Frank Deinzer, Joachim Denzler, Heinrich Niemann

Viewpoint Selection - A Classifier Independent Learning Approach

appeared in:
IEEE Southwest Symposium on Image Analysis and Interpretation (SSIAI2000)
Austin, Texas, USA
p. 209–213
2000

Viewpoint Selection - A Classifier Independent Learning Approach

F. Deinzer*, J. Denzler, H. Niemann
Chair for Pattern Recognition
Friedrich-Alexander University Erlangen-Nürnberg
Martensstr. 3, D–91058 Erlangen, Germany
{deinzer,denzler,niemann}@informatik.uni-erlangen.de

Abstract

This paper deals with an aspect of active object recognition for improving the classification and localization results by choosing optimal next views at an object. The knowledge of "good" next views at an object is learned automatically and unsupervised from the results of the used classifier. For that purpose methods of reinforcement learning are used in combination with numerical optimization. The major advantages of the presented approach are its classifier independence and that the approach does not require a priori assumptions about the objects. The presented results for synthetically generated images show that our approach is well suited for choosing optimal views at objects.

1. Motivation

One important task in image analysis is the classification and localization of objects. The classification rate and the localization accuracy mainly depends — due to ambiguities between objects — on the chosen viewpoint of the camera. In the past 15–20 years, the realization of classification systems, for example, statistical classifier, neural networks etc., were based on a *passive approach* using a single image. This image was used for feature extraction followed by a classification. If the information in the image was not suited to decide for a certain class and pose, usually an error in recognition occurred, or the object was rejected.

Think, for example, of several cups which differ only in some kind of symbol on one side. They can only be distinguished by that symbol. In a passive approach the only source of information is a single image of a cup, perhaps with the symbol not visible. Thus, it is quite natural in such a situation to collect additional information by choosing new views, which might help in identifying the class of the object more reliable. The problem of course is where to move the camera to and take an image of the next view.

As one wants to maximize the number of correct classifications without looking at the object from every possible position an *active approach* is necessary which chooses a minimum number of dedicated views at an object. The advantage of such an active approach — in the following called *viewpoint selection* — for object classification and localization is obvious and has been discussed recently in the literature. For example, [7] has presented an active object recognition system based on a statistical measure, the transinformation. Performing a statistical classification the most promising viewpoint can be calculated for the complete set of objects in a supervised training step in advance. In [3] different frameworks for handling uncertainty and decision making (probabilistic, possibilistic and Dempster–Shafer theory) have been compared with respect to viewpoint selection. But this approach can only handle discrete positions and viewpoints. In [2] a knowledge based approach using semantic networks is used for active scene exploration.

In this contribution we will tackle the problem of optimal viewpoint selection for object recognition without making a priori assumptions about the objects and the classifier. The problem is to determine the next view at an object given a certain decision concerning the class and the estimated pose of the object. Therefore we need to construct a function, which maps the class and pose decision of a classifier to an action, i.e. a movement of the camera to a new position, from which it is expected to improve the reliability of classification and localization. The function, of course, should be learned automatically by defining a quality measure for each class/pose and action pair, i.e. how useful it is to choose a certain new view given a classification and localization result. Further on, the method should be classifier independent and should handle continuous viewpoint and pose spaces.

The description and characterization of the problem and the demands lead directly to the so called *reinforcement learning* approach, which is well known in the artificial in-

*This work was supported by the DFG under grant SFB603

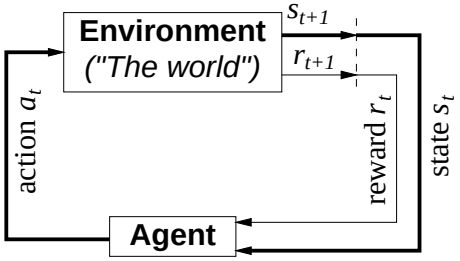

Figure 1. Principles of reinforcement learning. A system overview [8].

telligence community but which has not been used widely in the area of computer vision. A short summary of the basics of reinforcement learning is presented in the next section together with a detailed description, how we map this framework to the problem of viewpoint selection for object recognition. In section 3 we shortly describe the classifier we used for our experiments. The results of our experiments are presented in section 4.

2. Reinforcement Learning for Viewpoint Selection

2.1 Principles of Reinforcement Learning

First we summarize briefly the general principle of reinforcement learning (RL), which can be seen in Figure 1 (following [8]). The agent observes at time step t the *state* s_t of the world and interacts with it by performing a so called *action* a_t. Each action changes the state of the environment according to a certain probability. The usefulness of an action in a certain state is judged by a so called *reward* r_t, which is returned by the environment: high rewards for "good" actions and low rewards for "bad" actions. A more detailed description and introduction into RL can be found in [8].

The goal is now, to perform a sequence of actions to reach a final state and maximize the accumulated and weighted rewards — the *return* R_t — during a sequence of state–action pairs:

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \quad \text{with weights } \gamma \in [0;1]. \qquad (1)$$

One of the central terms of reinforcement learning is the *action–value function* $Q(s,a)$ which describes the *expected return*, starting from state s and taking the action a:

$$Q(s,a) = E\{R_t \,|\, s_t{=}s, a_t{=}a\}. \qquad (2)$$

How can the action–value function for each state–action pair be computed? One simple method (known as Monte Carlo learning [8]) is to define the action–value function as the average over all returns which were observed for a state–action pair. These action–values are used to compute the agent's *decision policy* $\pi(\cdot)$. As the agent wants to choose actions that maximize the return, an optimal policy can be written as

$$\pi(s) = \operatorname*{argmax}_a Q(s,a). \qquad (3)$$

In practice, one is choosing the optimal policy only with a certain probability and a random policy otherwise. This allows the agent to choose random actions and enables him to evaluate state–action pairs that have not been visited before so that it can be assured that the agent is improving and adapting continuously.

We now transfer this general principle to our viewpoint selection problem. In the approach presented in this paper we allow camera movements on a circle around the object, i.e. we have one degree of freedom for the viewing position. The states $s \in \mathbb{N} \times [0;360)$ correspond to the estimated class-number of the object and its estimated pose — the viewing position on the circle. The actions $a \in [0;360)$ correspond to the relative movement of the camera on the circle around the object.

The reward is given by the uniqueness of the classification result, i.e. the difference between the first and the second best hypothesis for the object's class (c.f. section 3). A large distance means, that the object's class has been estimated reliably. Of course, other definitions of the reward are possible, for example, adding costs for moving the camera. The reward is the interface to the classifier's results and its definition is the only classifier dependent portion which has to be adapted upon the replacement of the classifier.

2.2 Function Approximation for Continuous Reinforcement Learning

If the environment is restricted to discrete and finite sets of states and actions, learning a "perfect" behavior is simply a task of collecting enough returns for each possible state–action pair and searching tables of action–values. But dealing with real–world viewpoint selection, one has to take into account, that neither the states nor the actions are discrete. The continuous state and action space makes it impossible to collect all possible state–action combinations. For that reason, an estimation function $\hat{Q}(s,a)$ is introduced which approximates the expected reward $Q(s,a)$ for arbitrary, continuous state–action pairs. This estimation is based on the collected returns $Q(s,a)$ of the previously visited state–action pairs. It has to provide some properties:

- the function represented by $\hat{Q}(s,a)$ has to be smooth without discontinuities
- only very few collected returns $Q(s,a)$ should be necessary for calculating $\hat{Q}(s,a)$

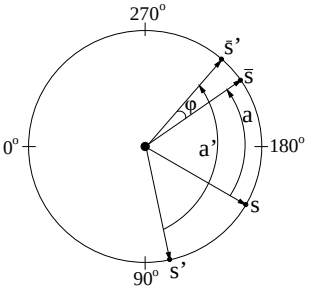

Figure 2. Source states (s, s'), executed actions (a, a') and resulting expected destination states $(\bar{s}, \bar{s}')$. The distance $d(s, a, s', a')$ between the expected destination states is denoted as $\sphericalangle(\bar{s}, \bar{s}') = \varphi$.

- a priori assumptions about the objects must not be necessary

From a general point of view an approximation of an action–value can be written as a weighted average over all previously collected returns [6]:

$$\hat{Q}(s,a) = \frac{\sum\limits_{(s',a')\in\mathcal{Q}} K(d(s,a,s',a')) \cdot Q(s',a')}{\sum\limits_{(s',a')\in\mathcal{Q}} K(d(s,a,s',a'))}, \quad (4)$$

where $\mathcal{Q}$ denotes the set of all visited state–action pairs for the current class, $K(d_i)$ a suitable kernel function (e.g. Gauss) for weighting the distance $d(s, a, s', a')$ of the state–action pairs (s, a) and (s', a'). State–action pairs that are "close" to each other shall be rated high in the weighted average because these are the valuable pairs. Their information can contribute a lot to the approximation. The modeling of the distance function was influenced by the following idea: The distance $d(s, a, s', a')$ between the two *expected destination states* of the state–action pairs (s, a) and (s', a').

Assuming that the actions a and a' lead to two new destination states $\bar{s}$ and $\bar{s}'$:

$$s \xrightarrow{a} \bar{s}, \quad s' \xrightarrow{a'} \bar{s}' \quad (5)$$

The closer the two destination states are to each other, the more adaptable $Q(s', a')$ for the estimation of $\hat{Q}(s, a)$. For calculating the expected destination states we are currently assuming that the pose estimation of the state s is correct and that the action a is affecting the environment in an "optimal" way. For example, if the estimated position of s is $200°$ and an action a moves the camera $100°$ the pose of the expected destination state $\bar{s}$ will be $300°$.

As pose estimation is not done with absolute coordinates in space, but with angles on the circle, calculating distances between states can be done by measuring the angle between the vector to the points of the circle given by the states (see Figure 2):

$$d(s,a,s',a') = \sphericalangle(\bar{s},\bar{s}') \text{ with } d(s,a,s',a') \leq 180° \quad (6)$$

This distance leads to our approximation $\hat{Q}(s, a)$ of any action–value (eq. (4)) with the kernel function $K(\cdot)$ defined as

$$K(d) = exp\left(-\frac{d^2}{D^2}\right) \quad (7)$$

The parameter D describes how local (for a small D) or global (for a large D) the approximation is working. "Local" means, that faraway states have only a slight influence on the approximated action–value resulting in a very detailed approximation. This is very useful if you have a lot of state–action pairs. On the contrary, a "global" approximation includes data over a wide area of distances and is suitable, if only a very limited set of collected action–values is available.

2.3 Calculating the Optimal Action

Returning to our reinforcement learning problem and the calculation of an optimal decision policy, $\pi(s)$ can now be formulated in a very similar way to the original, discrete reinforcement learning approach (see eq. (3)), by solving

$$\pi(s) = \underset{a}{\operatorname{argmax}}\, \hat{Q}(s,a) \quad (8)$$

with one of the many well-known numerical techniques. Given a state s (the current class and pose estimation) searching for the action that maximizes $\hat{Q}(s, a)$ will lead to the optimal camera movement.

In our approach we are currently using Adaptive Random Search ARS [9] combined with a local simplex for solving eq. (8).

3 Classifier used for Viewpoint Selection

Currently we are evaluating our approach for viewpoint selection with an implementation [2] of a classifier based on the eigenspace approach introduced by [4].

Object recognition is dealing with assigning a class number κ to an object found in an image of size $N \times M$ which is represented by the column vector $\boldsymbol{f} \in \mathbb{R}^{(N \cdot M)}$. This image vector $\boldsymbol{f}$ is transformed into a feature vector $\boldsymbol{c} = (c_1, c_2, \ldots, c_K)^T \in \mathbb{R}^K$ by a linear transformation

$$\boldsymbol{c} = \boldsymbol{\Phi}^\kappa \boldsymbol{f} \in \mathbb{R}^{K \times (N \cdot M)}, \quad \boldsymbol{\Phi}^\kappa = (\boldsymbol{\phi}_1^\kappa, \boldsymbol{\phi}_2^\kappa, \ldots, \boldsymbol{\phi}_K^\kappa)^T.$$

The $\boldsymbol{\phi}_i^\kappa$ correspond to the K largest eigenvectors of the covariance matrix of the n training images $\boldsymbol{f}_1^\kappa, \boldsymbol{f}_2^\kappa, \ldots, \boldsymbol{f}_n^\kappa$

Figure 3. Examples of synthetical images of five cups showing the viewing angles $0°$, $50°$, $270°$, $320°$, $130°$, $270°$ and the cups' coordinate system with the handle at $90°$ and the digit at $270°$.

of class κ. During the training each training image $\boldsymbol{f}_i^\kappa$ is projected into the object's eigenspace building the model database $\boldsymbol{c}_i^\kappa$. For pose estimation the known pose parameters of each image $\boldsymbol{f}_i^\kappa$ are stored together with the feature vector $\boldsymbol{c}_i^\kappa$.

Classification of an image $\boldsymbol{f}$ in eigenspace is done by searching for the feature vector with minimum distance d to the image:

$$\kappa = \underset{\kappa}{\operatorname{argmin}}\, d(\boldsymbol{f}|\kappa) = \underset{\kappa}{\operatorname{argmin}}\, \underbrace{\min_i ||\boldsymbol{\Phi}^\kappa \boldsymbol{f} - \boldsymbol{c}_i^\kappa||^2}_{d(\boldsymbol{f}|\kappa)} \quad (9)$$

In our reinforcement learning problem we are using this distance measure for calculating the reward r_t (see eq. (1)):

$$r = \underbrace{\min_{\lambda,\lambda\neq\kappa} \left(\min_\kappa d(\boldsymbol{f}|\kappa) - d(\boldsymbol{f}|\lambda) \right)}_{\substack{\text{difference between the first and} \\ \text{the second best hypothesis}}} \quad (10)$$

Calculating the reward does not consider whether or not the classification is correct. This is not relevant for our experiments as we are working with synthetically generated images (see the following section for details). Reconsidering the definition of the reward may be necessary when using real images with a weak classifier.

4 Experiments and Results

The experiments we performed to evaluate the capacity of our approach are based on synthetically generated ("raytraced") images of the cups shown in Figure 3. These cups are completely the same except a unique digit from 1 to 5 on the front side of the cup. This digit can be seen — i.e. the cups can be distinguished — if the cups are viewed from a position from $215°$–$330°$. Since we concentrate on the classifier independent viewpoint selection aspect using RL, we are currently interested in a fast image acquisition which is provided by the synthetical images generated by the raytracer POV-Ray [1]. This allowed us to change the RL method quickly for comparison without the need to spend a lot of time for taking new views at the object.

Our RL system is modelled in the way described in section 2.1. The states represent the classifier's estimated class

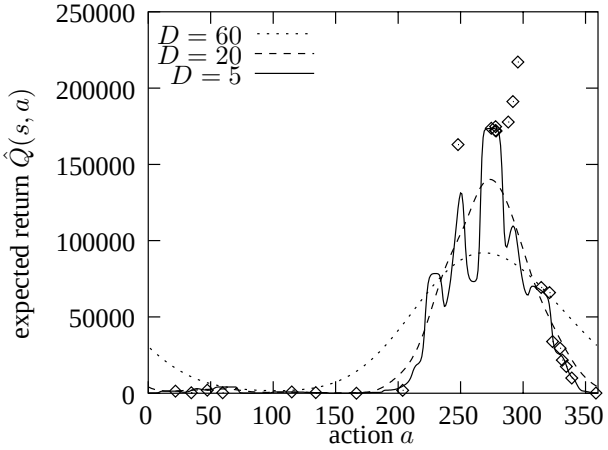

Figure 4. The approximation $\hat{Q}(s,a)$ of the action–value function for state $s = (1, 0°)^T$. The rhombs are showing the collected 20 action–values $Q(s',a')$ for cup 1.

and localization parameters ($\{1,2,3,4,5\} \times [0;360)$). The actions are the intended relative changes of the viewing angle ($[0;360)$). This means. e.g., taking action 180 is moving the camera to the current contrary position. The reward expresses the significance of the current view and is defined as described in eq. (10).

Training was done by collecting the rewards of 100 and 1000 (20 and 200 for each cup) random state–action pairs. This means, starting at a randomly chosen viewing angle, and taking a random action, the resulting reward was stored as $Q(s',a')$ for this state–action pair. As we are working with synthetical images, we added uniform noise to both the reward ($\pm$ 10%) and the pose estimation ($\pm\, 5°$). As training is simply a task of collecting rewards for state–action pairs, the CPU time needed depends mainly on the speed of the raytracer and the classifier. In our case, executing one action and storing the resulting reward takes approx. 10 seconds on a SGI O^2 (R10000, 195MHz). This results in a total time needed for training of 17 resp. 170 minutes.

After collecting the training action–values $Q(s',a')$, we are able to approximate $\hat{Q}(s,a)$ for arbitrary, continuous state–action pairs (s,a). In Figure 4 and 5 approximations of $\hat{Q}(s,a)$ for the state $s = (1, 0°)^T$ with different settings

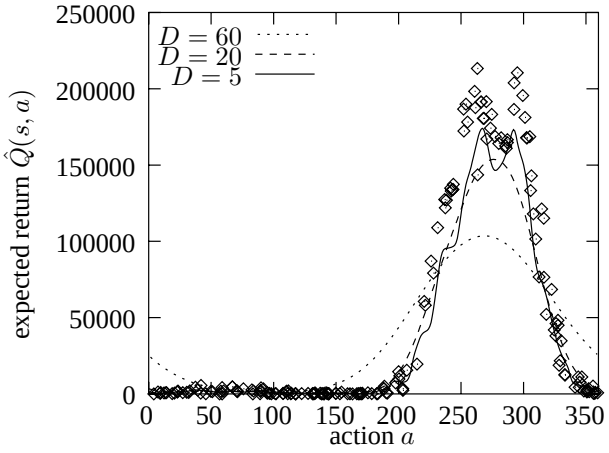

Figure 5. The approximation $\hat{Q}(s,a)$ of the action–value function for state $s = (1, 0°)^T$. The rhombs are showing the collected 200 action–values $Q(s', a')$ for cup 1.

of the parameter D (see eq. (7)) are shown. This means, the plots show the usefulness of a specific action — the relative movement of the camera — if the camera is having the view at the leftmost picture in Figure 3 to cup 1. The high action values in the plots between $215°$ and $330°$ are indicating significant viewpoint positions. This is the area of the cups from where the digit is visible. The smaller the parameter D the more detailed the approximation. But if the number of collected state–action pairs is too small, an approximation with many local maxima may occur as it is shown in Figure 4 for $D = 5$. An important point in our approach is, that we are only interested in a correct position of the maximum of $\hat{Q}(s,a)$ and not a minimal approximation error.

The evaluation of our system was done by performing approx. 400 viewpoint selections for randomly chosen starting classes and viewing angles. The system's task was to find an action that changes the viewing angle to a position where the digit is visible. A viewpoint selection was classified as correct, if the chosen action lead to a viewing angle between $215°$ and $330°$. We performed our viewpoint selections on the two training sets described above:

- **100 action–values:** all chosen actions resulted in viewing angles that made an unique identification of the cups possible. This means that 100% of the viewpoint selections are correct.

- **1000 action–values:** as expected after the result for 100 action–values, correct viewing angles were found always.

We want to emphasize that a correct viewpoint does not necessarily imply a correct classification.

The CPU time required for one viewpoint selection (i.e. finding the estimation function's maximum) is 0.14 resp. 1.15 seconds for the training set with 100 resp. 1000 state–action pairs. The approximation of one $\hat{Q}(s,a)$ takes approximately $8 \cdot 10^{-4}$ resp. $6 \cdot 10^{-3}$ seconds.

5 Conclusion

We have presented an approach for classifier independent optimal viewpoint selection based on the ideas of reinforcement learning. Our method extends the primarily discrete techniques of reinforcement learning to a continuous optimization problem for finding the optimal next viewpoint. Therefore the expected significance of the images taken from an arbitrary viewpoint is approximated. The experiments have proven the suitability and advantages of the proposed method. The system was always able to chose a view at the object that enables the classifier to make an unique classification.

Our future work will concentrate on the use of other statistical classifiers, e.g. [5], the extension of our approach to 2-D viewpoint selection and the handling of multiple ambiguities within single objects.

References

[1] The Persistence of Vision Raytracer (POV-Ray). http://www.povray.org, checked on Sep. 22, 1999.

[2] U. Ahlrichs, J. Fischer, J. Denzler, C. Drexler, H. Niemann, E. Nöth, and D. Paulus. Knowledge based image and speech analysis for service robots. In *Proceedings Integration of Speech and Image Understanding*, pages 21–47. IEEE Computer Scociety, 1999.

[3] 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.

[4] H. Murase and S. Nayar. Visual Learning and Recognition of 3–D Objects from Appearance. *International Journal of Computer Vision*, 14:5–24, 1995.

[5] J. Pösl and H. Niemann. Wavelet features for statistical object localization without segmentation. In *Proceedings of the International Conference on Image Processing (ICIP)*, volume 3, pages 170–173, 1997.

[6] J. C. Santamaria, R. S. Sutton, and A. Ram. Experiments with Reinforcement Learning in Problems with Continuous State and Action Spaces. Technical report, University of Massachusetts, Amherst, Computer Science, 1996.

[7] B. Schiele and J. Crowley. Transinformation for active object recognition. In *Proceedings of the Sixth International Conference on Computer Vision*, Bombay, India, 1998.

[8] R. Sutton and A. Barto. *Reinforcement Learning*. A Bradford Book, Cambridge, London, 1998.

[9] A. Törn and A. Žilinskas. *Global Optimization*, volume 350 of *Lecture Notes in Computer Science*. Springer, Heidelberg, 1987.