Viewpoint Selection – Planning Optimal Sequences of Views for Object Recognition

Frank Deinzer⋆, Joachim Denzler, and Heinrich Niemann

Chair for Pattern Recognition, Department of Computer Science,
University of Erlangen-Nürnberg, Martensstr. 3, 91058 Erlangen
{deinzer,denzler}@informatik.uni-erlangen.de
http://www.mustererkennung.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, like the fusion of several views and the selection of the best 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. We also present an approach for the fusion of multiple views based on recursive density propagation.
The experimental results show that our viewpoint selection is able to select a minimal number of views and perform an optimal object recognition with respect to the classification.

1 Introduction

The results of 3D object classification and localization depend strongly on the images which have been taken of the object. Based on ambiguities between objects in the data set some views might result in better recognition rates, others in worse. 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. The number of views should be as small as possible to delimit viewpoint selection from randomly taking a large number of images.

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, where no user interaction is necessary. 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 applications. Third, the possible viewpoints are continuous, so that a discretization of the viewpoint space is avoided, as has been done before, for example in the work of [2].

Viewpoint selection has been investigated in the past in several applications. Examples are 3D reconstruction [11] or optimal segmentation of image data [10].

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

In object recognition some active approaches have already been discussed as well. [12] 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 receptive field [13]. 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 [9] 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. [5] showed that the optimal action is the one that maximizes the mutual information between the observation and the state to be estimated.

In section 2.1 we will show how the fusion of multiple views can be done. We will present our approach for viewpoint selection in section 2.2. The experimental results in section 3 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 4.

2 Planning of view sequences

The goal of this work is to provide a solution to the problem of optimal viewpoint selection for 3D object recognition without making a priori assumptions about the objects and the classifier. The problem is to determine the next view of an object given a series of previous decisions and observations. The problem can also be seen as the determination of a function which maps a history of observations to a new viewpoint. This function should be estimated automatically during a training step and should improve over time. The estimation must be done by defining a criterion, which measures how useful it is to choose a certain view given a history of observations. Additionally, the function should take uncertainty into account in the recognition process as well as in the viewpoint selection. The latter one is important, since new views are usually taken by e.g. moving a robot arm or a mobile platform. So the final position of the robot arm or the platform will always be error-prone. Last not least, the function should be classifier independent and be able to handle continuous viewpoints.

The realization of the described problem can be separated into two major parts. First, as a sequence of views will be necessary to compute classification results, we have to be able to perform a fusion of several views. A way to solve this problem using particle filters is given in section 2.1. Second, the main task, the planning of view sequences, must be properly formulated. An approach based on reinforcement learning [14] is presented in section 2.2

2.1 Fusion of Multiple Views by Density Propagation

In active object recognition a series of observed images $\boldsymbol{f}_t, \boldsymbol{f}_{t-1}, \ldots, \boldsymbol{f}_0$ of an object are given together with the camera movements $\boldsymbol{a}_{t-1}, \ldots, \boldsymbol{a}_0$ 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 and the *continuous* pose of the object. This fact is important for the following discussion.

In the context of a Bayesian approach, the knowledge on the object's state is given in form of the a posteriori density $p(\boldsymbol{q}_t|\boldsymbol{f}_t, \boldsymbol{a}_{t-1}, \boldsymbol{f}_{t-1}, \ldots, \boldsymbol{a}_0, \boldsymbol{f}_0)$ and can be calculated from

$$p(\boldsymbol{q}_t|\boldsymbol{f}_t, \boldsymbol{a}_{t-1}, \ldots, \boldsymbol{a}_0, \boldsymbol{f}_0) = \frac{1}{k_t} p(\boldsymbol{q}_t|\boldsymbol{a}_{t-1}, \boldsymbol{f}_{t-1}, \ldots, \boldsymbol{a}_0, \boldsymbol{f}_0) p(\boldsymbol{f}_t|\boldsymbol{q}_t) \tag{1}$$

where $k_t = p(\boldsymbol{f}_t, \boldsymbol{a}_{t-1}, \ldots, \boldsymbol{a}_0, \boldsymbol{f}_0)$ denotes a normalizing constant that is ignored in the following considerations. Under the Markov assumption for the state transition, (1) can be recursively rewritten as

$$p(\boldsymbol{q}_t|\boldsymbol{a}_{t-1}, \boldsymbol{f}_{t-1}, \ldots) = \int_{\boldsymbol{q}_{t-1}} p(\boldsymbol{q}_t|\boldsymbol{q}_{t-1}, \boldsymbol{a}_{t-1}) \cdot p(\boldsymbol{q}_{t-1}|\boldsymbol{a}_{t-1}, \boldsymbol{f}_{t-1}, \ldots) d\boldsymbol{q}_{t-1} \,. \tag{2}$$

Obviously this probability depends only on the camera movement $\boldsymbol{a}_{t-1}$. Its inaccuracy is modeled with a normally distributed noise component.

The classic approach for solving this recursive density propagation is the Kalman Filter [8]. 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. In real world applications this density $p(\boldsymbol{f}_t|\boldsymbol{q}_t)$ usually is not normally distributed 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 [7]. The basic idea is to approximate the a posteriori density by a set of weighted particles. In our approach we use the Condensation Algorithm [7]. It uses a sample set $C_t = \{\boldsymbol{c}_1^t, \ldots, \boldsymbol{c}_K^t\}$ to approximate the multimodal probability distribution in (1). Please note that we do not only have a continuous state space for $\boldsymbol{q}_t$ but a *mixed discrete/continuous state space* for object class and pose, as mentioned at the beginning of this section.

Now we will show how to use the Condensation Algorithm in a practical realization of sensor data fusion of multiple views. As noted above we need to include the class and pose of the object into our state $\boldsymbol{q}_t$ to classify and localize objects. This leads to the definitions of the state $\boldsymbol{q}_t = \left(\Omega_\kappa \; {}^1\varphi^t \ldots {}^J\varphi^t \right)^T$. The samples $\boldsymbol{c}$ and camera movements $\boldsymbol{a}$ are defined as

$$\boldsymbol{c}_i^t = \left(\Omega_{\kappa_i} \; {}^1\varphi_i^t \ldots {}^J\varphi_i^t \right)^T \quad \text{and} \quad \boldsymbol{a}_t = \left(\Delta^1\varphi^t \ldots \Delta^J\varphi^t \right)^T \tag{3}$$

with the class numbers Ω_κ and Ω_{κ_i}. ${}^j\varphi^t$ denotes the pose of the j-th degree of freedom for the camera position and $\Delta^j\varphi^t$ the relative changes of the viewing position of the camera.

In the practical realization of the Condensation Algorithm, one starts with an initial sample set $C^0 = \{\boldsymbol{c}_1^0, \ldots, \boldsymbol{c}_K^0\}$ with samples distributed uniformly over the state space. For the generation of a new sample set C^t, samples $\boldsymbol{c}_i^t$ are

1. drawn from C^{t-1} with probability $\frac{p(\boldsymbol{f}_{t-1}|\boldsymbol{c}_i^{t-1})}{\sum_{j=1}^K p(\boldsymbol{f}_{t-1}|\boldsymbol{c}_j^{t-1})}$
2. propagated with the necessarily predetermined sample transition model $\boldsymbol{c}_i^t = \boldsymbol{c}_i^{t-1} + \left(0 \; r_1 \ldots r_J \right)^T$ with $r_j \sim \mathcal{N}(\Delta^j\varphi^t, \sigma_j)$ and the variance parameters of the Gaussian transition noise σ_j. They model the inaccuracy of the camera movement under the assumption that the errors of the camera movements are independent between the degrees of freedom. These variance parameters have to be defined in advance.
3. evaluated in the image by $p(\boldsymbol{f}_t|\boldsymbol{c}_i^t)$. 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 the context of our viewpoint selections the densities represented by sample sets have to be evaluated. This can be done, for example, by a Parzen estimation over the sample set [15]. For a more detailed explanation on the theoretical background of the approximation of (1) by a sample set we refer to [7].

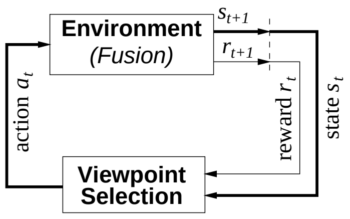

Fig. 1. Reinforcement learning

At this point we want to note that it is important to include the class Ω_κ in the object state $\boldsymbol{q}_t$ and the samples $\boldsymbol{c}_i^t$. An alternative would be to omit this by setting up several sample sets – one for each object class – and perform the Condensation Algorithm separately on each set. But this would not result in an integrated classification/localization, but in separated localizations on each set under the assumption of observing the corresponding object class. No fusion of the object class over the sequence of images would be done in that case.

2.2 Reinforcement Learning Applied to Viewpoint Selection

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*

$$s_t = p(\boldsymbol{q}_t|\boldsymbol{f}_t, \boldsymbol{a}_{t-1}, \boldsymbol{f}_{t-1}, \ldots, \boldsymbol{a}_0, \boldsymbol{f}_0) \tag{4}$$

is the density as given in (1). Additionally, the classifier returns a so called *reward* r_t, which measures the quality of the chosen viewpoint. For a viewpoint that increases the information observed so far the reward should have a large value. A well-know measure for expressing the informational content that fits our requirements is the entropy

$$r_{t+1} = -H(s_t) = -H\left(p(\boldsymbol{q}_t|\boldsymbol{f}_t, \boldsymbol{a}_{t-1}, \boldsymbol{f}_{t-1}, \ldots, \boldsymbol{a}_0, \boldsymbol{f}_0)\right) \tag{5}$$

It is worth noting that the reward might also include costs for the camera movement, so that large movements of the camera are punished. In this paper we neglect costs for camera movement for the time being.

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*

$$R_t = \sum_{n=0}^{\infty} \gamma^n r_{t+n+1} = -\sum_{n=0}^{\infty} \gamma^n H(s_{t+n+1}) \quad \text{with } \gamma \in [0;1]\,. \tag{6}$$

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* $Q(s, \boldsymbol{a})$

$$Q(s, \boldsymbol{a}) = E\left\{R_t|s_t = s, \boldsymbol{a}_t = \boldsymbol{a}\right\} \tag{7}$$

is defined, which describes the expected return when starting at 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 sensor 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 sensor fusion returns s as classification result, select that camera movement which maximizes the expected accumulated and weighted rewards. This function is called the *policy*

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

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.

In reinforcement learning a series of *episodes* are performed: Each episode k consists of a sequence of state/action pairs $(s_t, \boldsymbol{a}_t), t \in \{0, 1, \dots, T\}$, where the performed action $\boldsymbol{a}_t$ in state s_t results in a new state s_{t+1}. A final state 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 classification and localization is possible with high confidence. During the episode new returns $R_t^{(k)}$ are collected for those state/action pairs $(s_t^k, \boldsymbol{a}_t^k)$ which have been 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 $(s, \boldsymbol{a})$ for all episodes.

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 [1].

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

$$\widehat{Q}(s, \boldsymbol{a}) = \frac{\sum_{(s', \boldsymbol{a}')} K\left(d\left(\theta(s, \boldsymbol{a}), \theta(s', \boldsymbol{a}')\right)\right) \cdot Q(s', \boldsymbol{a}')}{\sum_{(s', \boldsymbol{a}')} K\left(d\left(\theta(s, \boldsymbol{a}), \theta(s', \boldsymbol{a}')\right)\right)} \ , \tag{9}$$

which can be evaluated for any continuous state/action pair $(s, \boldsymbol{a})$. Basically, this is a weighted sum of the action-values Q(s',$\boldsymbol{a}$') of all previously collected state/action pairs $(s', \boldsymbol{a}')$. The other components within (9) 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 (4) it can be seen as a density transformation

$$\begin{aligned} \theta(s_t, \boldsymbol{a}_t) &= \theta\left(p(\boldsymbol{q}_t | \boldsymbol{f}_t, \boldsymbol{a}_{t-1}, \boldsymbol{f}_{t-1}, \dots, \boldsymbol{a}_0, \boldsymbol{f}_0), \boldsymbol{a}_t\right) \\ &= \det\left(\boldsymbol{J}_{\boldsymbol{\zeta}_{\boldsymbol{a}_t}^{-1}}(\boldsymbol{q}_t)\right) p\left(\boldsymbol{\zeta}_{\boldsymbol{a}_t}^{-1}(\boldsymbol{q}_t) | \boldsymbol{f}_t, \boldsymbol{a}_{t-1}, \boldsymbol{f}_{t-1}, \dots, \boldsymbol{a}_0, \boldsymbol{f}_0\right) \end{aligned} \tag{10}$$

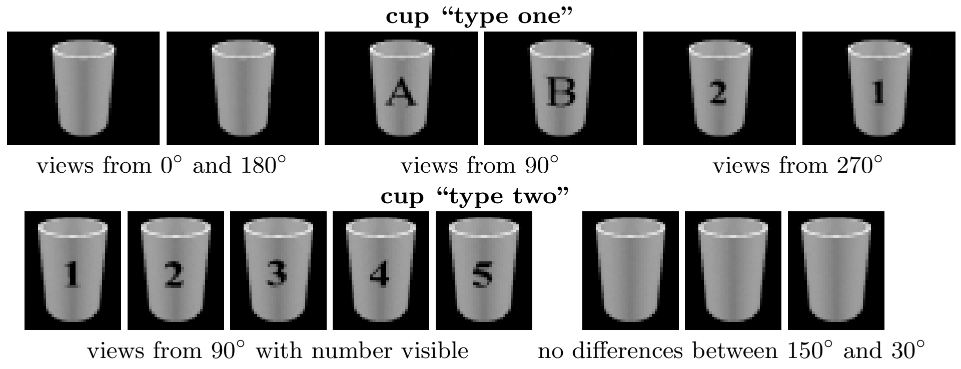

Fig. 2. Examples for objects that require viewpoint selection and fusion of images for proper recognition.

$$\boldsymbol{\zeta}_{\boldsymbol{a}}^{-1}(\boldsymbol{q}) = \begin{pmatrix} q_1 + a_1 \\ \vdots \\ q_m + a_m \end{pmatrix}, \quad \boldsymbol{J}_{\boldsymbol{\zeta}_{\boldsymbol{a}}^{-1}}(\boldsymbol{q}) = \begin{pmatrix} \frac{\partial(\zeta_{\boldsymbol{a}}^{-1})_1}{\partial q_1} & \cdots & \frac{\partial(\zeta_{\boldsymbol{a}}^{-1})_m}{\partial q_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial(\zeta_{\boldsymbol{a}}^{-1})_1}{\partial q_m} & \cdots & \frac{\partial(\zeta_{\boldsymbol{a}}^{-1})_m}{\partial q_m} \end{pmatrix} = \begin{pmatrix} 1 & & 0 \\ & \ddots & \\ 0 & & 1 \end{pmatrix}.$$

This density transformationwhich simply performs a shift of the density.

- A distance function $d(\cdot,\cdot)$ 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 (10) results in a density, the *Kullback-Leibler Distance*

$$\begin{aligned} d_{KL}(s_n, s_m) &= d_{KL}\left(p(\boldsymbol{q}|\boldsymbol{f}_n, \boldsymbol{a}_{n-1}, \boldsymbol{f}_{n-1}, \ldots), p(\boldsymbol{q}|\boldsymbol{f}_m, \boldsymbol{a}_{m-1}, \boldsymbol{f}_{m-1}, \ldots)\right) \\ &= \int p(\boldsymbol{q}|\boldsymbol{f}_n, \boldsymbol{a}_{n-1}, \boldsymbol{f}_{n-1}, \ldots) \log \frac{p(\boldsymbol{q}|\boldsymbol{f}_n, \boldsymbol{a}_{n-1}, \boldsymbol{f}_{n-1}, \ldots)}{p(\boldsymbol{q}|\boldsymbol{f}_m, \boldsymbol{a}_{m-1}, \boldsymbol{f}_{m-1}, \ldots)} d\boldsymbol{q}, \end{aligned}$$

  which can easily be extended to a symmetric distance measure, the so called *extended Kullback-Leibler Distance*

$$d_{EKL}(s_n, s'_m) = d_{KL}(s_n, s'_m) + d_{KL}(s'_m, s_n,), \tag{11}$$

  can be used. Please note that in general there is no analytic solution for (11) but as we represent our densities as particle sets anyway (see section 2.1) there are well-known ways to approximate (11) by Monte Carlo techniques.
- 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 π, can now be written, according to (8), as the optimization problem

$$\pi(s) = \underset{\boldsymbol{a}}{\operatorname{argmax}}\, \widehat{Q}(s, \boldsymbol{a}). \tag{12}$$

3 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

[3, 4] that the 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 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. In this setup the camera is restricted to move around the object on a circle, so that (3) reduces to $\boldsymbol{c}_i^t = (\Omega_{\kappa_i}\ {}^1\varphi_i^t)^T$ and $\boldsymbol{a}_t \in [0°, 360°]$. The classifier used to evaluate $p(\boldsymbol{f}_t|\boldsymbol{c}_i^t)$ for the fusion of images (see section 2.1) is based on the continuous statistical eigenspace approach presented in [6].

The four cups of "type one" in 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. In a training step a total of about 600 action-values $Q(\cdot,\cdot)$ were collected during 200 episodes; each for different settings of the return parameter γ (6). The evaluation was performed with 500 sequences with randomly chosen classes and starting views. There exists a theoretical minimum for the necessary sequence length of ≈ 2.3 views: Number and letter are visible within about 120° each, requiring 2 views for that case and 3 views for the remaining viewing area. The recognition rates at the end of the sequences were as expected 100% for these rather simple objects. But it is very interesting, and this is the main point, that the average length of the planned sequences shown in Table 1 is very close to the calculated minimum of necessary views. This indicates very strongly that the learned strategy for recognition is optimal. Due to the nature of the objects, the learned strategy is the same for all values of γ.

The cups of "type two" in 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. The cups can be classified correctly and stable within an area of nearly 120°. Localization of the cups is possible within an area of about 144°. In the training a total of about 430 action-values Q were collected during the 200 performed episodes that generated about 600 different views. The optimal strategy must bring up the number with a minimum number of views. We first expected our viewpoint selection to learn a strategy that moves the camera by 120° if the cup can not be classified, as this would result in an average minimum sequence length of 2.0. But the learned strategy — which is the same for each trained value of γ — moves the camera by 180° if no classification or localization is possible. The reason for this strategy is that it allows for a better fusion and thus for an unambiguous recognition. Surprisingly, even this learned strategy has a theoretical minimum for the necessary sequence length of only ≈ 2.03 steps. As the results in Table 1 show, the average sequence length required by our viewpoint selection is just about the minimal number of required views.

4 Conclusion and Future Work

In this paper we have presented a general framework for viewpoint selection and the fusion of the generated sequence of views. The approach works in continuous state and action spaces and is independent of the chosen statistical classifier. Furthermore the system can be be trained automatically without user interaction. We claim that these properties have not yet been provided by any other approach. The experimental results on two objects that require different strategies for recognition have shown that an optimal planning strategy was learned.

Table 1. Results of viewpoint selection. Calculation time given is for planning one step, computed on a Pentium IV 2.4GHz.

	cup "type one"			**"cup type two"**		
	$\gamma = 0$	$\gamma = 0.5$	$\gamma = 1.0$	$\gamma = 0$	$\gamma = 0.5$	$\gamma = 1.0$
classification rate	100% (as expected)					
average sequence length	2.36	2.28	2.31	2.13	1.96	2.13
calculation time	$\approx$ 13.1s			$\approx$ 12.3s		

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 [3, 4]. Another important point for real world applications are *costs* of movement that could not be discussed in this paper. Finally, for higher dimensional state spaces, other reinforcement learning methods might be necessary to reduce training complexity.

References

1. D.P. Bertsekas and J.N. Tstsiklis. *Neuro–Dynamic Programming.* Athena Scientific, 1996.
2. H. Borotschnig, L. Paletta, M. Prantl, and A. Pinz. A Comparison of Probabilistic, Possibilistic and Evidence Theoretic Fusion Schemes for Active Object Recognition. *Computing*, 62:293–319, 1999.
3. F. Deinzer, J. Denzler, and H. Niemann. On Fusion of Multiple Views for Active Object Recognition. In *DAGM 2001*, pages 239–245, Berlin, 2001. Springer.
4. F. Deinzer, J. Denzler, and H. Niemann. Improving Object Recognition By Fusion Of Multiple Views. In *3rd Indian Conference on Computer Vision Graphics and Image Processing*, pages 161–166, Ahmedabad, Indien, 2002. Allied Publishers Pvt. Ltd.
5. J. Denzler and C.M. Brown. Information theoretic sensor data selection for active object recognition and state estimation. *PAMI*, 24(2), 2002.
6. Ch. Grässl, F. Deinzer, and H. Niemann. Continuous Parametrization of Normal Distributions for Improving the Discrete Statistical Eigenspace Approach for Object Recognition. In *PRIP 2003*, May 2003. submitted.
7. M. Isard and B. Andrew. CONDENSATION — Conditional Density Propagation for Visual Tracking. *IJCV 98*, 29(1):5–28, 1998.
8. R.E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, pages 35–44, 1960.
9. B. Krebs, M. Burkhardt, and B. Korn. Handling Uncertainty in 3D Object Recognition using Bayesian Networks. In *ECCV 98*, pages 782–795, Berlin, 1998.
10. C.B. Madsen and H.I. Christensen. A Viewpoint Planning Strategy for Determining True Angles on Polyhedral Objects by Camera Alignment. *PAMI*, 19(2), 1997.
11. P. Lehel and E.E. Hemayed and A.A. Farag. Sensor Planning for a Trinocular Active Vision System. In *CVPR*, pages II:306–312, 1999.
12. S. D. Roy, S. Chaudhury, and S. Banerjee. Recognizing Large 3-D Objects through Next View Planning using an Uncalibrated Camera. In *ICCV 2001*, pages II: 276 – 281, Vancouver, Canada, 2001. IEEE Computer Press.
13. B. Schiele and J.L. Crowley. Transinformation for Active Object Recognition. In *ICCV 98*, pages 249–254, Bombay, India, 1998.
14. R.S. Sutton and A.G. Barto. *Reinforcement Learning.* A Bradford Book, Cambridge, London, 1998.
15. P. Viola and W.M. Wells III. Alignment by Maximization of Mutual Information. *International Journal of Computer Vision*, 24(2):137–154, 1997.