Published in :
Pattern Recognition - 28th DAGM Symposium 2006
Editors: K. Franke, K.-R. Müller, B. Nickolay, R. Schäfer
Springer
ISBN-10 : 3-540-44412-2
Pages : 637-646

Handling Camera Movement Constraints in Reinforcement Learning Based Active Object Recognition

Christian Derichs[*] and Heinrich Niemann

Chair for Pattern Recognition, Department of Computer Science, University Erlangen-Nürnberg, Martensstr. 3, 91058 Erlangen
{derichs,niemann}@informatik.uni-erlangen.de

Abstract. In real world scenes, objects to be classified are usually not visible from every direction, since they are almost always positioned on some kind of opaque plane. When moving a camera selectively around those objects for classifying them in an active manner, a hemisphere is fully sufficient for positioning meaningful camera viewpoints. Based on this constraint, this paper addresses the problem of handling planned camera actions which nevertheless lead to viewpoints beyond the plane of that hemisphere. Those actions arise from the uncertainty in the current vertical camera position combined with the view planning method's request of a relative action. The latter is based on an optimized and interpolating query of a knowledge base which is built up in a Reinforcement Learning training phase beforehand.
This work discusses the influence of three different, intuitive and optimized, methods for handling invalid action suggestions generated by Reinforcement Learning. Influence is measured by the difference in classification results after each step of merging the image data information with active view planning.
Keywords Active Vision, Viewpoint Selection, Reinforcement Learning

1 Introduction

The basic idea of active object recognition is the optimized selection of the viewpoints relative to an item in order to classify it reliably with a minimal amount of recorded sensor data, such as camera images. Naturally, this comes along with the necessity of moving a camera around the considered object and fusing the gathered information. Aspiring to optimality, some kind of camera movement planning is required, which in our approach is based on Reinforcement Learning [11]. We have already shown in various publications [4] [5] that this approach outperforms a random viewpoint selection when the camera movement is restricted to a circular path around the object.

The enhancement from the 1-dimensional path to the complete, 2-dimensional sphere around the item is quite simple at a first glance. But the restriction appearing in practice is that the majority of objects is positioned on an opaque plane, like a table or the floor. Consequently, cameras cannot be positioned in any way to take a meaningful image from below those objects. Keeping this in mind, it is a reasonable approach to initially

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

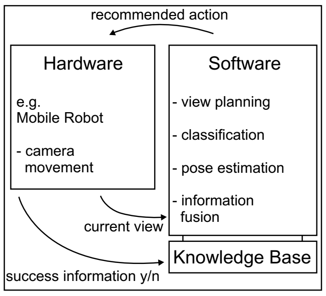

Fig. 1. The elementary structure of the active view planning system

model the object only with image information gained from camera viewpoints that are arranged on a hemisphere around this object. In contrast to this modeling, later real world camera movements, as well as their horizontal and vertical positions relative to the considered object, are almost always afflicted with uncertainty. In particular, a camera controlling mobile robot may encounter an object worth classifying at some time during its so far undirected movement in an arbitrary environment. So regarding the relation between camera and object position, no initial alignment of coordinate systems can be done. Consequently, incoming information is limited to the recorded image of the object. Based on this information, a probabilistic suggestion about the relative camera position and the object class can then be established and an optimal next viewpoint for recognition can be calculated. Approaching the latter mostly means moving the whole robot as well as changing the camera angle, both being afflicted with inaccuracy.

Now the problem regarding the hemisphere constraint results from the fact that the software part (see figure 1) might randomly choose or even plan a relative action which actually would not result in a position on the mentioned hemisphere. Unfortunately, neither the hardware front end is not able to detect such an invalid action without actually performing it. Since the movement is at least partially performed when eventually recognizing the impracticability of the proposed action, a replanning would waste one step in the recognition process, which is considered worst-case in optimal, active view planning. Therefore, the hardware in particular must not reject or request an action from the software part, but has to promptly handle the given action instruction somehow. Considering the other direction, in order to keep the approach universal the software is assumed to not explicitely get to know anything about the success of the real action, it merely obtains the next image. In our framework, success information is just used for rating executed actions (see chapter 2).

Thus, the problem under consideration is how to immediately deal with requested actions that are invalid in a sense of camera movement constraints and how to adapt this to the knowledge representation, in fact without raising the planning effort disproportionately. The latter is exactly the point most of the related work is missing. For example, [9] explicitely puts a lot of effort into the exact recovery of conditions obtained before performing a failed action in order to try another action then. [10] proposes an approach which emerges from robot navigation and modifies a planned, demanded action or action sequence if this did not lead to a recognizable state enhancement in the preceding time steps. Of course, in our active view planning system we cannot wait for the system to recognize such a dead end. Other work from the area of *Replanning* displaces the error handling work towards the training phase, like [2], which alternately inhibits action components during training in order to directly learn backup plans. Similarly, [13] introduces a *plan transformation* that varies a proposed action to similar one which is most probably valid. But this transformation advice is again based on an expanded training phase including the buildup of a *plan library*. Thus, those approaches are worth considering if training complexity is not a matter. A completely different, Reinforcement Learning adjusted idea is suggested by [6], which incorporates the risk of a valid action to become invalid due to some inaccuracy into an action's rating. A similar approach and it's common disadvantages are discussed in chapter 3.

Chapter 2 gives an outline of the basic methods of Reinforcement Learning and explains the role of the problem specific variables within this framework. Afterwards, chapter 3 first introduces the probabilistic description of the class and pose assumptions of an object via a set of particles and then shows three methods for propagating these particles according to the action handling direction under consideration. Chapter 4 compares the proposed methods regarding their impact on experimental classification results.

2 Reinforcement Learning

2.1 Basic Principles

In a basic Reinforcement Learning approach, the three decisive items are states s_t, actions $\boldsymbol{a}_t$ that lead to new positions and rewards r_t that rate the performed action $\boldsymbol{a}_t$ given state s_t as its starting point. A timestamp t clarifies that in the assumed environment we have multiple episodes of T temporally successive actions $\langle \boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_T \rangle$ and resulting states $\langle s_1, s_2, \ldots, s_{T+1} \rangle$ each.

For our purpose:

• **States** within the development environment are multi-modal probability distributions (see [4] [5]). They are discrete concerning the number of object classes Ω_κ and continuous within the values for horizontal (Φ_h) and vertical (Φ_v) camera positions relative to the object. Consequently, they contain probabilistic assumptions about the class and pose of an object under consideration. Those assumptions arise from the information extracted from the image data $\boldsymbol{f}_t$ the camera acquires in each time step. Thus, the state densities can be presented by

$$s_t = p(\boldsymbol{q}(t)| \langle f \rangle_t , \langle \boldsymbol{a} \rangle_{t-1}) \quad \text{with} \quad \boldsymbol{q} = (\Omega_\kappa, \Phi_h, \Phi_v)^T . \tag{1}$$

Here, $\langle \boldsymbol{f} \rangle_t = \boldsymbol{f}_t, \boldsymbol{f}_{t-1}, \ldots, \boldsymbol{f}_1$ and $\langle \boldsymbol{a} \rangle_{t-1} = \boldsymbol{a}_{t-1}, \ldots, \boldsymbol{a}_1$ indicate the fusion of all image information and camera movement knowledge gathered during a whole episode. Therefore, the probability distribution s_t itself is a fused product of information gathered in multiple time steps. For a detailed explanation of the fusion procedure refer to [3].

• **Actions** are the movements of the camera in-between the taking of two consecutive images. For the purpose of this work, actions $\boldsymbol{a}_t = (a_{h,t}, a_{v,t})^T$ are limited to two components, the horizontal (a_h) and the vertical (a_v) movement of the camera fixed on a predefined hemisphere around the object.

• **Rewards** follow the usual definition in Reinforcement Learning

$$R_t = \sum_{n=0}^{\infty} \gamma^n r_{t+n} \quad \text{with } \gamma \in [0;1] \,, \tag{2}$$

where r_t is the immediate reward when analyzing a next state s_t and γ is a weighting factor whose influence is increasingly reduced with a growing step indicator n. Summing up the sequentially appearing rewards r_t results in a forward-looking reward, called return R, when at time step t within an episode. In practice, episode lengths are finite and summation in (2) is aborted accordingly. Since rewards are calculated from the resulting state representation s_t, the choice of a significant property of those densities is the crucial task in Reinforcement Learning. So we decided for high rewards when the most probable class Ω_κ out of k classes has high confidence, according to

$$r_t = \max_i \int_{\Phi_h} \int_{\Phi_v} p(\boldsymbol{q}^i(t+1)|\, \langle f \rangle_{t+1} \,, \langle \boldsymbol{a} \rangle_t)\, d\boldsymbol{q}^i \quad \text{with} \quad \boldsymbol{q}^i = (\Omega_{\kappa=i}, \Phi_h, \Phi_v)^T \,. \tag{3}$$

Consequently, rewards in this approach obey the relation $k^{-1} \le r_t \le 1$.

2.2 Calculation of the Behavior Policy

Now given a rewarding rule, we can build up a knowledge base during training containing action-value functions Q which represent the quality of an action $\boldsymbol{a}$ in state s depending on the return's expectation value:

$$Q(s, \boldsymbol{a}) = E\left\{R_t | s_t = s,\ \boldsymbol{a}_t = \boldsymbol{a}\right\} \tag{4}$$

Of course, during training we acquire a fully calculated return R_t since we perform the whole episode before expanding the knowledge base. But at runtime in every time step t we can only make assumptions about the future behavior, so an expectation value is the appropriate formulation. In general, we cannot expect to get only such states s during evaluation that we have already seen in the training phase, since we work in a continuous environment. To nevertheless be able to declare a best action in every situation we acquire, an approximation term $\widehat{Q}(s, \boldsymbol{a})$ is necessary:

$$\widehat{Q}(s, \boldsymbol{a}) = \frac{\sum\limits_{(s', \boldsymbol{a}')} K\left(d\left(\theta(s, \boldsymbol{a}), \theta(s', \boldsymbol{a}')\right)\right) Q(s', \boldsymbol{a}')}{\sum\limits_{(s', \boldsymbol{a}')} K\left(d\left(\theta(s, \boldsymbol{a}), \theta(s', \boldsymbol{a}')\right)\right)} \,. \tag{5}$$

- $(s', \boldsymbol{a}')$ are the state-action pairs already stored in the knowledge base.
- $\theta(s, \boldsymbol{a})$ is the resulting multi-modal density function when transforming s according to an action $\boldsymbol{a}$. The various rules for this transformation are the gist of this paper and will be extensively discussed in chapter 3.
- d calculates the distance between two density functions using the extended Kullback-Leibler distance
- $K(x) = exp(-x^2/D^2)$ is a Gaussian kernel for weighting those distances d. The free kernel parameter D determines the smoothness or rather the local fineness of the approximation in (5).

Readdressing the topic of this work, the approximative character of $\widehat{Q}(s, \boldsymbol{a})$ is—next to the movement uncertainty—the main reason for obtaining invalid actions during runtime at all. The final step for finding the best action in the current state is an optimized global Adaptive Random Search [12] over all actions in question, followed by a local Simplex. Consequently, we obtain the optimal considered action $\breve{a}$:

$$\breve{a} = \underset{\boldsymbol{a}}{\operatorname{argmax}} \widehat{Q}(s, \boldsymbol{a}). \tag{6}$$

3 Handling Critical Actions

To understand the complete process when invalid actions occur, we first give a more precise insight into the structure of the multi-modal densities representing the states s_t. Note that in object recognition tasks, pose and class probability functions are generally not normally distributed. So unfortunately we cannot make use of the well known Kalman Filter [8] for the necessary density propagation. Instead, the proposed method applies a Particle Filter [1] to that problem, resulting in a probability density that is represented by a set $\varGamma_t = \left\{\rho_t^1, \ldots, \rho_t^N\right\}$ of N single particles ρ_t. Each of those particles $\rho = \left\{\tilde{\varOmega}_\kappa, \tilde{\varPhi}_h, \tilde{\varPhi}_v, \omega\right\}$ contains information about the class $\tilde{\varOmega}_\kappa$, the horizontal pose ($\tilde{\varPhi}_h$) and the vertical pose ($\tilde{\varPhi}_v$) it is representing. Additionally it holds an entry for its own weighting ω. Density propagation is then easily accomplished via the Condensation algorithm which is directly aligned to those particle representations. For details about this method refer to [7].

The purpose of this work is to provide and compare meaningful instructions for camera movements based on invalid movement demands, i.e. when they would exceed either the north pole or the hemisphere's plane in the vertical direction. Note that the north pole was additionally addressed as a critical edge in order to provide some symmetry to the planning task. Besides, this assumption should prevent problems when integrating actions' costs into the learning process, which is outside the scope of this paper.

Finally, the movement adaption strategies $\mathcal{H}$ are :

1.) Pseudo-Persistence $\mathcal{P}_F$ with Penalization F

2.) Movement up to the Critical Edge and Stopping $\mathcal{S}$

3.) Edge-Reflected Movement $\mathcal{R}$

To keep things concise, we reduce the following examinations to the critical, vertical action a and state component $\varPhi$, respectively. Camera movements are called valid if $\varPhi \in \varUpsilon = [0°; 90°]$ for the resulting vertical camera position $\varPhi$. Actions ending up at $\varPhi = 90°$ are named a^+, those leading to $\varPhi = 0°$ are called a^-. Please note that we can exactly determine having reached 0° or 90° via the camera mechanics whereas all other position specifications are just probabilistic. This is due to the inaccuracy of relative actions which have to be performed by the hardware, e.g. a mobile robot. In particular, we will show the demand of repeating a performed action a which accordingly cannot be done exactly. Thus the repeated action is symbolized by $\tilde{a}$.

1.) The first approach is quite intuitive and obeys mainly the idea of Reinforcement Learning. Here, the actual proceeding concerning the camera movement is to return to the starting point if a required vertical action $\breve{a}$ turns out to be inexecutable *during motion*. Otherwise the action is simply performed. Considering the former, that means we

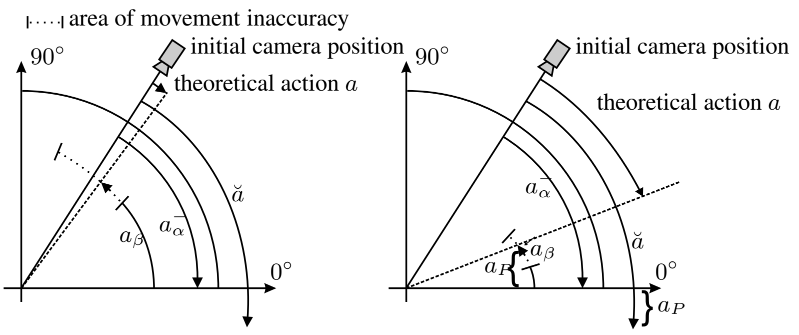

Fig. 2. Exemplary camera movement handling for invalid actions using the pseudo-persistence (left) or edge-reflection method (right). The inaccuracy within the second sub-action a_β is chosen exemplary.

can divide the whole movement process at this time step into two sub-movements indicated by subscripts α and β. So, first an action $a_\alpha = a_\alpha^{+/-}$ physically takes the camera to a critical edge. Then we need to reverse this action in order to go back to the starting point. Regarding the action inaccuracies the rule for this is $a_\beta = -\tilde{a}_\alpha^{+/-}$ (see the left drawing of figure 2). Please remember that we cannot just fix the camera in the starting point for this time step since we don't know about the possible success of a movement before actually trying it (see section 1). Since then the camera position would only change relative to the action inaccuracy, this behavior is called **pseudo-persistence**. To understand the particle handling correctly, remember that the software part in figure 1 is not aware of any information about the hardware's action operability. So each particle of the current state density is moved in a translative manner just according to its represented hypothesis and the proposed action $\breve{a}$. To point out the resulting similarities and differences between propagating the particles in the state representation and the physical action handling, invalid actions (combinations of sub-movements) are mapped to the one *valid*, but still theoretical action a which would result in the corresponding final camera position:

$$a = \begin{cases} a_\alpha^+ - \tilde{a}_\alpha^+ & \text{if } \breve{a} > a_\alpha^+ \\ a_\alpha^- - \tilde{a}_\alpha^- & \text{if } \breve{a} < a_\alpha^- \\ \breve{a} & \text{else} \end{cases} \quad ; \qquad \tilde{\Phi}(\rho_{t+1}) = \begin{cases} \tilde{\Phi}(\rho_t) & \text{if } \tilde{\Phi}(\rho_t) + \breve{a} \notin \Upsilon \\ \tilde{\Phi}(\rho_t) + \breve{a} & \text{else} \end{cases} \tag{7}$$

Above all, that implies that huge amounts of particles might change position even if only pseudo-persistence movement is physically performed.

Knowing that an illegal move leads to a next view from almost the same position, which in general adds very little new information to the classification task, it appears to be a rational and Reinforcement Learning consistent idea to punish those actions during training. So, if using the pseudo-persistence method $\mathcal{P}_F$ and applying the punishment F to the reward function ($r_t = F$ in (3)), it should be possible to learn to avoid those illegal actions during the following evaluation phase. However, that is where the problem occurs with this approach. Since we have no a-priori knowledge about the domain of appearing rewards for valid moves in general it is impossible to establish an optimal F beforehand. So either the determination of adequate values for such a punishment has to be integrated into a far more complex and time consuming learning process [6] or it has to be set generously high by hand. The latter is obviously suboptimal since also legal movements into the border area would be partially disadvantaged because of the approximation mentioned in (5). This means that optimal next best viewpoints might be ignored merely because of their proximity to the illegal area, which is a quite well known problem in Reinforcement Learning. Even originally valid actions within an episode during training could be negatively affected if an invalid action follows later on in the same episode, regarding the return (2) with $\gamma > 0$. Nevertheless, this is the approach a problem-unspecific Reinforcement Learning method would apply. Thus, chapter 4 will also show the classification results when applying various punishment terms F.

2.) The second approach tackles the afore mentioned problem of unadjusted punishment terms by avoiding their occurrence in general. This is simply done by **stopping a critical camera movement** if either the north pole or the hemisphere's plane is reached, thus $a_\beta = 0$. This way, non-executable actions cannot be avoided either, but each of them can again be uniquely mapped to a valid action and can be rated by the usual Reinforcement Learning reward (3). Again, density particles have to be handled with the same procedure, yielding the following directions:

$$a = \begin{cases} a_\alpha^+ & \text{if } \breve{a} > a_\alpha^+ \\ a_\alpha^- & \text{if } \breve{a} < a_\alpha^- \\ \breve{a} & \text{else} \end{cases} ; \quad \tilde{\Phi}(\rho_{t+1}) = \begin{cases} 90° & \text{if } \tilde{\Phi}(\rho_t) + \breve{a} > 90° \\ 0° & \text{if } \tilde{\Phi}(\rho_t) + \breve{a} < 0° \\ \tilde{\Phi}(\rho_t) + \breve{a} & \text{else} \end{cases} \tag{8}$$

Please note that stopping at the critical edges differs from most other, possibly randomly chosen successive actions in the way that it is deterministic. In particular, each particle can be propagated in exactly the same manner a camera would move when being located at this particle's parameter hypotheses - in fact without knowing the real outcome of the camera movement beforehand.

3.) The obvious drawback of the previous approach is the preference of edges as arrival points for the camera movement. Since we always permit relative vertical actions within the range of $[-90°; 90°[$, theoretically every second move ends up at one of those two edges during the Reinforcement Learning training phase, since we only perform random actions here. Intuitively, this might result in a heavily unbalanced knowledge base built during this training.

In order to overcome this potential barrier as well, our proposed idea for an optimized action handling is the **edge-reflected** movement and particle propagation, respectively. As the name says, a *remaining* action potential a_P is continued in the reverse direction whenever reaching a critical edge at runtime, thus $|a_\beta| = |\breve{a}| - |a_\alpha|$. The corresponding illustration can be found in the right drawing of figure 2. This way, each valid movement has exactly two related movements it can emerge from, the originally valid and the reflected one. In contrast to the edge-stopping method this relation is one-to-one now. This results in a uniform probability distribution for reaching any viewpoint in the next time step. Additionally, the deterministic behavior is assured once more and the calculation instructions can be expressed as:

$$a = \begin{cases} a_\alpha^+ - (\breve{a} - \tilde{a}_\alpha^+) & \text{if } \breve{a} > a_\alpha^+ \\ a_\alpha^- + (\tilde{a}_\alpha^- - \breve{a}) & \text{if } \breve{a} < a_\alpha^- \\ \breve{a} & \text{else} \end{cases} ;$$

$$\tilde{\Phi}(\rho_{t+1}) = \begin{cases} 180^\circ - \tilde{\Phi}(\rho_t) - \breve{a} & \text{if } \tilde{\Phi}(\rho_t) + \breve{a} > 90^\circ \\ -\tilde{\Phi}(\rho_t) - \breve{a} & \text{if } \tilde{\Phi}(\rho_t) + \breve{a} < 0^\circ \\ \tilde{\Phi}(\rho_t) + \breve{a} & \text{else} \end{cases} . \qquad (10)$$

4 Experimental Results

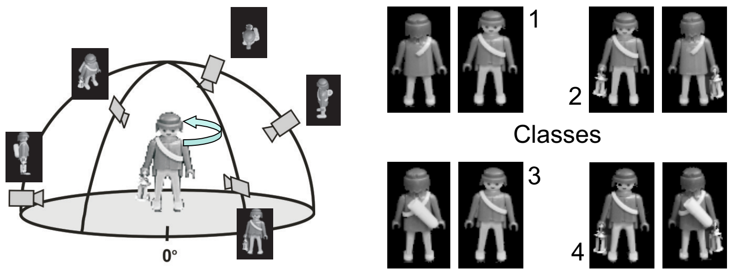

Fig. 3. Views of the toy object classes

For experimental evaluation of our active object recognition task, we have chosen four classes of real toy figures, discriminable by a quiver and a lamp. Corresponding to the 2-dimensional approach of this paper, figure 3 shows some examples of images that can be acquired from viewpoints situated on the hemisphere around the object. Please note that every recorded image was superposed by a uniform Gaussian noise before processing it any further. This way overall classification results are equally downgraded, but differences in the various applied methods for action handling will be more distinguishable. For classification itself image features are extracted by a Principal Component Analysis using only the ten best eigenvectors in the transformation matrix.

Reinforcement Learning training was done with 50 episodes per class, each containing eight camera actions in between nine recorded images, whereas all actions were performed completely randomly. These training stages were separately executed for every combination of action handling $\mathcal{H} \in \{\mathcal{P}_{-1}, P_0, P_{0.5}, S, R\}$ and weighting $\gamma \in \{0, 0.5\}$ introduced in (2). During the evaluation phase, 50 exploiting episodes were performed for each class based on the particular knowledge base built up during training. The kernel parameter D, which varies $K(x)$ in (5), was set to a well-proven value of $D = 10$ (see [5]). Figure 4 shows the classification results we achieved with the various combinations after the fusion of the information data of n images. Results are displayed up to a step width of $n = 5$ since later results just converge to a saturation and thus would just reduce clarity. In any case, the most important values are those of early steps ($n = 2, 3$) as they show the immediate gain or loss in classification certainty most real decisions would rely on. Concerning this, it is obvious that the proposed new methods of $\mathcal{H} = \mathcal{S}, \mathcal{R}$ clearly outperform those based on a punishment for invalid camera actions. Nevertheless, especially the right chart of figure 4 points out that there are indeed penalization terms (like $F = 0$) that can achieve a comparably high learning quality. But as mentioned, this comes at a price of having a priori knowledge about the range of the regular Reinforcement Learning rewards, and then it still needs some experience and object specific previous knowledge to optimize the punishment value. Those preconditions cease to apply when using the improved edge-reflecting or edge-stopping method.

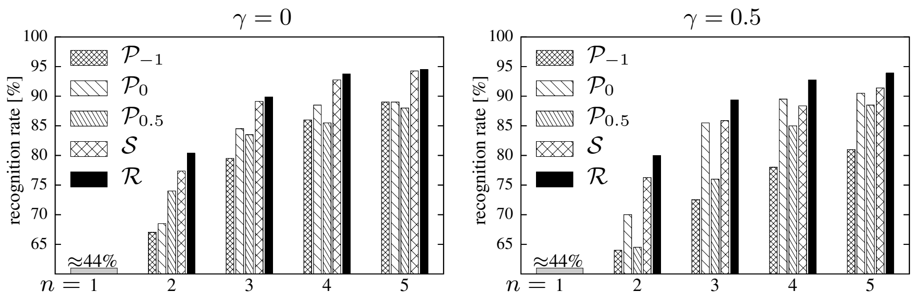

Fig. 4. Recognition rate [%] after n planned actions and information fusion. The various columns display the influence of the compared action handling methods $\mathcal{H}$ for invalid action demands. The left figure ($\gamma = 0$) considers one-step returns in the Reinforcement Learning, the right one ($\gamma = 0.5$) represents ahead looking returns.

Since results for $\mathcal{H} = \mathcal{S}$ are quite close to those of $\mathcal{H} = \mathcal{R}$ regarding the *recognition rates*, we should additionally pay attention to the *pose estimation* accuracy which might turn the balance then. To evaluate this, in figure 5 we additionally depicted the localization error of the vertical object pose after n steps of image fusion. Therefore, we concentrated on $\mathcal{H} = \mathcal{S}, \mathcal{R}$ and again distinguished between $\gamma = 0$ and $\gamma = 0.5$. These results explicitely show that, for pose estimation accuracy as well, the edge-reflected method is the one to prefer. As with the classification results, we achieve obvious enhancements mainly within the early steps. Obviously, when using $\mathcal{H} = \mathcal{S}$, the accumulation of particles at one of the critical edges for invalid camera movement demands takes effect, resulting in an unbalanced distribution and thus a more imprecise pose representation. Regarding the weighting γ we can postulate that its value does not affect the general ranking of the three proposed action handling methods. Thus, it can be considered a noncritical parameter for the method selection decision.

5 Summary and Future Work

The focus of this work was on object recognition tasks with predefined constraints in valid camera positions and thus camera movements. Therefore, we emphasized the problem of providing an immediate alternative camera movement direction when the requested one turns out to be non-executable. For that purpose, we compared three

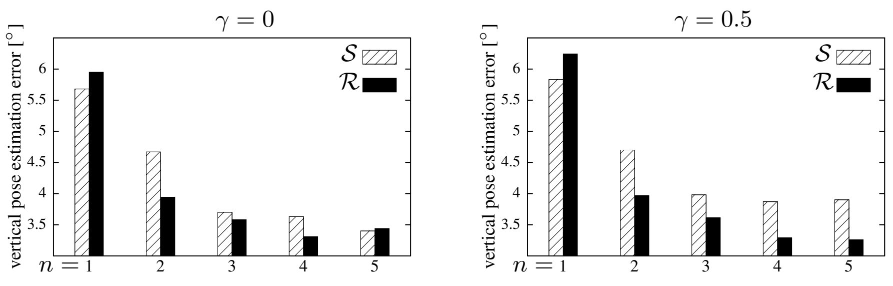

Fig. 5. Estimation error in the vertical object pose [°] after n planned actions and information fusion. Values are compared for the edge-stopping ($\mathcal{H} = \mathcal{S}$) and the edge-reflecting ($\mathcal{H} = \mathcal{R}$) action handling methods. Again, the left figure shows results for $\gamma = 0$ and the right one for $\gamma = 0.5$ in the Reinforcement Learning return function (2).

different action handling approaches and their influence on the underlying state density representation. Summing up the visualized results, the edge-reflecting version for handling non-executable actions turned out to outperfor the others in practice, according to our presumption. We pointed out the advantage of the edge-stopping and the edge-reflecting methods compared to other approaches introduced in the literature. In particular, the former explicitly support our demand of classification with a minimal number of camera movements since they avoid replanning and thus discarding already performed movements.

Further work on this topic will concentrate on the assignment of costs to the various movement actions, bringing the proposed problem to a completely new dimension. One of the main questions is whether the applied action handling methods can address cost integration at all. In any case, adjustment should nevertheless be a complex procedure.

References

1. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking. *IEEE Transactions of Signal Processing*, 50:174–188, 2002.
2. K.H. Chang and M. Edhala. Execution Error Recovery for Planning Systems. In *In Proceedings of the 7th Annual International Phoenix Conference on Computers and Communications*, pages 492–496, Scottsdale, USA, 1988.
3. F. Deinzer, J. Denzler, and H. Niemann. On Fusion of Multiple Views for Active Object Recognition . In *Pattern Recognition – 23rd DAGM Symposium* , pages 239–245, Munich, Germany, 2001.
4. F. Deinzer, J. Denzler, and H. Niemann. Viewpoint Selection - Planning Optimal Sequences of Views for Object Recognition . In *Computer Analysis of Images and Patterns - CAIP '03* , number 2756 in Lecture Notes in Computer Science , pages 65–73, Groningen, Netherlands, 2003.
5. F. Deinzer, Ch. Derichs, and H. Niemann. Aspects of optimal viewpoint selection and view-point fusion. In *Computer Vision - ACCV 2006*, volume 2, pages 902–912, Hyderabad, India, Januar 2006.
6. P. Geibel and F. Wysotzki. Risk-Sensitive Reinforcement Learning Applied to Control under Contstraints. *Journal of Artificial Intelligence Research*, 24:81–108, 2005.
7. M. Isard and A. Blake. 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. C. A. Knoblock. Planning, Executing, Sensing, and Replanning for Information Gathering. In *In Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1686–1693, Montreal, Canada, 1995.
10. A. Ranganathan and S. Koenig. A Reactive Robot Architecture with Planning on Demand. In *In Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, volume 2, pages 1462–1468, Las Vegas, USA, 2003.
11. R.S. Sutton and A.G. Barto. *Reinforcement Learning*. A Bradford Book, Cambridge, London, 1998.
12. A. Törn and A. Žilinskas. *Global Optimization*, volume 350 of *Lecture Notes in Computer Science*. Springer, Heidelberg, 1987.
13. R. van der Krogt, M. de Weerdt, and C. Witteveen. A Resource Based Framework for Planning and Replanning. In *In Proceedings of the IEEE/WIC International Conference on Intelligent Agent Technology*, pages 173–186, Halifax, Canada, 2003.