) An invalid gripping angle. (b) The worst still acceptable gripping angle. (c) The optimal gripping angle.

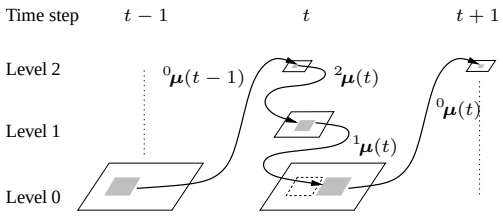


Figure 3: The hierarchical extension to the region tracker.

from this direction. This is related to active viewpoint selection for object recognition (Borotschnig et al., 2000). The viable grabbing positions were not input directly into the system; rather, the robot underwent a training phase, using reinforcement learning.

It should be noted that, unlike robot grip planning work as described in (Mason, 2001) or (Bicchi and Kumar, 2000), determining the optimal gripping position from the shape, outline or silhouette of the object is *not* a focus of our work. Instead, the optimal gripping position is learned during the training phase through feedback. This feedback could come from successful or unsuccessful gripping attempts, or even (as in our case) from a human judge. The robot can generalize from this training and evaluate new, untrained gripping positions.

The rest of this paper is organized as follows: Section 2 details the methods used for object tracking, object classification and grip planning. It also shows the interdependencies between the different subsystems. Section 3 contains some experimental results. It explains the exact setup that was used, the measurements taken, and evaluates the results. Section 4 concludes this paper, and outlines future enhancements which aim to enable the system to grip moving targets as well.

## 2 METHODS

This section details the methods used in our system. Section 2.1 describes the object tracker, section 2.2 the object classifier and grip planner, and section 2.3 the co-integration of the two.

### 2.1 Object Tracking

The purpose of the object tracking system is to determine the target object's location relative to the robot at all times. This information is critical for the robot's approach and the classification system (see section 2.2). The location is the 3D position of the target ($x$, $y$ and $z$ coordinates in mm) relative to the stereo head.

The vision system consists of two cameras on a pan-tilt unit with a vergence axis per camera (see figure 1 (right)). The pan axis is not used, leaving both cameras a shared tilt and individual vergence axes.

The object is first visually tracked in the camera images with two 2D template-based object trackers, based on a system presented by (Hager and Belhumeur, 1998). A detailed description of the tracker is beyond the scope of this paper; it is only necessary to note that the tracker generates a time-dependant motion parameter estimate $\boldsymbol{\mu}(t)$, describing the motion of the tracked object in the image. The estimate $\boldsymbol{\mu}(t-1)$ from the previous time step is used as a starting point for determining $\boldsymbol{\mu}(t)$. The tracking system on our robot uses motion parameters which describe only translations in the image plane $\boldsymbol{u}$ and a relative scaling factor $s$; these are sufficient for tracking a stationary object, given the planar robot movement.

One problem of the original implementation is that it can only handle small object motions. Too large motions will cause the trackers to lose the object. To allow the object to move farther in the camera image per time step, the tracker used in our system contains a hierarchical extension (Zobel et al., 2002).

An *image pyramid* is created by scaling the camera image and the reference template downwards $k-1$ times, creating $k$ levels of hierarchy. Typically, each level has half the resolution of the one below it.

Each of these $k$ levels then runs its own region based tracker, scaled appropriately. Whereas the non-hierarchical tracker uses the motion parameter estimate $\boldsymbol{\mu}(t-1)$ as an initial value while tracking the object from time step $t-1$ to time step $t$, the hierarchical tracker propagates the previous estimate through its different levels. The initial estimate is still the motion parameter vector ${}^0\boldsymbol{\mu}(t-1)$ from level 0 of the previous time step. However, it is now passed to the tracker operating on level $k-1$, the highest level, which results in a (rough) estimate ${}^{k-1}\boldsymbol{\mu}(t)$. This is then propagated down to level $k-2$ and so on, the last level receiving the estimate ${}^1\boldsymbol{\mu}(t)$ and calculating the estimate ${}^0\boldsymbol{\mu}(t)$. Figure 3 shows an example with 3 hierarchy levels.

Using this scheme, the complete tracker can handle larger movements, typically $2^{k-1}$ times as large, while still producing results just as accurate as the non-hierarchical version. Of course, care needs to be taken to scale the motion parameters between different levels; in our case of doubling resolutions, the translation parameter $\boldsymbol{u}$ needs to be divided by $2^{k-1}$

when going from ${}^0\boldsymbol{\mu}(t-1)$ to ${}^{k-1}\boldsymbol{\mu}(t)$, and multiplied by 2 for each descended level from there on. The scaling parameter $s$ is resolution independent and does not need to be adjusted.

In our system, we use $k = 3$ hierarchy levels, providing a good compromise between tracking capability and computation time requirements.

In order to combine, and smooth, the noisy 2D information provided by the two trackers, an extended Kalman filter (Bar-Shalom and Fortmann, 1988) is employed. The (extended) Kalman filter is a standard state estimation tool and will not be described here. A brief overview of the notation used here follows.

The object's true state at time $t$ is denoted by $\boldsymbol{q}_t$. This is a 9-dimensional vector comprised of the 3D position, velocity and acceleration of the object. For each time step, the filter receives an observation $\boldsymbol{o}_t$ (comprised, in our case, of four scalar values: the $x$ and $y$ image coordinates of the target centers in the two camera images) and incorporates this into its current belief. The filter's belief about the true state at time $t$ is a probability density function $p(\boldsymbol{q}_t|\langle\boldsymbol{o}\rangle_t)$, where $\langle\boldsymbol{o}\rangle_t$ denotes all observations received up to time $t$. In the case of the Kalman filter, this is assumed to be a normal distribution $\mathcal{N}(\hat{\boldsymbol{q}}_t^+, \boldsymbol{P}_t^+)$. The filter uses a *state transition model* to predict a new state estimate $p(\boldsymbol{q}_{t+1}|\boldsymbol{q}_t) = \mathcal{N}(\hat{\boldsymbol{q}}_{t+1}^-, \boldsymbol{P}_{t+1}^-)$ from the previous estimate. Upon receiving the next observation $\boldsymbol{o}_{t+1}$, the filter compares this to a predicted observation and updates its belief to $p(\boldsymbol{q}_{t+1}|\langle\boldsymbol{o}\rangle_{t+1})$.

During the tracking process, it is possible to adjust the focal lengths of the cameras. It is clear that such an adjustment will have an effect on the observation function used in the Kalman filter. This adds an *action parameter* $\boldsymbol{a}$ to the object state belief, giving $p(\boldsymbol{q}_t|\langle\boldsymbol{o}\rangle_t, \langle\boldsymbol{a}\rangle_t)$. The goal of the zoom planning subsystem is to find an action $\boldsymbol{a}^*$ (in our case two zoom levels, that is, the field-of-view of each camera) that is optimal for the Kalman filter, i.e. one that minimizes the *uncertainty* of the positional belief generated in the next time step.

In the Kalman filter, this uncertainty is described by the a-posteriori covariance matrix $\boldsymbol{P}^+$. The "larger" the covariance matrix is, the more likely the true state is deemed to be farther away from the estimated mean $\hat{\boldsymbol{q}}^+$. Several different measurements for the covariance matrix have been proposed in the context of estimation evaluation (Puckelsheim, 1993), such as the determinant of the matrix, or the inverse of the trail of its inverse (D- and A-Optimality, respectively).

Following (Denzler et al., 2003), we employ the *entropy* of the posterior distribution as an uncertainty measure (Shannon, 1948). The entropy of the a-posteriori density $p(\boldsymbol{q}_t|\langle\boldsymbol{o}\rangle_t, \langle\boldsymbol{a}\rangle_t)$ is defined as

$$H(\boldsymbol{q}) = -\int p(\boldsymbol{q}_t|\langle\boldsymbol{o}\rangle_t, \langle\boldsymbol{a}\rangle_t) \log p(\boldsymbol{q}_t|\langle\boldsymbol{o}\rangle_t, \langle\boldsymbol{a}\rangle_t) d\boldsymbol{q}_t. \tag{1}$$

Since (for our system) $p(\boldsymbol{q}_t|\langle\boldsymbol{o}\rangle_t, \langle\boldsymbol{a}\rangle_t)$ is an $n$-dimensional normal distribution $\mathcal{N}(\hat{\boldsymbol{q}}^+, \boldsymbol{P}^+)$, the entropy can be calculated as

$$H(\boldsymbol{q}) = \frac{n}{2} + \frac{1}{2}\log\left((2\pi)^n|\boldsymbol{P}^+|\right) \tag{2}$$

Unfortunately, the correct entropy can only be calculated *after* the most recent observation $\boldsymbol{o}_t$ has taken place, yet we wish to choose an action $\boldsymbol{a}_t$ based on this entropy *before* the observation. What we need to calculate is the *conditional entropy*

$$H(\boldsymbol{q}_t|\boldsymbol{o}_t, \boldsymbol{a}_t) = \int p(\boldsymbol{o}_t|\boldsymbol{a}_t) H(\boldsymbol{q}_t) d\boldsymbol{o}_t. \tag{3}$$

This is, in effect, the *expected* entropy of the belief over all observations $\boldsymbol{o}_t$, given the action $\boldsymbol{a}_t$. In the case of the Kalman filter, $\boldsymbol{P}^+$ is independent of the actual observation $\boldsymbol{o}_t$. Using this information and removing all terms which are irrelevant to the optimization, we obtain

$$\boldsymbol{a}_t^* = \arg\min_{\boldsymbol{a}_t} |\boldsymbol{P}_t^+| \tag{4}$$

There is one problem with this approach, however: this formula assumes that there will always be an observation, no matter how large the focal length. This is clearly not always the case. One of the main problems with a large focal length is the associated small field of view. A larger focal length increases the chance that the object's projection will in fact be outside of a camera's sensor. In this case, no (usable) observation has occurred, and the best estimate for the object's current state is the a-priori state estimate $p(\boldsymbol{q}_t|\langle\boldsymbol{q}\rangle_{t-1}, \langle\boldsymbol{o}\rangle_{t-1}) = \mathcal{N}(\hat{\boldsymbol{q}}_t^-, \boldsymbol{P}_t^-)$.

Splitting the conditional entropy into successful and unsuccessful observations, an unsuccessful observation being an observation which lies outside one of the camera's sensors, one can define $w_1(\boldsymbol{a}_t)$ to be the chance that the object will be visible and the observation will be successful, and $w_2(\boldsymbol{a}_t)$ as the chance that the object will not be visible (Denzler et al., 2003). $w_1(\boldsymbol{a}_t)$ and $w_2(\boldsymbol{a}_t)$ can be estimated using Monte Carlo sampling, or by closed-form evaluation in the case of normal distributions.

Although any irrelevant terms for the optimization can still be eliminated, the logarithms can no longer be avoided. This leaves the optimization problem

$$\begin{aligned}\boldsymbol{a}_t^* \quad = \quad & \arg\min_{\boldsymbol{a}_t} \Big( w_1(\boldsymbol{a}_t) \log\left(|\boldsymbol{P}_t^+(\boldsymbol{a}_t)|\right) \\ & + w_2(\boldsymbol{a}_t) \log\left(|\boldsymbol{P}_t^-|\right) \Big)\end{aligned} \tag{5}$$

## 2.2 Object Classification — Grip Planning

One of the goals of this work is to provide a solution to the problem of selecting an optimal grippoint resp.

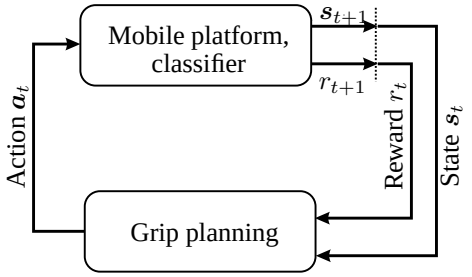


Figure 4: Principles of Reinforcement Learning applied to grip planning.

grip positions without making a priori assumptions about the objects and the classifier used to recognize the class and pose of the object. The problem is to determine the next view of an object given the current observations. The problem can also be seen as the determination of a function which maps an observation to a new grippoint. This function should be estimated automatically during a training step and should further improve over time. The estimation must be done by defining a criterion, which measures how good it is to grip an object from a specific position. Additionally, the function should take uncertainty into account in the recognition process as well as in the grippoint selection. The latter one is important, since the robot must move around the object to reach the planned grippoint; a noisy operation. So the final position of the robot will always be error-prone. Last not least, the function should be classifier independent and be able to handle continuous grippoints.

A straight forward and intuitive way to formalizing the problem is given by looking at figure 4. A closed loop between sensing $\boldsymbol{s}_t$ and acting $\boldsymbol{a}_t$ can be seen. The chosen *action*

$$\boldsymbol{a}_t = (\Delta\varphi) \quad \text{with} \quad \Delta\varphi \in [0°; 360°[ \tag{6}$$

corresponds to the movement of the mobile platform. As it will only move on a circle around the object in the application presented in this paper, the definition of (6) is sufficient. The sensed *state*

$$\boldsymbol{s}_t = (\Omega_\kappa, \varphi)^T \quad \text{with} \quad \varphi \in [0°; 360°[ \tag{7}$$

contains the class $\Omega_\kappa$ and pose $\varphi$ (the rotation) of the object relative to the robot. This state is estimated by the employed classifier. In this paper we use a wavelet-based classifier as described in (Grzegorzek et al., 2003) but other classification approaches can be applied. Additionally, a so called *reward* $r_t$, which measures the quality of the chosen grippoint is required. The better the chosen direction for gripping an object, the higher the yielding reward has to be. In our case we decided for $r_t \in [0; \text{MAX}]$, $\text{MAX} = 10$ with $r_t = 0$ for the worst grip position (figure 2(a)) and $r_t = 10$ for the best grip position (figure 2(c)). It is important to notice that the reward should also include costs for the robot movement, so that large movements of the robot are punished. These costs

$$\text{cost}(\boldsymbol{a}) = \begin{cases} m \cdot \text{MAX} \cdot \frac{\Delta\varphi}{360} & \Delta\varphi \leq 180 \\ m \cdot \text{MAX} \cdot \frac{360-\Delta\varphi}{360} & \Delta\varphi > 180 \end{cases} \tag{8}$$

with $m \in [0; 1]$ are subtracted from each reward.

At time $t$ during the decision process, the goal will be to maximize the accumulated and weighted future rewards, called the *return*

$$R_t = \sum_{n=0}^{\infty} \gamma^n r_{t+n+1} \tag{9}$$

with $\gamma \in [0; 1]$. The weight $\gamma$ defines how much influence a future reward will have on the overall return $R_t$ at time $t + n + 1$. For the application of selecting the optimal grip position $\gamma = 0$ is sufficient as only one step is necessary to reach the goal, the optimal grip position.

Of course, the future rewards cannot be observed at time step $t$. Thus, the following function, called the *action-value function*

$$Q(\boldsymbol{s}, \boldsymbol{a}) = E\{R_t | \boldsymbol{s}_t = \boldsymbol{s}, \boldsymbol{a}_t = \boldsymbol{a}\} \tag{10}$$

is defined, which describes the expected return when starting at time step $t$ in state $\boldsymbol{s}$ with action $\boldsymbol{a}$. In other words, the function $Q(\boldsymbol{s}, \boldsymbol{a})$ models the expected quality of the chosen movement $\boldsymbol{a}$ for the future, if the classifier has returned $\boldsymbol{s}$ before.

Viewpoint selection can now be defined as a two step approach: First, estimate the function $Q(\boldsymbol{s}, \boldsymbol{a})$ during training. Second, if at any time the classifier returns $\boldsymbol{s}$ as classification result, select that camera movement which maximizes the expected accumulated and weighted rewards. This function is called the *policy*

$$\pi(\boldsymbol{s}) = \text{argmax}_{\boldsymbol{a}}\, Q(\boldsymbol{s}, \boldsymbol{a})\ . \tag{11}$$

The key issue of course is the estimation of the function $Q(\boldsymbol{s}, \boldsymbol{a})$, which is the basis for the decision process in (11). One of the demands of this paper is that the selection of the most promising grip position should be learned without user interaction. Reinforcement learning provides many different algorithms to estimate the action-value function based on a trial and error methods (Sutton and Barto, 1998). Trial and error means that the system itself is responsible for trying certain actions in a certain state. The result of such a trial is then used to update $Q(\cdot, \cdot)$ and to improve its policy $\pi$.

In reinforcement learning a series of *episodes* are performed: Each episode $k$ consists of a sequence of state/action pairs $(\boldsymbol{s}_t, \boldsymbol{a}_t), t \in \{0, 1, \ldots, T\}$, where the performed action $\boldsymbol{a}_t$ in state $s_t$ results in a new state $\boldsymbol{s}_{t+1}$. A final state $\boldsymbol{s}_T$ is called the terminal state, where a predefined goal is reached and the episode ends. In our case, the terminal state is the state where

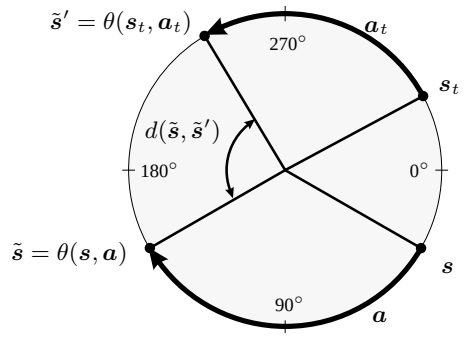


Figure 5: Illustration of the transformation function $\theta(\boldsymbol{s}, \boldsymbol{a})$ and the distance function $d(\cdot, \cdot)$.

gripping an object is possible with high confidence. During the episode, new returns $R_t^{(k)}$ are collected for those state/action pairs $(\boldsymbol{s}_t^k, \boldsymbol{a}_t^k)$ which were visited at time $t$ during the episode $k$. At the end of the episode, the action-value function is updated. In our case, so called Monte Carlo learning is applied, and the function $Q(\cdot, \cdot)$ is estimated by the mean of all collected returns $R_t^{(i)}$ for the state/action pair $(\boldsymbol{s}, \boldsymbol{a})$ for all episodes. Please note that is sufficient for the scope of this paper to restrict an episiode to only one chosen action. Longer episodes have been discussed for more complicated problems of *viewpoint selection* in (Deinzer et al., 2003).

As a result for the next episode one gets a new decision rule $\pi_{k+1}$, which is now computed by maximizing the updated action value function. This procedure is repeated until $\pi_{k+1}$ converges to the optimal policy. The reader is referred to a detailed introduction to reinforcement learning (Sutton and Barto, 1998) for a description of other ways for estimating the function $Q(\cdot, \cdot)$. Convergence proofs for several algorithms can be found in (Bertsekas, 1995).

Most of the algorithms in reinforcement learning treat the states and actions as discrete variables. Of course, in grippoint selection parts of the state space (the pose of the object) and the action space (the camera movements) are continuous. A way to extend the algorithms to continuous reinforcement learning is to approximate the action-value function

$$\widehat{Q}(\boldsymbol{s}, \boldsymbol{a}) = \frac{\sum\limits_{(\boldsymbol{s}', \boldsymbol{a}')} K\left(d\left(\theta(\boldsymbol{s}, \boldsymbol{a}), \theta(\boldsymbol{s}', \boldsymbol{a}')\right)\right) \cdot Q(\boldsymbol{s}', \boldsymbol{a}')}{\sum\limits_{(\boldsymbol{s}', \boldsymbol{a}')} K\left(d\left(\theta(\boldsymbol{s}, \boldsymbol{a}), \theta(\boldsymbol{s}', \boldsymbol{a}')\right)\right)} \tag{12}$$

which can be evaluated for any continuous state/action pair $(\boldsymbol{s}, \boldsymbol{a})$. Basically, this is a weighted sum of the action-values $Q(\boldsymbol{s}', \boldsymbol{a}')$ of all previously collected state/action pairs $(\boldsymbol{s}', \boldsymbol{a}')$. The other components within (12) are:

- The *transformation function* $\theta(\boldsymbol{s}, \boldsymbol{a})$ (see figure 5) transforms a state $\boldsymbol{s}$ with a known action $\boldsymbol{a}$ with the intention of bringing a state to a "reference point" (required for the distance function in the next item). In the context of the current definition of the states from (7) it can be seen as a "shift" of the state:

$$\begin{aligned} \theta(\boldsymbol{s}, \boldsymbol{a}) = \boldsymbol{s} + \begin{pmatrix} 0 \\ \boldsymbol{a} \end{pmatrix} &= \underbrace{\begin{pmatrix} \Omega_\kappa \\ \varphi \end{pmatrix}}_{\boldsymbol{s}} + \underbrace{\begin{pmatrix} 0 \\ \Delta\varphi \end{pmatrix}}_{\text{contains } \boldsymbol{a}} \\ &= \begin{pmatrix} \Omega_\kappa \\ (\varphi + \Delta\varphi) \mod 360 \end{pmatrix} \end{aligned} \tag{13}$$

- A *distance function* $d(\cdot, \cdot)$ (see figure 5) to calculate the distance between two states. Generally speaking, similar states must result in low distances. The lower the distance, the more transferable the information from a learned action-value to the current situation is. As one has to compare two states, the following formula meets the requirements:

$$\begin{aligned} d(\boldsymbol{s}, \boldsymbol{s}') &= d\left(\begin{pmatrix} \Omega_\kappa \\ \varphi \end{pmatrix}, \begin{pmatrix} \Omega_\lambda \\ \varphi' \end{pmatrix}\right) \\ &= \begin{cases} |\varphi - \varphi'| & \text{for } \Omega_\kappa = \Omega_\lambda \\ \infty & \text{otherwise} \end{cases} \end{aligned} \tag{14}$$

- A *kernel function* $K(\cdot)$ that weights the calculated distances. A suitable kernel function is the Gaussian $K(x) = \exp(-x^2/D^2)$, where $D$ denotes the width of the kernel.

Viewpoint selection, i.e. the computation of the policy $\pi$, can now be written, according to (11), as the optimization problem

$$\pi(\boldsymbol{s}) = \operatorname*{argmax}_{\boldsymbol{a}} \widehat{Q}(\boldsymbol{s}, \boldsymbol{a})\,. \tag{15}$$

## 2.3 Combined Execution

There are several points that arise from the fact that the object tracking and the object classification systems make use of the same cameras, at the same time. The tracking system needs to keep continuous track of the target, so the classifier must use the same camera settings and images.

During the classification process, the object tracking still calculates the optimal zoom level for tracking purposes. However, the classification process performs best when the camera is at the same zoom level as during the classifier's training. This conflict is resolved by augmenting the zoom level optimization system (5) with a weighting function[1] $w_c(\boldsymbol{a}_t)$:

$$\begin{aligned} \boldsymbol{a}_t^* \quad = \quad & \arg\min_{\boldsymbol{a}_t} \Big( w_1(\boldsymbol{a}_t) \log\left(|\boldsymbol{P}_t^+(\boldsymbol{a}_t)|\right) \\ & + w_2(\boldsymbol{a}_t) \log\left(|\boldsymbol{P}_t^-|\right) \Big) \cdot w_c(\boldsymbol{a}_t) \end{aligned} \tag{16}$$

[1] Inherent bounds on their arguments prevent the logarithms from ever becoming negative.

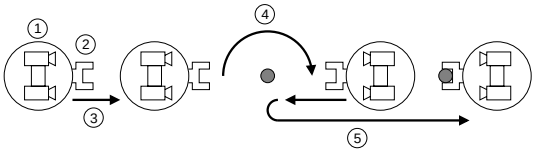


Figure 6: Overview of the experimental setup. See the text for a description of the individual steps.

$w_c(\boldsymbol{a}_t) \equiv 1$ unless an object classification needs to be performed, in which case it becomes very large (1000) for actions in which the left camera zoom level diverges from the classifier's preferred level by more than a fixed tolerance (the object classification uses only the image from the left camera). The same optimization system is then always used for the zoom levels, independent of the task the robot is currently performing.

Another restriction of the pose estimation process is that it was only trained for one degree of freedom, namely the rotation. This means that the robot must move to a fixed distance from the target in order to eliminate any tilting rotation. The object tracking system needs to provide a reliable depth estimate for the target at all times; merely detecting that the object has been reached is insufficient.

# 3 EXPERIMENTS

For experimental evaluation of the system, the gripping task was repeatedly performed with different robot starting locations and object orientations. The final position of the robot when gripping the object is used to evaluate the object tracker, while the orientation of the robot relative to the object assesses the classifier and grip planner.

## 3.1 Setup

The experiments were performed as shown in figure 6. The robot is placed a random distance from the object. The object is always at a fixed height. After the target object is selected, its 3D coordinates are tracked (1). The robot then orients itself to the target (2) and begins to move towards it (3). At a fixed distance, the robot stops to perform its classification and pose detection.

Once the pose is known, the robot moves around in a circular path (4) by the angle determined by the the grip planner. Then it moves towards the target until it is close enough for the object to be gripped (5). This final distance is still determined by visual tracking of the target only, no proximity or tactile sensors are used. After the robot has gripped the object, it lifts it from its stand, moves back a short distance and places the object on the floor.

The object classification was trained by the use of a turntable placed in front of the robot, and capturing the objects through the robot's left camera at different angles and lightings, as in (Grzegorzek et al., 2003). The distance from the robot to the objects was constant, resulting in a single degree of freedom and the object tracking requirements mentioned in section 2.3. The grip planning was trained by placing the robot in front of the object. The robot classified the object, then performed a random action, i.e. it moved in a circle around the object by a random amount. A human operator then rated the action between 0 (bad) and 10 (good) for the reinforcement learning (see section 2.2). This rating was subjective and therefore unstable; reinforcement learning can cope with such input, however. The object tracking requires no training of any kind.

## 3.2 Measurements and Evaluation

Twelve different series of experiments were performed, each series consisting of at least 10 individual experiments as described above (if the object tracker lost the object during phase 3, an experiment was repeated). For each new series, a different robot starting position and orientation was chosen. The robot was returned to approximately this starting position at the beginning of each experiment in a series. The first six series were performed with zoom planning disabled, and the second six series used zoom planning.

A total of 133 experiments were conducted. In 13 cases, the object tracking system lost the object at the beginning of the experiment, where the target is very small in the camera image. In 10 cases, the object was lost near the end of the experiment, where the target is viewed increasingly from above, causing its appearance to diverge from the original template (these experiments still yielded orientation measurements). In 105 cases, the robot's final orientation towards the target was within the valid grip position limits (see figure 2(b)), and in 15 cases, outside of them.

Before each experiment, the ground truth position and pose of the target was calculated using a calibration pattern placed at the bottom of the target's stand. The calibration pattern indicates the pose of the stand itself, while a rotational scale affixed to the bottom of the target shows its orientation relative to the stand (see figure 7). The calibration pattern is removed before the experiment starts; the robot does not have access to the calibration results.

For each experiment, the target position estimate is acquired during a 10 second pause after the robot has finished orienting itself towards the object, but before it starts moving closer. All position estimates during this time (about 55–60 measurements) are averaged to obtain the position estimate for one experiment. Since the self-orientation of the robot towards the target at

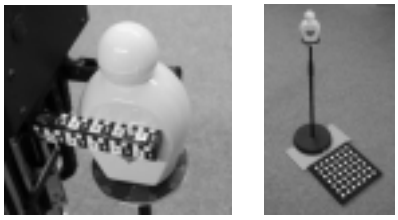

Figure 7: (left) The gripper with the affixed scale, and the rotational scale on the target. In this example, the grip position is 72 mm. (right) The stand with the calibration pattern.

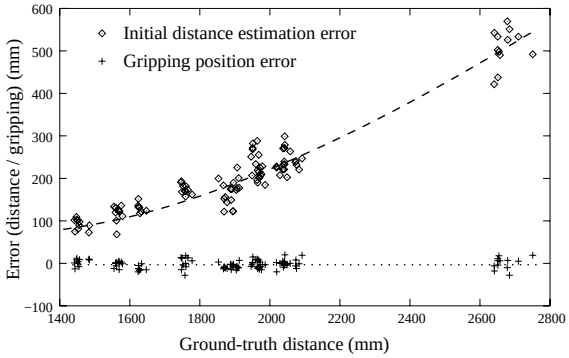


Figure 8: Evaluation of the distance estimation error and final gripping position as a function of the ground-truth distance. As the distance increases, the estimation error increases. However, the gripping position error remains constant. All values are in mm.

the beginning of each experiment is rather accurate, only the distance estimation from the robot to the target is evaluated, and compared to the ground-truth distance obtained from the calibration pattern.

The final gripping position is determined with the use of a scale attached to the robot's gripper (figure 7). The gripping position is defined as the length the target intrudes into the gripper at the farthest point the gripper still touches the target. This is compared to the ideal gripping position of 80 mm.

Figure 8 shows the distance estimation error, plotted against the ground-truth distance (positive values mean the distance was overestimated; this was the case in each experiment). Generally, the farther away the target is, the larger the error in the distance estimation becomes. At the same time, the final gripping position, as described above, is not affected by the starting position. The continuous fusion of new positional information by the Kalman filter allows the robot to recover from the inaccurate and uncertain initial position estimate.

In this application, since the robot moves slowly enough, the ability to change the zoom level has a negligible effect on the position estimation, compared to (for example) the influence of inaccuracies in the camera vergences. The main benefit of a flexible zoom level, for this task, occurs during the template selection scheme. Allowing the template matching to be performed at higher zoom levels allows one to increase the size of the template in the camera image, which increases the robustness of the tracking system. If the robot starts sufficiently far away, using the fixed minimal zoom level, the template covering the target's head is about $16 \times 16$ pixels. If the zoom level is increased prior to template matching, a larger template (such as $32 \times 32$ pixels) can be fitted over the same target region.

The robot will typically remain at these high zoom levels at the beginning, gradually reducing the zoom as it approaches the bottle. This greatly reduces the chance that the 2D trackers lose the object near the beginning of an experiment; early object loss occurred in 11 out of 71 experiments with fixed zoom, but only in 2 out of 62 experiments with a variable zoom, a reduction of 79%.

To evaluate the object classification, the pose estimate of the target was compared to the ground truth pose as calculated above to obtain the pose classification error. The pose classification error turned out to be unbiased (near zero mean). The 90th, 75th and 50th percentile of the absolute error is 18.7, 12.5 and 7 degrees, respectively. This compares very favorably to the cutoff error for acceptance of 20 degrees, as shown in figure 2(b).

For the evaluation of the grip planning, the pose estimation from the classifier is added to the action (movement in degrees) proposed by the grip planner, for each experiment. This resulting grip angle is compared to the ideal gripping angles (for this target) of 90 and 270 degrees. In our experiments, the planning error has a mean of -1.52 degrees, with a standard deviation of 1.36 degrees. This bias is the result of the cost function applied to the action selection in section 2.2, as shown in figure 9. Since the target is equally grippable from two locations, the gripping system chooses the closer one (requiring less movement) by weighting the action ratings as in equation (8). A side effect of this weighting is that the modes of the rating, too, get shifted slightly to favor less movement. This shift, however, is negligible when compared to the pose estimation error.

Finally, the actual gripping angle is evaluated. This is the pose of the target relative to the robot at step (5), and is measured externally by use of the scale affixed to the target. This angle is compared to the ideal gripping angles (90 and 270 degrees in our case) to obtain the final grip position error. The deviation was again unbiased (near zero mean), with the 90th, 75th and 50th percentile at 23, 15 and 9 degrees respectively. As demonstrated in figure 2(b), an absolute error below 20 degrees was deemed "acceptable", while anything above was "not acceptable". Out of 120 experiments which resulted in grippoint selections, only

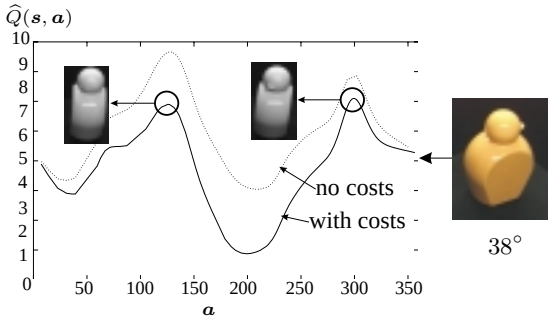


Figure 9: Grippoint selection incorporating costs. The target is estimated to be at 38 degrees. The influence of the cost function on the rating function is clearly visible.

15 were not acceptable, as the result of an incorrect pose estimation by the classifier.

# 4 CONCLUSION AND OUTLOOK

In this paper, we have presented the combination of two systems, an object tracker and an object classifier, which are able to grip a non-trivial object using only visual feedback.

An important aspect is that neither of these systems require any explicit modeling, neither in the behavior of the focal length adjustment, nor in the selection of the gripping angle. Instead, the focal length adjustment comes automatically from the information theoretic approach, while the correct angle is trained, allowing the system to generate its own model.

Future work will focus on improving the individual components of this system, motivated by the goal of tracking and gripping a moving target. This comprises prediction of the target's position multiple steps into the future, automatic adaptation of tracking features to cope with visually changing objects, and evaluation of reinforcement learning techniques which allow learning an optimal sequence of actions.

# REFERENCES


Bar-Shalom, Y. and Fortmann, T. (1988). *Tracking and Data Association*. Academic Press, Boston, San Diego, New York.

Bertsekas, D. P. (1995). *Dynamic Programming and Optimal Control*. Athena Scientific, Belmont, Massachusetts. Volumes 1 and 2.

Bicchi, A. and Kumar, V. (2000). Robotic grasping and contact: A review. In *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, volume 1, pages 348–353, San Francisco.

Borotschnig, H., Paletta, L., Prantl, M., and Pinz, A. (2000). Appearance-based active object recognition. *Image and Vision Computing*, 18(9):715–727.

Deinzer, F., Denzler, J., and Niemann, H. (2003). Viewpoint Selection – Planning Optimal Sequences of Views for Object Recognition. In *Computer Analysis of Images and Patterns – CAIP 2003*, LNCS 2756, pages 65–73, Heidelberg. Springer.

Denzler, J. and Brown, C. (2002). Information Theoretic Sensor Data Selection for Active Object Recognition and State Estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2):145–157.

Denzler, J., Zobel, M., and Niemann, H. (2003). Information Theoretic Focal Length Selection for Real-Time Active 3-D Object Tracking. In *International Conference on Computer Vision*, pages 400–407, Nice, France. IEEE Computer Society Press.

Grzegorzek, M., Deinzer, F., Reinhold, M., Denzler, J., and Niemann, H. (2003). How Fusion of Multiple Views Can Improve Object Recognition in Real-World Environments. In *Vision, Modeling, and Visualization 2003*, pages 553–560, München. Aka GmbH, Berlin.

Hager, G. and Belhumeur, P. (1998). Efficient region tracking with parametric models of geometry and illumination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(10):1025–1039.

Mason, M. (2001). *Mechanics of Robotic Manipulation*. MIT Press. Intelligent Robotics and Autonomous Agents Series, ISBN 0-262-13396-2.

Paletta, L. and Pinz, A. (2000). Active Object Recognition by View Integration and Reinforcement Learning. *Robotics and Autonomous Systems*, 31(1–2):71–86.

Puckelsheim, F. (1993). *Optimal Design of Experiments*. Wiley Series in Probability and Mathematical Statistics. John Wiley & Sons, New York.

Shannon, C. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423,623–656.

Smith, C. and Papanikolopoulos, N. (1996). Vision-guided robotic grasping: Issues and experiments. In *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, pages 3203–3208.

Sutton, R. and Barto, A. (1998). *Reinforcement Learning*. A Bradford Book, Cambridge, London.

Tordoff, B. and Murray, D. (2001). Reactive Zoom Control while Tracking Using an Affine Camera. In *Proceedings of the 12th British Machine Vision Conference*, volume 1, pages 53–62.

Zobel, M., Denzler, J., and Niemann, H. (2002). Binocular 3-D Object Tracking with Varying Focal Lengths. In *Proceedings of the IASTED International Conference on Signal Processing, Pattern Recognition, and Application, Crete, Greece*, pages 325–330, Anaheim, Calgary, Zurich. ACTA Press.