Computer Vision - ACCV 2006
7th Asian Conference on Computer Vision, Hyderabad, India, January 13-16, 2006, Proceedings, Part I
Series: Lecture Notes in Computer Science , Vol. 3851
Volume package Computer Vision - ACCV 2006
Sublibrary: Image Processing, Computer Vision, Pattern Recognition, and Graphics
Narayanan, P.J.; Nayar, Shree K.; Shum, Heung-Yeung (Eds.)
2006, XXXI, 973 p., Softcover
ISBN-10: 3-540-31219-6
ISBN-13: 978-3-540-31219-2
Online version available

This item usually ships in 2-3 days.

$119.00

Print version
Recommend to others

E-content

Online Version Click here !

All books by these editors
Narayanan, P.J.
Nayar, Shree K.
Shum, Heung-Yeung

Related subjects
Artificial Intelligence
Computer Imaging, Graphics & Vision
Foundations of Computing

About this book

About this book

The two volume set LNCS 3851 and LNCS 3852 constitutes the refereed proceedings of the 7th Asian Conference on Computer Vision, ACCV 2006, held in Hyderabad, India in January 2006.

The volumes present together a total of 64 revised full papers and 128 revised posters papers selected from about 500 submissions. The papers are organized in topical sections on camera calibration, stereo and pose, texture, face recognition, variational methods, tracking, geometry and calibration, lighting and focus, in the first volume. The papers of the second volume cover topics as detection and applications, statistics and kernels, segmentation, geometry and statistics, signal processing, and video processing.

Written for:
Researchers and professionals

Keywords:
3D vision
bioinspired computing
classification
clustering
computer vision
contour tracking
face recognition
image analysis
image processing
image registration

image retrieval
image segmentation
level set method
machine learning
motion tracking
object detection
object recognition
pattern recognition
segmentation
signal processing
speech animation
texture analysis
video processing
watermarking

This publication :
Volume 2
Pages : 902-912

Manuscript

Aspects of Optimal Viewpoint Selection and Viewpoint Fusion

F. Deinzer[1⋆], J. Denzler[2], Ch. Derichs[1⋆], and H. Niemann[1]

[1] Chair for Pattern Recognition, University Erlangen-Nürnberg, 91058 Erlangen, Germany
{deinzer,derichs,niemann}@informatik.uni-erlangen.de
[2] Chair for Image Processing, University Jena, 07737 Jena, Germany
denzler@informatik.uni-jena.de

Abstract. In the past decades, most object recognition systems were based on passive approaches. But in the last few years a lot of research was done in the field of active object recognition. In this context, there are several unique problems to be solved, such as the fusion of views and the selection of an optimal next viewpoint.
In this paper we present an approach to solve the problem of choosing optimal views (viewpoint selection) and the fusion of these for an optimal 3D object recognition (viewpoint fusion). We formally define the selection of additional views as an optimization problem and we show how to use reinforcement learning for viewpoint training and selection in continuous state spaces without user interaction. In this context we focus on the modeling of the reinforcement learning reward. We also present an approach for the fusion of multiple views based on density propagation, and discuss the advantages and disadvantages of two approaches for the practical evaluation of these densities, namely Parzen estimation and density trees.

1 Introduction

The results of 3D object classification and localization depend strongly on the images which have been taken of the object. For difficult data sets, usually more than one view is necessary to decide reliably on a certain object class. Viewpoint selection tackles exactly the problem of finding a sequence of optimal views to increase classification and localization results by avoiding ambiguous views or by sequentially ruling out possible object hypotheses. The optimality is not only defined with respect to the recognition rate, but also with respect to the number of views necessary to get reliable results.

In this paper, we present an approach for viewpoint selection based on reinforcement learning. The approach shows some major benefits: First, the optimal sequence of views is learned automatically in a training step without any user interaction. Second, the approach performs a fusion of the generated views, where the fusion method does not depend on a special classifier. This makes it applicable for a very wide range of

⋆ This work was partially funded by DFG under grant SFB 603/TP B2. Only the authors are responsible for the content.

applications. Third, the possible viewpoints are continuous, so that a discretization of the viewpoint space is avoided.

Viewpoint selection has been investigated in the past in several applications. Examples are 3D reconstruction [1] or optimal segmentation of image data [2]. In object recognition, some active approaches have already been discussed as well. [3] plans the next view for a movable camera based on probabilistic reasoning. The active part is the selection of a certain area of the image for feature selection. The selected part is also called the receptive field [4]. Compared to our approach, no camera movement is performed, neither during training nor during testing. Thus, the modeling of viewpoints in continuous 3D space is also avoided. The work of [5] uses Bayesian networks to decide on the next view to be taken. But the approach is limited to special recognition algorithms and to certain types of objects, for which the Bayesian network has been manually constructed. In other words, the approach is not classifier independent and cannot be applied without user interaction. [6] showed that the optimal action is the one that maximizes the mutual information between the observation and the state to be estimated.

In section 2 we will show how the viewpoint fusion of multiple views can be done based on recursive density propagation in a continuous state space. Our reinforcement learning approach for viewpoint selection is presented in section 3. The experimental results in section 4 show that the presented approach is able to learn an optimal strategy for viewpoint selection that generates only the minimal number of images. The paper concludes with a summary and an outlook to future work in section 5.

2 Viewpoint Fusion

In active object recognition, a series of observed images $\langle \boldsymbol{f} \rangle_t = \boldsymbol{f}_t, \boldsymbol{f}_{t-1}, \ldots, \boldsymbol{f}_0$ of an object are given together with the camera movements $\langle \boldsymbol{a} \rangle_{t-1} = \boldsymbol{a}_{t-1}, \ldots, \boldsymbol{a}_0$ between these images. Based on these observations of images and movements, one wants to draw conclusions for a non-observable state $\boldsymbol{q}_t$ of the object. This state $\boldsymbol{q}_t$ must contain both the *discrete* class Ω_κ and the *continuous* pose $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_J)^T$ of the object, leading to the state definition $\boldsymbol{q}_t = (\Omega_\kappa, \phi_1^t, \ldots, \phi_J^t)^T$. Please note that most related work is either restricted to just handling the class [7] or does not even claim to work on continuous poses [8]. The actions $\boldsymbol{a}_t$ consist of the *relative* camera movement with J degrees of freedom, in the following written as $\boldsymbol{a}_t = (\Delta\phi_1^t, \ldots, \Delta\phi_J^t)$. Generally, disturbances of these actions by some kind of inaccuracy within the movement have to be taken into consideration.

In the context of a Bayesian approach, the knowledge on the object's state is given in form of the a posteriori density $p(\boldsymbol{q}_t | \langle \boldsymbol{f} \rangle_t, \langle \boldsymbol{a} \rangle_{t-1})$ and can be calculated from

$$p\left(\boldsymbol{q}_t | \langle \boldsymbol{f} \rangle_t, \langle \boldsymbol{a} \rangle_{t-1}\right) = \frac{p\left(\boldsymbol{q}_t, \langle \boldsymbol{f} \rangle_t, \langle \boldsymbol{a} \rangle_{t-1}\right)}{p\left(\langle \boldsymbol{f} \rangle_t, \langle \boldsymbol{a} \rangle_{t-1}\right)} \tag{1}$$

$$= \frac{p\left(\boldsymbol{f}_t | \boldsymbol{q}_t, \langle \boldsymbol{f} \rangle_{t-1}, \langle \boldsymbol{a} \rangle_{t-1}\right) p\left(\boldsymbol{q}_t | \langle \boldsymbol{f} \rangle_{t-1}, \langle \boldsymbol{a} \rangle_{t-1}\right)}{\underbrace{p\left(\boldsymbol{f}_t | \langle \boldsymbol{f} \rangle_{t-1}, \langle \boldsymbol{a} \rangle_{t-1}\right)}_{=k}} \tag{2}$$

$$= \frac{1}{k} \cdot p\left(\boldsymbol{f}_t | \boldsymbol{q}_t\right) p\left(\boldsymbol{q}_t | \langle \boldsymbol{f} \rangle_{t-1}, \langle \boldsymbol{a} \rangle_{t-1}\right) \tag{3}$$

$$= \frac{1}{k} \cdot p\left(\boldsymbol{f}_t|\boldsymbol{q}_t\right) \int p\left(\boldsymbol{q}_t|\boldsymbol{q}_{t-1}, \boldsymbol{a}_{t-1}\right) \cdot p\left(\boldsymbol{q}_{t-1}|\left\langle \boldsymbol{f}\right\rangle_{t-1}, \left\langle \boldsymbol{a}\right\rangle_{t-2}\right) d\boldsymbol{q}_{t-1} \tag{4}$$

(1) results directly from the definition of the conditional probability $p(A|B) = p(AB)/p(B)$ and the further steps to (2) from the multiplication theorem for probability densities. In (3) the Markov assumption $p(\boldsymbol{f}_t|\boldsymbol{q}_t, \left\langle \boldsymbol{f}\right\rangle_{t-1}, \left\langle \boldsymbol{a}\right\rangle_{t-1}) = p(\boldsymbol{f}_t|\boldsymbol{q}_t)$ is applied. The formulation of $p(\boldsymbol{q}_t|\left\langle \boldsymbol{f}\right\rangle_{t-1}, \left\langle \boldsymbol{a}\right\rangle_{t-1})$ as an integral in (4) results from the total probability theorem. Obviously the probability (4) depends only on the camera movement $\boldsymbol{a}_{t-1}$. The inaccuracy of $\boldsymbol{a}_{t-1}$ is modeled within the state transition component $p(\boldsymbol{q}_t|\boldsymbol{q}_{t-1}, \boldsymbol{a}_{t-1})$.

The classic approach for solving this recursive density propagation is the Kalman Filter. But in computer vision, the necessary assumptions for the Kalman Filter ($p(\boldsymbol{f}_t|\boldsymbol{q}_t)$ being normally distributed) are often not valid due to object ambiguities, sensor noise, occlusion, etc. This is a problem since it leads to a distribution which is not analytically computable. An approach for the complicated handling of such multimodal densities are the so called particle filters [9]. The basic idea is to approximate the a posteriori density by a set of weighted samples. In our approach we use the Condensation Algorithm [9] which uses a sample set $Y_t = \{y_t^1, \ldots, y_t^K\}$ to approximate the multimodal probability distribution $p(\boldsymbol{q}_t|\left\langle \boldsymbol{f}\right\rangle_t, \left\langle \boldsymbol{a}\right\rangle_{t-1})$ by K samples $y_t^i = \{\boldsymbol{x}_t^i, p_t^i\}$. Each sample y consists of the position $\boldsymbol{x} = (\Omega_\kappa, \phi_1, \ldots, \phi_J)$ within the state space and a sample weighting p with $\sum_i p_t^i = 1$.

The Condensation Algorithm starts with an initial sample set Y_0. The samples of this set are distributed uniformly over the state space in our application as we have no knowledge given about the objects before observing the first image. For the generation of a new sample set Y_t, K new samples y_t^i are

1. drawn from Y_{t-1} with probability proportional to the sample weightings.
2. propagated with the necessarily predetermined sample transition model $\boldsymbol{x}_t^i = \boldsymbol{x}_{t-1}^i + (0, r_1, \ldots, r_J)^T$ with $r_j \sim \mathcal{N}(\Delta\phi_j^t, \sigma_j)$ and the variance parameters of the Gaussian transition noise σ_j.
3. evaluated in the image by $p(\boldsymbol{f}_t|\boldsymbol{x}_t^i)$. This evaluation is performed by the classifier. The only requirement for the classifier that shall be used together with our fusion approach is its ability to evaluate this density. In this work we use a classifier based on the continuous statistical eigenspace approach as presented in [10]. Other classifiers have been proven to work as well with the presented fusion approach.

In the context of our viewpoint selection, the densities which are represented by sample sets have to be evaluated. The direct evaluation of them beneath the positions given by the individual samples is not possible. It is necessary to find a continuous representation of the density. This will be done in two different ways in this paper.
Parzen estimation: A common way to evaluate non-parametric densities is the Parzen estimation [11] which is calculated from a sample set Y by

$$p(\boldsymbol{q}_t|\left\langle \boldsymbol{f}\right\rangle_t, \left\langle \boldsymbol{a}\right\rangle_{t-1}) \approx \frac{1}{K} \sum\nolimits_{i=1}^{K} g_0\left(\boldsymbol{q}_t - \boldsymbol{x}_t^i\right) \quad , \tag{5}$$

with $g_0(\boldsymbol{v}) = \mathcal{N}(\boldsymbol{v}|\boldsymbol{\mu} = \boldsymbol{0}, \boldsymbol{\Sigma})$ denoting a windowing function. In this paper only a Gaussian window function is used. The choice of the mean vector $\boldsymbol{\mu} = \boldsymbol{0}$ is comprehensible as the difference $(\boldsymbol{q}_t - \boldsymbol{x}_t^i)$ in (5) results in zero-mean data. In contrast, the definition of the covariance matrix requires a careful consideration of methods like the

mean minimal distance of samples or the entropy-based approach of [12] and will be omitted in this paper. For a more detailed explanation on the theoretical background of the approximation of (1) by a sample set we refer to [9].

Density trees: Another way to evaluate the densities represented by the sample set are the so-called *density trees* [13]. They use a tree structure to transform the discrete samples into a continuous density. Each node of the density tree represents a hyper-rectangle in the state space over $\boldsymbol{q}$. A density is built by the repeated partitioning of the parameter space and refining the tree structure until a stop criterion is reached. A detailed description of that process is given in [13].

3 Viewpoint Selection

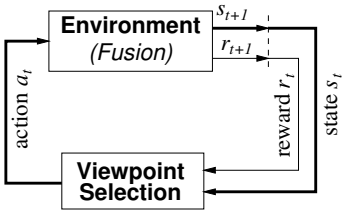

Fig. 1. Reinforcement learning

A straight forward and intuitive way to formalizing the problem is given by looking at Fig. 1. A closed loop between sensing s_t and acting $\boldsymbol{a}_t$ can be seen. The chosen *action* $\boldsymbol{a}_t$ corresponds to the executed camera movement, the sensed *state*

$$s_t = p(\boldsymbol{q}_t | \langle \boldsymbol{f} \rangle_t, \langle \boldsymbol{a} \rangle_{t-1}) \tag{6}$$

is the density as given in (1). Additionally, the classifier returns a so called *reward* r_t, which measures the quality of the chosen action resp. the resulting viewpoint. It is well known that the definition of the reward is an important aspect, as this reward should model the goal that has to be reached. Proper definitions for the reward in the context of our viewpoint selection problem are given later in this paper.

At time t during the decision process, i.e. the selection of a sequence of viewpoints, the goal will be to maximize the accumulated and weighted future rewards, called the *return*

$$R_t = \sum_{n=0}^{\infty} \gamma^n r_{t+n+1} \quad \text{with } \gamma \in [0;1],\ 0^0 =: 1\ . \tag{7}$$

The weight γ defines how much influence a future reward will have on the overall return R_t at time $t+n+1$. Of course, the future rewards cannot be observed at time step t. Thus, the following function, called the *action-value function* $Q(s, \boldsymbol{a}) = E\{R_t | s_t = s, \boldsymbol{a}_t = \boldsymbol{a}\}$ is defined, which describes the expected return when starting at an arbitrary time step t in state s with action $\boldsymbol{a}$. In other words, the function $Q(s, \boldsymbol{a})$ models the expected quality of the chosen camera movement $\boldsymbol{a}$ for the future, if the viewpoint fusion has returned s before.

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

$$\pi(s) = \underset{\boldsymbol{a}}{\operatorname{argmax}}\, Q(s, \boldsymbol{a})\ . \tag{8}$$

The key issue of course is the estimation of the function $Q(s, \boldsymbol{a})$, which is the basis for the decision process in (8). One of the demands defined in section 1 is that the

selection of the most promising view should be learned without user interaction. Reinforcement learning provides many different algorithms to estimate the action value function based on a trial and error method [14]. 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 π.

As a result for the next episode one gets a new decision rule π_{k+1}, which is now computed by maximizing the updated action value function. This procedure is repeated until π_{k+1} converges to the optimal policy. The reader is referred to a detailed introduction to reinforcement learning [14] for a description of other ways for estimating the function $Q(\cdot,\cdot)$. Convergence proofs for several algorithms can be found in [15].

We are still missing the definition of the reward r_t. In the context of viewpoint selection the following two different definitions of rewards make sense.

Fixed Value: A way to model the goal is to define a reward that has a value of 0 except when reaching the terminal state:

$$r_{t+1} = \begin{cases} C & s_t \text{ is terminal state, } C > 0 \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

This approach has the advantage that the goal is defined very clearly. But the environment has to decide when the confidence of the classification is high enough to stop the viewpoint selection. If this decision is hard to make, no proper strategy will be learned. The advantage is that (9) maximizes the return of an episode for short episodes (at least for $\gamma \neq 0, \gamma \neq 1$). So this strategy promises to look for episodes with only a minimal number of views. In our work we use $C = 1.0$.

Entropy Based: Another approach follows the idea that viewpoints that increase the information observed so far should have large values for the reward. A well-known measure for expressing the informational content that fits our requirements is the negative entropy $-H$, yielding

$$r_{t+1} = -H(s_t) = -H\left(p(\boldsymbol{q}_t | \langle \boldsymbol{f} \rangle_t, \langle \boldsymbol{a} \rangle_{t-1})\right) \quad . \tag{10}$$

In that sense the reward expresses the gain of knowledge about the object. (10) has the advantage that the goal is to improve the classification without only trying to reach a stop criterion. But it can not be made sure that maximizing the sum of entropies in (10) will always and under any circumstances lead to the absolutely shortest episodes.

Most of the algorithms in reinforcement learning treat the states and actions as discrete variables. Of course, in viewpoint 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

$$\widehat{Q}(s, \boldsymbol{a}) = \frac{\sum\limits_{(s', \boldsymbol{a}')} K\left(d\left(\theta(s, \boldsymbol{a}), \theta(s', \boldsymbol{a}')\right)\right) Q\left(s', \boldsymbol{a}'\right)}{\sum\limits_{(s', \boldsymbol{a}')} K\left(d\left(\theta(s, \boldsymbol{a}), \theta(s', \boldsymbol{a}')\right)\right)} \tag{11}$$

which can be evaluated for any continuous state/action pair $(s, \boldsymbol{a})$. Basically, this is a weighted sum of the action-values $Q\left(s', \boldsymbol{a}'\right)$ of all previously collected state/action pairs $(s', \boldsymbol{a}')$. The other components within (11) are:

The **transformation function** $\theta(s, \boldsymbol{a})$ transforms a state 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 (6) it can be seen as a density transformation

$$\begin{aligned}\theta(s_t, \boldsymbol{a}_t) &= \theta\left(p(\boldsymbol{q}_t|\left\langle \boldsymbol{f}\right\rangle_t, \left\langle \boldsymbol{a}\right\rangle_{t-1}), \boldsymbol{a}_t\right) \\ &= \det\left(\boldsymbol{J}_{\boldsymbol{\zeta}_{\boldsymbol{a}_t}^{-1}}(\boldsymbol{q}_t)\right) p\left(\boldsymbol{\zeta}_{\boldsymbol{a}_t}^{-1}(\boldsymbol{q}_t)|\left\langle \boldsymbol{f}\right\rangle_t, \left\langle \boldsymbol{a}\right\rangle_{t-1}\right)\end{aligned} \tag{12}$$

with $\boldsymbol{\zeta}_{\boldsymbol{a}}^{-1}(\boldsymbol{q}) = (q_1 + a_1, \ldots, q_m + a_m)^T$. It has been shown in [16] that the density transformation simply performs a shift of the density, so that $\boldsymbol{J}_{\boldsymbol{\zeta}_{\boldsymbol{a}}^{-1}}(\boldsymbol{q}) = \boldsymbol{I}$.
A **distance function** $d(\cdot,\cdot)$ is needed 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 the transformation function (12) results in a density, the *Kullback-Leibler Distance* $d_{KL}(s_n, s'_m)$ between the two states $s_n = p(\boldsymbol{q}|\left\langle \boldsymbol{f}\right\rangle_n, \left\langle \boldsymbol{a}\right\rangle_{n-1})$ and $s'_m = p(\boldsymbol{q}|\left\langle \boldsymbol{f}'\right\rangle_m, \left\langle \boldsymbol{a}'\right\rangle_{m-1})$, which can easily be extended to a symmetric distance measure, the so called *extended Kullback-Leibler Distance* $d_{EKL}(s_n, s'_m) = d_{KL}(s_n, s'_m) + d_{KL}(s'_m, s_n,)$, can be used. Please note that in general there is no analytic solution for the extended Kullback-Leibler Distance, but as we represent our densities as sample sets anyway (see section 2), there are well-known ways to approximate it by Monte Carlo techniques. The Monte Carlo techniques will use either the Parzen estimation or the density trees to evaluate the densities.
A **kernel function** $K(\cdot)$ 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. Low values for D will result in very detailed approximations provided that a lot of action-values $Q\left(s', \boldsymbol{a}'\right)$ are available.

Viewpoint selection, i.e. the computation of the policy π, can now be written, according to (8), as an optimization problem which is solved in this work by applying a global Adaptive Random Search Algorithm [17] followed by a local Simplex:

$$\pi(s) = \underset{\boldsymbol{a}}{\operatorname{argmax}}\, \widehat{Q}\left(s, \boldsymbol{a}\right) \quad . \tag{13}$$

4 Experimental Evaluation

Our primary goal in the experiments was to show that our approach is able to learn and perform an *optimal sequence* of views. We have shown in several publications (e.g. [18]) that the viewpoint fusion of a sequence of *randomly* chosen views works very well in real world environments and improves classification and localization result significantly. For that reason we decided to use the rather simple (from the pure object recognition's point of view) synthetic images of the two types of cups shown in Fig. 2 for the evaluation of our viewpoint selection approach. It was explicitly desired to have objects that can reach a 100% recognition rate given the optimal views.

The four cups of "type one" in the upper row of Fig. 2 show a number **1** or **2** on one, and a letter **A** or **B** on the other side. A differentiation between the 4 possible objects is only possible if number and letter have been observed and properly fused. The five cups of "type two" in the lower row of Fig. 2 show a number (**1 2 3 4 5**) on the front side. If this number is not visible the objects can not be distinguished or localized.

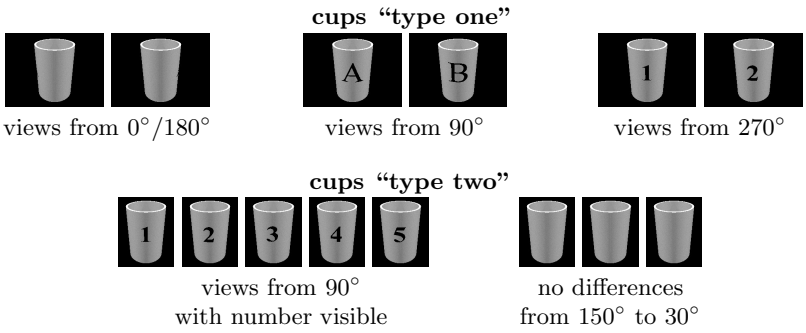

Fig. 2. Examples for objects that require viewpoint selection and fusion of images for proper recognition.

The cups can be classified correctly and stably within an area of about 120°. Localization of the cups is possible within an area of approximately 140°. In our setup the camera is restricted, for both types of cups, to a movement around the object on a circle, so that the definition of the samples reduces to $\boldsymbol{x} = (\Omega_\kappa, \phi_1)$ with actions $\boldsymbol{a}_t = (\Delta\phi_1^t)$, $\Delta\phi_1^t \in [0°, 360°]$. Our sample sets had size of $K = 1440$ (cups “type one”) resp. $K = 1800$ (cups “type two”) samples.

In our experiments we evaluated scenarios that differ in the way they evaluate the densities represented by our sample sets (Parzen estimation or density trees) and in the type of reward used (fixed value according to (9) or entropy-based as given in (10)), leaving the four variations r^{ep} (entropy-based reward, Parzen estimation), r^{ed} (entropy-based reward, density trees), r^{fp} (fixed value, Parzen estimation) and r^{fd} (fixed value, density trees). Additionally, three different values of the return parameter $\gamma \in \{0, 0.5, 1\}$ (see (7)) were used as they cover the two extreme values 0 and 1 which might have significant influence on the learned strategy and a value of 0.5 which represents the whole parameter range in-between. In a training step a total of 1000 episodes (with a maximal total length of 8 steps independent of the fact that the stop criterion was reached or not) were performed for every object, each value of $\gamma \in \{0, 0.5, 1\}$ and any of the variations r^{ep}, r^{ed}, r^{fp} and r^{fd}. The evaluation was performed on the results of a total of 1000 (for cups of “type one”) resp. 1250 (for cups of “type two”) episodes with randomly chosen classes and starting views.

In a first step we look at the recognition results of the viewpoint selection for the four variations r^{ep}, r^{ed}, r^{fp}, r^{fd} given values of $\gamma = 0.5$ and $D = 50$ in the kernel function $K(\cdot)$ for the approximation of the action-value function (11). As one can see in Fig. 3, the recognition results of the viewpoint selection reach a recognition rate of or close to 100%, as expected.

So the next question is if best viewpoints are selected in sense of the minimal numbers of views required. Number and letter are visible within the area stated above. Considering this, a theoretical minimum for the necessary mean sequence length exists:

- ≈ 2.2 views for the cups of “type one”. Two views are required if number or letter is initially visible, three views otherwise.
- ≈ 2.0 for the cups of “type two”. Depending on the strategy three to four views are required if number is not initially visible, one view otherwise. Anyhow, the

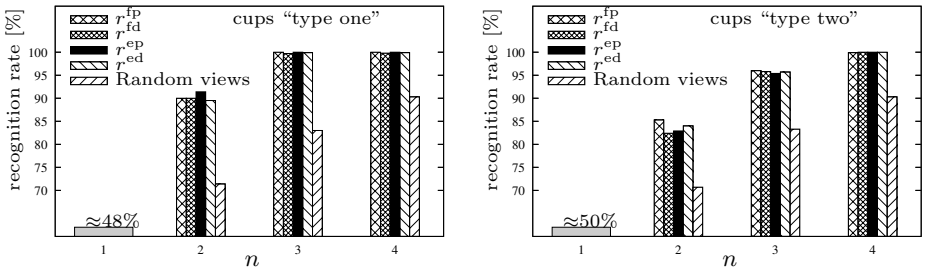

Fig. 3. Recognition rates of the viewpoint selection after planning n steps for the four different variations r^{ep}, r^{ed}, r^{fp}, r^{fd} compared to randomly chosen views. At step $n = 1$ all results are the same as no planning was done. These results compare to recognition rate that could be reached by pure passive recognition approaches. The parameters used for these results are $D = 50$ and $\gamma = 0.5$

theoretical minimum for the necessary mean sequence length can be shown to be always ≈ 2.0 steps.

Setting $D = 2$ since 1000 training episodes justify a detailed approximation and stopping when the probability of the best class is at least 95%, the mean number of views required to reach the stop criterion are summarized in Table 1. These numbers show that most configurations are very close to the theoretical minimum of required views. Exceptions are the variations r^{fp} and r^{fd} in combination with $\gamma = 1.0$. The reason can be found in the definition of the this reward in (9). Above we mentioned that the reward has to model the intended goal. This was done correctly in (9) but a reward of 0 means that if the end of the episode is not reached with the next step no "costs" are caused. In combination with $\gamma = 1$ this results in a total return according to (7) that is 1 independent of the length of the episode. In the sense of reinforcement learning, there is no need for the agent to look for short episodes. As one can see by means of the rightmost graph of the approximated action-value function in Fig. 4 no proper strategy was learned since all possible actions show nearly the same value. The small dents at 0° and 180° result from the limitations of the episode length to 8 steps (see above) that forces the system to learn at least a little bit of knowledge. This behavior could be changed in (9) if a negative value instead of 0 is returned. This could be seen as costs that force the agent to minimize the episode length. But the discussion of how to properly model costs in viewpoint selection is outside the scope of this paper.

Another observation from Table 1 is that the results for the variations that use the Parzen estimation for the evaluation of the densities $p(\boldsymbol{q}_t | \langle \boldsymbol{f} \rangle_t, \langle \boldsymbol{a} \rangle_{t-1})$ are better than the ones that use the density trees. The reason is obvious if one looks at the left and middle approximated action-value function in Fig. 4. The variations that use the Parzen estimation have a smoothly approximated action-value function. In contrast the approximations of the density trees are highly jagged. This is due to the nature of the density trees: They approximate densities by piecewise constant values, leading to densities that are not continuous.

The computational effort and the required memory resources for planning a new viewpoint is rather high. For the cups of "type one" one planning step, i.e. the evalu-

cups "type one"			
Vari-	D=2, γ=...		
ation	0	0.5	1
r^{fp}	2.17	2.18	2.40
r^{fd}	2.28	2.29	5.70
r^{ep}	2.19	2.17	2.20
r^{ed}	2.21	2.23	2.22

cups "type two"			
Vari-	D=2, γ=...		
ation	0	0.5	1
r^{fp}	2.06	2.00	2.15
r^{fd}	2.02	2.03	2.82
r^{ep}	2.06	2.01	2.05
r^{ed}	2.10	2.02	2.00

Table 1. Mean number of views needed to allow for a reliable classification for different system settings. Object recognition stopped when the probability of the best class reached at least 95%.

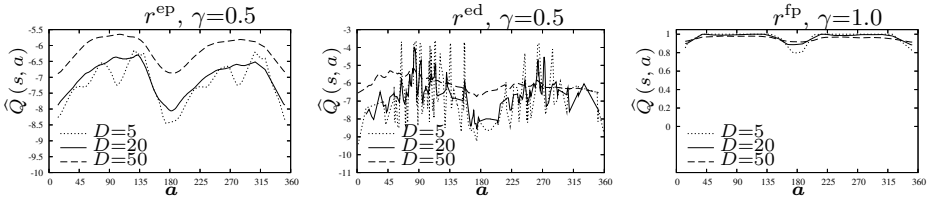

Fig. 4. Influence of D, γ and the type of reward and density evaluation to the approximation of the action-value function. All graphs show the estimated quality of the possible action for a current view of 0° to the cups of "type one". Graphs for $\gamma = 0$ are very similar to the ones with $\gamma = 0.5$ and omitted for that reason.

ation of (13), requires 550 to 750 evaluations of (11), each lasting ≈130ms. The 3670 action-values collected during the 1000 training episodes allocate 371 MB of memory if using the Parzen estimation and 96 MB for the density trees. The cups of "type two" require between 120 and 220 evaluations of (11) each lasting ≈150ms. The memory allocation for storing the 3160 action-values of the 1000 training episodes is 382 MB (Parzen estimation) resp. 116 MB (density trees). All numbers were evaluated on a Linux PC with a Xeon 2.80 GHz processor and 2 GB of main memory.

The conclusion of the experiments are that both types of introduced rewards lead to good planning results, at least for $\gamma \neq 1$. The necessary evaluation of the densities from the viewpoint fusion should be done with the Parzen estimation although the results of the density trees are better than the approximated action-value functions promise. If lack of memory is a problem the density tree variations might be an interesting alternative as they show huge memory saving compared to the Parzen estimation.

5 Summary and Future Work

In this paper we have presented the impact of several types of rewards and approaches for working with the densities given by the viewpoint fusion on the recognition rates of our general framework for viewpoint selection. We discussed several aspects of how to model the reward and the effects of different approaches for the evaluation of densities given as sample sets by the viewpoint fusion.

The viewpoint selection works in continuous state and action spaces and is independent of the chosen statistical classifier. Furthermore, the system can be trained

automatically without user interaction. The experimental results on two objects that require different strategies for recognition have shown that an optimal planning strategy was learned.

In our future work we will evaluate how much the planning of optimal view sequences improves object recognition rates on real world objects compared to the random strategy we used in [18]. Finally, for higher dimensional state spaces, other reinforcement learning methods might be necessary to reduce training complexity.

References

1. P. Lehel and E.E. Hemayed and A.A. Farag: Sensor Planning for a Trinocular Active Vision System. In: CVPR. (1999) II:306–312
2. Madsen, C., Christensen, H.: A Viewpoint Planning Strategy for Determining True Angles on Polyhedral Objects by Camera Alignment. PAMI **19** (1997)
3. Roy, S.D., Chaudhury, S., Banerjee, S.: Recognizing Large 3-D Objects through Next View Planning using an Uncalibrated Camera. In: ICCV 2001, Vancouver, Canada (2001) II: 276 – 281
4. Schiele, B., Crowley, J.: Transinformation for Active Object Recognition. In: ICCV 98, Bombay, India (1998) 249–254
5. Krebs, B., Burkhardt, M., Korn, B.: Handling Uncertainty in 3D Object Recognition using Bayesian Networks. In: ECCV 98, Berlin (1998) 782–795
6. Denzler, J., Brown, C.: Information theoretic sensor data selection for active object recognition and state estimation. PAMI **24** (2002)
7. Zhou, X., Comaniciu, D., Krishnan, A.: Conditional feature sensitivity: A unifying view on active recognition and feature selection. In: ICCV 03, Nice, France (2003)
8. Callari, G., P.Ferrie, F.: Active Object Recognition: Looking for Differences. International Journal of Computer Vision **43** (2001) 189–204
9. Isard, M., Andrew, B.: CONDENSATION — Conditional Density Propagation for Visual Tracking. IJCV 98 **29** (1998) 5–28
10. Gräßl, C., Deinzer, F., Mattern, F., Niemann, H.: Improving Statistical Object Recognition Approaches by a Parameterization of Normal Distributions. Pattern Recognition and Image Analysis **14** (2004) 222–230
11. Parzen, E.: On the estimation of a probability density function and mode. Annals of Mathematical Statistics **33** (1962) 1065–1076
12. Viola, P.: Alignment by Maximization of Mutual Information. PhD thesis, MIT, Department of Electrical Engineering and Computer Science, Cambridge, Massachusetts (1995)
13. Thrun, S., Langford, J.: Monte carlo hidden Markov models. Technical Report CMU-CS-98-179, Carnegie Mellon University, Computer Science Department, Pittsburgh, PA (1998)
14. Sutton, R., Barto, A.: Reinforcement Learning. A Bradford Book, Cambridge, London (1998)
15. Bertsekas, D., Tstsiklis, J.: Neuro–Dynamic Programming. Athena Scientific (1996)
16. Deinzer, F., Denzler, J., Niemann, H.: Viewpoint Selection – Planning Optimal Sequences of Views for Object Recognition. In: Computer Analysis of Images and Patterns. Volume 2756 of LNCS., Springer (2003) 65–73
17. Törn, A., Žilinskas, A.: Global Optimization. Volume 350 of Lecture Notes in Computer Science. Springer, Heidelberg (1987)
18. Deinzer, F., Denzler, J., Niemann, H.: On Fusion of Multiple Views for Active Object Recognition. In: DAGM 2001, Berlin, Springer (2001) 239–245