From K. Franke et al., *Pattern Recognition, 28th DAGM Symposium*, Springer, 2006, (pp. 536–545).

Multi-Step Multi-Camera View Planning for Real-Time Visual Object Tracking

B. Deutsch*, S. Wenhardt, H. Niemann

Chair for Pattern Recognition, University of Erlangen-Nuremberg,
{deutsch,wenhardt,niemann}@informatik.uni-erlangen.de,
WWW home page: http://www5.informatik.uni-erlangen.de

Abstract. We present a new method for planning the optimal next view for a probabilistic visual object tracking task. Our method uses a variable number of cameras, can plan an action sequence several time steps into the future, and allows for real-time usage due to a computation time which is linear both in the number of cameras and the number of time steps. The algorithm can also handle object loss in one, more or all cameras, interdependencies in the camera's information contribution, and variable action costs.
We evaluate our method by comparing it to previous approaches with a prerecorded sequence of real world images.

1 Introduction

This paper describes an enhanced method for selecting a sequence of *optimal sensor actions* for a probabilistic state estimation system. The optimal actions are those that minimize the expected uncertainty of the state probability distribution function, measured by the expected state entropy. We apply this method for view planning in an object tracking task. In this task, the sensor actions affecting the view are the camera zoom settings. However, this method is not restricted to zoom planning. It can also handle other camera actions, such as panning, tilting or translation, and is equally applicable to other active state estimation tasks.

A large amount of research in the area of view planning exists for object recognition tasks [1–3], in which the active selection of views directly reduces the uncertainty in classification. For active object tracking many works involve the changing of zoom settings [4–6]. However, these methods keep the size of the object in the images constant, as opposed to minimizing the uncertainty of the estimate of the object position. Previous work in uncertainty reduction includes [7], in which a subset from a set of sensors is chosen to meet certain threshold criteria. A more general approach is followed in [8], where actions are chosen which maximally reduce the *expected entropy* of the object position in space as a measure of positional uncertainty.

Previous work [9] has extended this approach to optimize a sequence of actions for view planning. This extension allows variable action costs, such as occur due to limited camera zoom motor speeds, to be incorporated into the optimization. Potential object

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

loss is dealt with by evaluating each possible sequence of object visibility in a *visibility tree* (see section 2 and Fig. 1). A subsequent improving work [10] aimed to address several shortcomings of this method, such as the inability to efficiently handle an object visible in only a subset of the cameras, with the use of the *sequential Kalman filter*. By applying a visibility tree to each camera separately, the computational cost is linear in the number of cameras, and partial visibility can be handled.

The main problem with the still remaining visibility tree is that its size, and therefore the computation costs for view planning, are still exponential in the number of time steps. In this work, we propose a new method, which flattens this visibility tree, thus achieving linear runtime.

We test our approach with a prerecorded image sequence from up to three cameras. This sequence is scaled with a variable scale factor to simulate a changing focal length, while allowing several algorithms to be compared independently on the same data.

This paper is organized as follows: The next section reviews the current state of view planning for active object tracking and describes the notation used in this work. Section 3 details the method of visibility tree linearization to reduce computation time. Section 4 covers the experiments, comparing the previous methods to our new one. The last section summarizes and concludes this paper, and lists potential future work.

2 Kalman filter and action selection

Tracking an object in 3D is defined as a state estimation problem, which we solve with the well-known Kalman filter [11], extended to handle sensor actions. To accommodate the non-linear nature of the observation functions involved, we use the *extended Kalman filter* [12, 13], although this distinction is not relevant for this work.

The (extended) Kalman filter estimates the state of a discrete-time dynamic system. At time t, the state is described in the state vector $\boldsymbol{q}_t \in \mathbb{R}^n$. The cameras generate an observation $\boldsymbol{o}_t \in \mathbb{R}^m$ from the state. The state change and observation equations are

$$\boldsymbol{q}_t = \boldsymbol{f}(\boldsymbol{q}_{t-1}) + \boldsymbol{w} \quad , \quad \boldsymbol{o}_t = \boldsymbol{h}(\boldsymbol{q}_t, \boldsymbol{a}_t) + \boldsymbol{r} \tag{1}$$

where $\boldsymbol{f}(\cdot) \in \mathbb{R}^n$ is the state transition function and $\boldsymbol{h}(\cdot,\cdot) \in \mathbb{R}^m$ the observation function, based on the cameras' projection function. $\boldsymbol{w}$ and $\boldsymbol{r}$ are normal error processes with zero mean and covariance matrices $\boldsymbol{W}$ and $\boldsymbol{R}$.

For active object tracking, the observation function also contains an *action* parameter $\boldsymbol{a}_t \in \mathbb{R}^l$, which combines all influences on the observation process, such as zooming, panning, tilting, or translating the camera. For this work, we focus on zoom planning as the camera action. The action is performed *before* an observation is made.

Given the noise terms, the state must be estimated each time step. Specifically, we must calculate the state probability distribution $p(\boldsymbol{q}_t|\langle\boldsymbol{o}\rangle_t, \langle\boldsymbol{a}\rangle_t)$, given the sequences of all observations $\langle\boldsymbol{o}\rangle_t$ and all actions $\langle\boldsymbol{a}\rangle_t$ taken up to, and including, time t. Within the Kalman filter framework, this distribution is assumed to be a normal, or Gaussian, distribution.

We use the following Kalman filter notation: $\widehat{\boldsymbol{q}}_t^-$ and $\widehat{\boldsymbol{q}}_t^+$ are the *a priori* and *a posteriori* state estimate means at time t. $\boldsymbol{P}_t^-$ and $\boldsymbol{P}_t^+$ are the covariance matrices for

the errors of the *a priori* and *a posteriori* state estimates. The extended Kalman filter performs the following steps for each time-step t:

1. Prediction of the state mean $\widehat{\boldsymbol{q}}_t^-$ and covariance $\boldsymbol{P}_t^-$:

$$\widehat{\boldsymbol{q}}_t^- = \boldsymbol{f}(\widehat{\boldsymbol{q}}_{t-1}^+) \quad , \quad \boldsymbol{P}_t^- = \boldsymbol{F}_t \boldsymbol{P}_{t-1} {\boldsymbol{F}_t}^{\mathrm{T}} + \boldsymbol{W} \tag{2}$$

2. Computation of the filter gain $\boldsymbol{K}_t$:

$$\boldsymbol{K}_t = \boldsymbol{P}_t^- {\boldsymbol{H}_t}^{\mathrm{T}}(\boldsymbol{a}_t) \left(\boldsymbol{H}_t(\boldsymbol{a}_t) \boldsymbol{P}_t^- {\boldsymbol{H}_t}^{\mathrm{T}}(\boldsymbol{a}_t) + \boldsymbol{R}\right)^{-1} \tag{3}$$

3. State update with the observation $\boldsymbol{o}_t$:

$$\widehat{\boldsymbol{q}}_t^+ = \widehat{\boldsymbol{q}}_t^- + \boldsymbol{K}_t \left(\boldsymbol{o}_t - \boldsymbol{h}(\widehat{\boldsymbol{q}}_t^-, \boldsymbol{a}_t)\right) \quad , \quad \boldsymbol{P}_t^+(\boldsymbol{a}_t) = \left(\boldsymbol{I} - \boldsymbol{K}_t \boldsymbol{H}_t(\boldsymbol{a}_t)\right) \boldsymbol{P}_t^- \tag{4}$$

$\boldsymbol{F}_t$ and $\boldsymbol{H}_t(\boldsymbol{a}_t)$ denote the Jacobians of $\boldsymbol{f}(\cdot)$ and $\boldsymbol{h}(\cdot,\cdot)$ at $\widehat{\boldsymbol{q}}_{t-1}^+$ and $\widehat{\boldsymbol{q}}_t^-$ respectively, to account for non-linear functions. Since $\boldsymbol{H}_t(\boldsymbol{a}_t)$ depends on the selected action $\boldsymbol{a}_t$, the *a posteriori* state covariance $\boldsymbol{P}_t^+$ does, too. If no valid observation $\boldsymbol{o}_t$ is made at time t, the update step cannot be performed and $\widehat{\boldsymbol{q}}_t^+, \boldsymbol{P}_t^+$ are equal to $\widehat{\boldsymbol{q}}_t^-, \boldsymbol{P}_t^-$.

Since we are interested in obtaining the most information about the state, we need to determine the optimal action $\boldsymbol{a}_t^*$ where the uncertainty is lowest. In [8], this is achieved by finding the action where the *entropy* of the *a posteriori* probability distribution $p(\boldsymbol{q}_t | \langle \boldsymbol{o} \rangle_t, \langle \boldsymbol{a} \rangle_t)$ is minimal. As this is a normal distribution $\mathcal{N}(\widehat{\boldsymbol{q}}_t^+, \boldsymbol{P}_t^+)$, the entropy is equal to $\log\left(|\boldsymbol{P}_t^+|\right)$ up to constant terms and factors. These constants can be ignored during optimization. The entropy depends on the covariance $\boldsymbol{P}_t^+$, and therefore on $\boldsymbol{a}_t$, but *not* on $\boldsymbol{o}_t$. This allows to determine $\boldsymbol{a}_t^*$ *before* making an observation.

The problem of visibility in object tracking is also addressed in [8]. An observation $\boldsymbol{o}_t$, containing the position of the object being tracked in each camera, is only valid if the object is in *every* camera's field of view. We refer to an $\boldsymbol{o}_t \in \mathbb{R}^m$ lying outside of the field of view as a *non-visible observation*. The probability w that the object lies in the field of view of all cameras can be calculated from the predicted observation for any action $\boldsymbol{a}_t$ by integrating the probability density of the observation over the camera sensor. The expected entropy for an action is then the weighted combination of the entropies for each case of visibility, or for optimization purposes

$$\boldsymbol{a}_t^* = \arg\min_{\boldsymbol{a}_t} \left(w \cdot \log\left(|\boldsymbol{P}_t^+(\boldsymbol{a}_t)|\right) + (1-w) \cdot \log\left(|\boldsymbol{P}_t^-|\right)\right) \tag{5}$$

This action selection has been extended to a sequence of future actions in [9]. For a sequence, w is extended to a *visibility tree*, which is a binary tree in which each branching represents a visible or non-visible outcome. The entropy for each possible sequence of visible or non-visible observations is calculated and then summed up by walking up the tree again.

An example of such a tree for two time steps is shown in Fig. 1. In this example, each node represents one of the two possible *a posteriori* covariance matrices. Light nodes are the predicted result of a visible observation, dark nodes of a non-visible one. In each time step, the probabilities of visibility or non-visibility are given by the w and

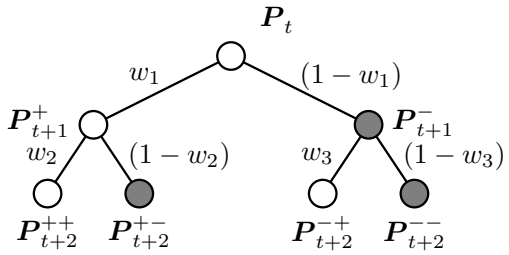

Fig. 1. The visibility tree for two time steps. The calculation starts at the top at time t. The nodes represent possible *a posteriori* covariance matrices for subsequent time steps. Light nodes are the result of a visible observation, dark nodes of a non-visible one. See the text for a deeper discussion of the visibility tree.

$(1-w)$ terms (note that $w_2 \neq w_3$). The total expected entropy is the weighted sum of the four entropies based on the four final covariances. The covariances are tagged with the visibility sequence they resulted from. For example, $\boldsymbol{P}^{+-}_{t+2}$ is obtained by first assuming a visible observation $(+)$ followed by a non-visible one $(-)$. The final expected entropy for this example is (ignoring constant terms):

$$\begin{aligned} & w_1 w_2 \log\left(|\boldsymbol{P}^{++}_{t+2}|\right) + w_1(1-w_2)\log\left(|\boldsymbol{P}^{+-}_{t+2}|\right) \\ + & (1-w_1)w_3 \log\left(|\boldsymbol{P}^{-+}_{t+2}|\right) + (1-w_1)(1-w_3)\log\left(|\boldsymbol{P}^{--}_{t+2}|\right) \end{aligned}$$

This action selection can also be performed with the *sequential Kalman filter* [12]. The sequential Kalman filter is a sequential evaluation method of the standard Kalman filter, in which the sensors are processed in sequence. This method is equivalent to providing each sensor with its private Kalman filter and can be used when the observation noise for each sensor is uncorrelated. The *a posteriori* distribution of one sensor's filter is used as the *a priori* distribution for the next. Fig. 2 gives an overview of the sequential state estimation process. The advantage of the sequential Kalman method is that the visibility is no longer determined by the object being in the field of view of all cameras; partially visible observations can also be handled by skipping a camera's filter if the object is not visible.

The disadvantages are that the sensor noise must be uncorrelated between sensors for the sequential Kalman filter, and that the result may depend on the order in which the sensors are processed. While the traditional Kalman filter with linear prediction and observation models does not depend on the order of the sensors, the extended Kalman filter obtains the Jacobians $\boldsymbol{F}_t$ and $\boldsymbol{H}_t(\boldsymbol{a}_t)$ by deriving at the current state estimate. Since this state estimate is affected by the observations from previous sensors, the Jacobians will differ if different sensors are processed beforehand. However, this difference is comparable to the differences encountered in the Jacobians in the non-sequential extended Kalman filter, where the Taylor expansion is also performed on the current best estimate instead of the true state, and is usually ignored.

The sequential Kalman filter is used for multi-step action selection in [10]. Each camera action is optimized independently by the method of [9], on the assumption that changing the zoom level in one camera will not influence the information gained in

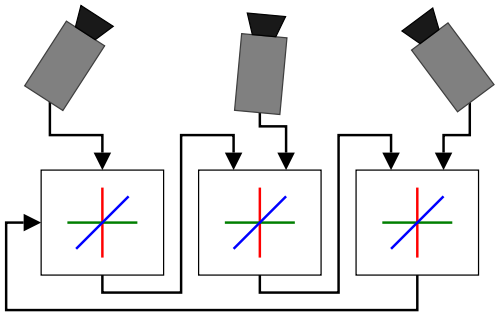

Fig. 2. The sequential Kalman filter. Each camera adds its observation to the state estimate in sequence. The *a posteriori* state estimate of the last camera is transformed to the *a priori* estimate of the first camera on the next time step.

another camera. This sequential multi-step action selection still uses the visibility tree from the original multi-step method. Partial visibility (in some, but not all cameras) is handled explicitly during tracking and implicitly in the optimization process.

However, this gives rise to another problem. During the action planning step, the predicted uncertainty is calculated by the contribution of a single camera, disregarding the others. In the Kalman filter's update step (equations (3) and (4)), $\boldsymbol{P}_t^+$ is derived from $\boldsymbol{P}_t^-$ using only the observation function of this single camera. This leads to an overestimation of the *a posteriori* covariance in the planning phase, which results in an overly cautious action planning. This omission can be rectified by including the effects of the other cameras on $\boldsymbol{P}_t^+$. However, to avoid another visibility dependency and keep the visibility tree small, these other cameras must follow actions which are assumed to guarantee an observation, which still overestimates the covariance during planning.

3 Linearization of the visibility tree

For the multi-step multi-camera sequential Kalman filter, as seen in Fig. 3, the output of each individual filter during tracking (such as the one marked in the figure) becomes the input of the next one. This output is the probability density of the state estimate at this time, with the observation of this camera embedded if it was visible, and skipped if it was not. For view planning, this means that each individual filter has *two* possible outputs which need to be considered, with covariance matrices $\boldsymbol{P}_t^+$ and $\boldsymbol{P}_t^-$, since the expected state mean is the same in both cases.

The previous methods have handled these with a visibility tree, as detailed in the last section. Spanning a visibility tree for the full sequential filter is prohibitive, since the size of the tree is exponential in the number of cameras and time steps. The solution which uses the sequential Kalman filter reduces this complexity somewhat by optimizing actions for each camera separately. However, the visibility tree size is still exponential in the number of time steps, and the expected covariance is overestimated, as mentioned previously.

The visibility tree can be flattened by closely looking at the two probability distributions that can result in one time step. The two resulting distributions are Gaussian

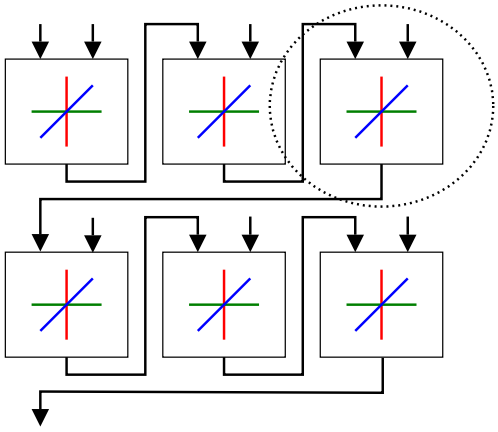

Fig. 3. The sequential Kalman filter during multi-step evaluation. The output of one individual filter becomes the input of the next filter in the same time step or the first filter in the next time step (after the state transformation, not shown here). The dashed circle marks one individual filter.

distributions and differ only in their covariances, $\boldsymbol{P}_t^+$ and $\boldsymbol{P}_t^-$, but not in their means, as the expected mean does not depend on the visibility in the view planning step. Since we know the probability $w \in [0, 1]$ that one of these two distributions will be the actual output, we can consider them to be two components of a mixture distribution $\mathcal{M}$,

$$\mathcal{M} = w \cdot \mathcal{N}(\widehat{\boldsymbol{q}}_t^+, \boldsymbol{P}_t^+) + (1 - w) \cdot \mathcal{N}(\widehat{\boldsymbol{q}}_t^+, \boldsymbol{P}_t^-), \tag{6}$$

which describes the expected distribution of the state after performing action $\boldsymbol{a}_t$. Since this is an unimodal distribution, we can approximate it by a new Gaussian distribution with the same covariance. As known in statistics, the covariance matrix of $\mathcal{M}$ is:

$$\boldsymbol{P}_t^\circ = w \cdot \boldsymbol{P}_t^+ + (1 - w) \cdot \boldsymbol{P}_t^- \tag{7}$$

Therefore, our approximating Gaussian is of the form $\mathcal{N}(\widehat{\boldsymbol{q}}_t^+, \boldsymbol{P}_t^\circ)$.

This distribution can now be used as an estimate of the resulting state probability distribution after visibility is considered. Note that the Gaussian distribution is an approximation of the mixture distribution, with same mean and covariance, but with different density functions.

The benefits of this approach are obvious. Since each individual filter in a multi-step multi-camera now only results in a single output distribution *during view planning as well*, the effects of an action can now be calculated in linear time in the number of cameras times the number of time steps. This can be seen in Fig. 3, which is now equally valid for the view planning process. Since the actions are optimized for all cameras at the same time, this approach also fully handles dependencies in the actions of different cameras, unlike the previous sequential method which used a separate optimization.

Although the entropy of the final expected distribution, based on $\log\left(|\boldsymbol{P}_{t+k}^\circ|\right)$, differs from the actual expected entropy, the *behavior* of the system is close enough such that the optimal action can be searched for. This can be seen in the next section, where

Fig. 4. The views of the test setup from all three cameras at the same time. The colored bottle is tracked as it turns on the turntable. The calibration pattern is used to initially calibrate all cameras, it is not used during tracking.

several approaches are compared on the same data. The behaviour is visible when comparing the original single-step approach to the one based on $\boldsymbol{P}^{\circ}_{t+k}$: both approaches are very similar when no visibility problems are encountered.

4 Experiments

We test our new method on a recorded video sequence, shown in Fig. 4. Three cameras take a high-resolution image of the scene, consisting of an object moved by a turntable. The cameras are calibrated to a global coordinate system with the calibration pattern, which is not used for the tracking process. The object is tracked with a color histogram tracker [14].

The prerecorded images allow several view planning methods to be compared on the same data. However, this precludes the effect of the camera zoom on these images, unless we simulate this zoom on the original images. The original images are 640 by 480 pixels, but the tracking process uses images of size 320 by 240 pixels. To obtain this size reduction, and simulate the camera zoom, we scale and crop the original image by an amount which depends on the associated zoom level. When fully zoomed in, the transformation only crops a 320 by 240 pixel image from the center of the original. As the zoom level decreases, the cropped region becomes larger and is subsequently scaled to the correct size. When fully zoomed out, the original images are only scaled, no cropping occurs. Using such a reduced image size ensures that, even when fully zoomed in, no upsampling artifacts occur.

The advantages of multi-step view planning have been detailed in [9] and [10]; no detailed comparison to single-step planning will be made here. We will focus primarily on a direct comparison between the previous planning method (cf. section 2), which used separate optimization, and the newly proposed one (cf. section 3).

We test both systems on the same data, as detailed above. Each planning system recommends the next view for the tracking system in the form of a set of actions. The optimal action set is planned with the global optimization technique of Adaptive Random Search [15], evaluating a total of 400 separate action sequences per time frame. Each action sequence contains the next actions for each camera for the next time steps.

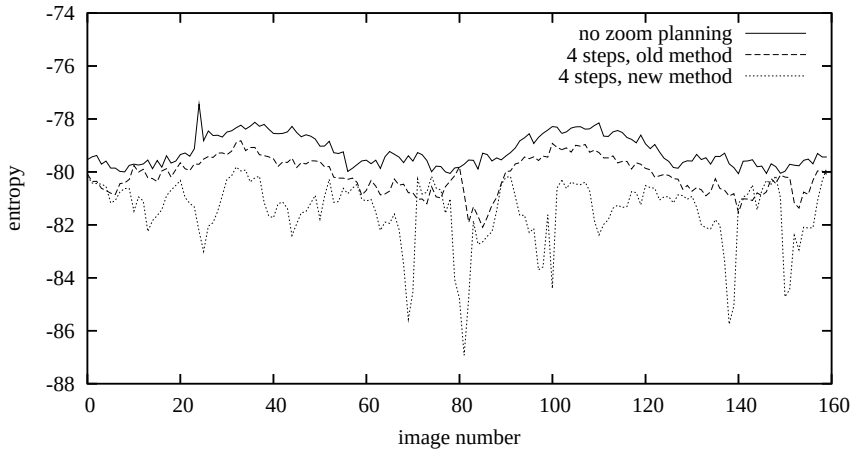

Fig. 5. The entropy as $\log\left(|\boldsymbol{P}_t^+|\right)$ with three cameras plotted for image number t. Shown are the plots for no planning, planning 4 steps ahead using the old, independently optimizing method, and the new proposed method.

In our experiments, we planned one to four steps ahead with up to three cameras, so each action sequence contains up to 12 separate zoom settings.

Fig. 5 shows the entropy of the object position during the tracking process by measuring $\log\left(|\boldsymbol{P}_t^+|\right)$. Note that $\boldsymbol{P}_t^+$ is the result of the actual tracking phase, and not a result from the view planning. However, the view planning influences the tracking results both positively (by providing zoomed in views, reducing the entropy) and negatively (by zooming in too far, causing object loss and raising the entropy). This experiment uses three cameras. The plots show the results for unplanned tracking, planning 4 steps with the independent optimization, and 4 steps with the new method.

Both planning methods result in an uncertainty, measured by the entropy, which is lower than when no zoom planning is used. But it can clearly be seen that the original approach with separate optimization still results in a higher uncertainty than the new approach. Since the expected *a posteriori* covariance is overestimated, the actions planned by this system are not as aggressive as those calculated with the new system. The new system plans views which are zoomed further in, which lowers the entropy, in many cases by quite a large amount. Only in a few cases (near image numbers 72 and 142) do these zoom levels prove overconfident, resulting in short object losses and a higher entropy than the original approach.

Fig. 6 compares the new multi-step approach, looking 4 time steps into the future, to the single step approach. The experiments are the same as in Fig. 5. Both plots are very similar, showing that the behaviour of the view planning using the combined covariance $\boldsymbol{P}_{t+k}^{\circ}$ is very close to the original behaviour, if at times slightly worse due to the more cautious approach of multi-step planning. The most notable differences occur around image numbers 101 and 153 in the right half of the plot. The object starts moving out of the field of view of one or even several cameras. The single step planning is caught off

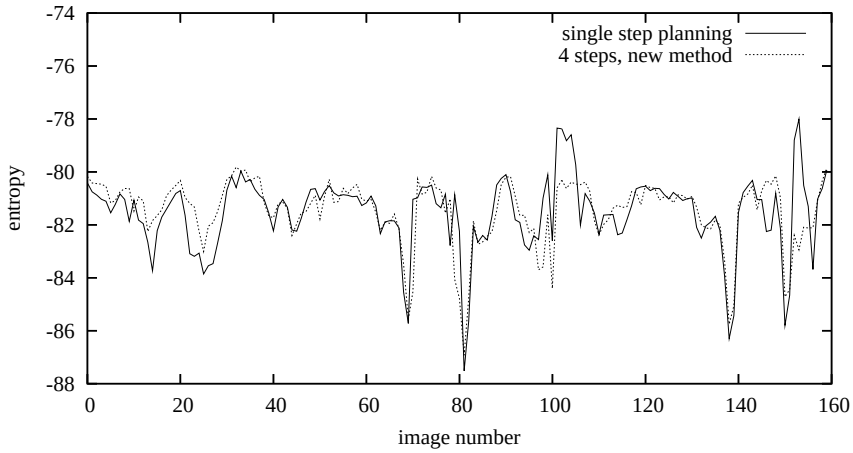

Fig. 6. The entropy as $\log\left(|\boldsymbol{P}_t^+|\right)$ with three cameras plotted for image number t. The single-step planning method and the new proposed multi-step method for 4 time steps are compared.

Table 1. Computation time. Shown are the average computation times for evaluating a single action sequence for one to four time steps, two and three cameras, with the old and the new method. All times in ms.

Time steps planned	old method		new method	
	two cameras	three cameras	two cameras	three cameras
1 step	0.115	0.173	0.095	0.124
2 steps	0.390	0.597	0.163	0.224
3 steps	0.945	1.457	0.247	0.333
4 steps	2.055	3.022	0.293	0.431

guard by this, resulting in object loss and large spikes in the uncertainty. The multi-step approach is able to predict the object loss better and avoids these spikes.

Another important aspect is the comparison of running times. The running times per frame for several different cases are given in table 1. Note that while the original algorithm required exponential time per frame (yet was linear in the number of cameras due to the independent treatment), the new approach is about linear in the number of time steps as well. All times are in milliseconds on a Pentium IV processor at 2.66 GHz.

5 Conclusion

We have presented a new approach for multi-step multi-camera view planning for object tracking, based on the method of entropy minimization. This approach runs in linear time in the number of cameras and time steps. It can incorporate action costs through the evaluation of several time steps into the future. It is capable of handling a variable number of cameras, partial visibility, and interdependence in the camera actions. The

general nature of this approach allows it to be applied to a wide variety of active state estimation problems outside of visual object tracking.

Additional work will focus on expanding the action space to also allow camera pan and tilt motions. Another topic is the combination of view planning for tracking with view planning for other tasks, such as object reconstruction or object recognition.

References

1. Paletta, L., Pinz, A.: Active object recognition by view integration and reinforcement learning. Robotics and Autonomous Systems **31, Issues 1-2** (2000) 71–86
2. Denzler, J., Brown, C.: Information Theoretic Sensor Data Selection for Active Object Recognition and State Estimation. IEEE Transactions on Pattern Analysis and Machine Intelligence **24** (2002) 145–157
3. Deinzer, F., Denzler, J., Niemann, H.: Viewpoint Selection – Planning Optimal Sequences of Views for Object Recognition. In: Computer Analysis of Images and Patterns – CAIP 2003. LNCS 2756, Heidelberg (2003) 65–73
4. Fayman, J., Sudarsky, O., Rivlin, E., Rudzsky, M.: Zoom tracking and its applications. Machine Vision and Applications **13** (2001) 25–37
5. Tordoff, B., Murray, D.: Reactive zoom control while tracking using an affine camera. In: Proc 12th British Machine Vision Conference, September 2001. Volume 1. (2001) 53–62
6. Micheloni, C., Foresti, G.L.: Zoom on Target While Tracking. In: Proceedings of the International Conference on Image Processing. Volume 3., Genua, Italy (2005) 117–120
7. Kalandros, M.K., Pao, L.Y., Ho, Y.: Randomization and super-heuristics in choosing sensor sets in target tracking applications. In: Proc. IEEE Conf. Decision and Control, Phoenix, AZ (1999) 1803–1808
8. Denzler, J., Zobel, M., Niemann, H.: Information Theoretic Focal Length Selection for Real-Time Active 3-D Object Tracking. In: International Conference on Computer Vision, Nice, France (2003) 400–407
9. Deutsch, B., Zobel, M., Denzler, J., Niemann, H.: Multi-Step Entropy Based Sensor Control for Visual Object Tracking. In: Pattern Recognition, 26th DAGM Symposium, Tübingen, Germany (2004) 359–366
10. Deutsch, B., Deinzer, F., Zobel, M., Denzler, J.: Multi-Step Active Object Tracking with Entropy Based Optimal Actions Using the Sequential Kalman Filter. In Araújo, H., Vieira, A., Braz, J., Encarnação, B., Carvalho, M., eds.: Proceedings of the International Conference on Image Processing. Volume 2., Setúbal, Portugal, INSTICC Press, Setúbal, Portugal (2005) 169–176
11. Kalman, R.: A new approach to linear filtering and prediction problems. Journal of Basic Engineering (1960) 35–44
12. Chui, C.K., Chen, G.: Kalman Filtering. Springer, Heidelberg (1991)
13. Bar-Shalom, Y., Fortmann, T.: Tracking and Data Association. Academic Press, Boston, San Diego, New York (1988)
14. Pérez, P., Hue, C., Vermaak, J., Gangnet, M.: Color-based probabilistic tracking. In: Proceedings of the European Conference on Computer Vision 2002, Copenhagen, Springer, Heidelberg (2002) 661–675
15. Törn, A., Žilinskas, A.: Global Optimization. Volume 350 of Lecture Notes in Computer Science. Springer, Heidelberg (1987)