MEAN SHIFT OBJECT TRACKING USING A 4D KERNEL AND LINEAR PREDICTION

Katharina Quast, Christof Kobylko and André Kaup
Multimedia Communications and Signal Processing, University of Erlangen-Nuremberg
Cauerstr. 7, 91058 Erlangen, Germany

Keywords: Object tracking, Mean shift tracking.

Abstract: A new mean shift tracker which tracks not only the position but also the size and orientation of an object is presented. By using a four-dimensional kernel, the mean shift iterations are performed in a four-dimensional search space consisting of the image coordinates, a scale and an orientation dimension. Thus, the enhanced mean shift tracker tracks the position, size and orientation of an object simultaneously. To increase the tracking performance by using the information about the position, size and orientation of the object in the previous frames, a linear prediction is also integrated into the 4D kernel tracker. The tracking performance is further improved by considering the gradient norm as an additional object feature.

1 INTRODUCTION

Object tracking is still an important and challenging task in computer vision. Among the many different methods developed for object tracking, the mean shift algorithm (Comaniciu and Meer, 2002) is one of the most famous tracking techniques, because of its ease of implementation, computational speed, and robust tracking performance. Besides, mean shift tracking doesn't require any training data as learning based trackers like (Kalal et al., 2010). In spite of its advantages, traditional mean shift suffers from the limitations of the use of a kernel with a fixed bandwidth. Since the scale and the orientation of an object changes over time, the bandwidth and the orientation of the kernel profile should be adapted accordingly.

An intuitive approach for adapting the kernel scale is to run the algorithm with three different kernel bandwidths, former bandwidth and former bandwidth $\pm 10\%$, and to choose the kernel bandwidth which maximizes the appearance similarity ($\pm 10\%$ method) (Comaniciu et al., 2003). A more sophisticated method using difference of Gaussian mean shift kernel in scale space has been proposed in (Collins, 2003). The method provides good tracking results, but is computationally very expensive.

Mean shift based methods which are adapting the scale and the orientation of the kernel are presented in (Bradski, 1998; Qifeng et al., 2007). In (Bradski, 1998) scale and orientation of a kernel are obtained by estimating the second order moments of the object silhouette, but that is of high computational costs. In (Qifeng et al., 2007) adaptation of the kernel scale and orientation is achieved by combining the mean shift method with adaptive filtering, which is based on the recursive least squares algorithm.

In this paper we propose a scale and orientation adaptive mean shift tracker, which doesn't require any other iterative or recursive method nor destroys the realtime capability of the tracking process. This is achieved by tracking the target in a virtual 4D search space considering the position coordinates as well as the target scale and rotation angle as additional dimensions. The tracking method is further enhanced by a linear prediction of the object scene parameters (position, scale and orientation) and by using the image gradient norm as an additional object feature.

The rest of the paper is organized as follows. Section 2 gives an overview of standard mean shift tracking. Mean shift tracking in the 4D search space is explained in Section 3. While the linear prediction is described in Section 4 and the image gradient norm is introduced in Section 5. Experimental results are shown in Section 6. Section 7 concludes the paper.

2 MEAN SHIFT OVERVIEW

Mean shift tracking discriminates between a target model in frame n and a candidate model in frame $n+1$. The target model is estimated from the dis-

Quast K., Kobylko C. and Kaup A..
MEAN SHIFT OBJECT TRACKING USING A 4D KERNEL AND LINEAR PREDICTION.
DOI: 10.5220/0003327305880593
In *Proceedings of the International Conference on Computer Vision Theory and Applications* (VISAPP-2011), pages 588-593
ISBN: 978-989-8425-47-8

crete density of the objects feature histogram $\mathbf{q}(\hat{\mathbf{x}}) = \{q_u(\hat{\mathbf{x}})\}_{u=1\ldots m}$ with $\sum_{u=1}^{m} q_u(\hat{\mathbf{x}}) = 1$.

The probability of a certain feature belonging to the object with the centroid $\hat{\mathbf{x}}$ is expressed as $q_u(\hat{\mathbf{x}})$, which is the probability of the feature $u = 1\ldots m$ occurring in the target model. The candidate model $\mathbf{p}(\hat{\mathbf{x}}_{new})$is defined analogous to the target model, for more details see (Comaniciu and Meer, 2002; Comaniciu et al., 2003). The mean shift algorithm computes the offset from an old object position $\hat{\mathbf{x}}$ to a new position $\hat{\mathbf{x}}_{new} = \hat{\mathbf{x}} + \Delta\mathbf{x}$ by estimating the mean shift vector

$$\Delta\mathbf{x} = \frac{\sum_i K(\mathbf{x}_i - \hat{\mathbf{x}}) w(\mathbf{x}_i)(\mathbf{x}_i - \hat{\mathbf{x}})}{\sum_i K(\mathbf{x}_i - \hat{\mathbf{x}}) w(\mathbf{x}_i)} \quad (1)$$

with kernel $K(\cdot)$ and weighting function $w(\mathbf{x}_i)$ which denotes the weight of $\mathbf{x}_i$ as

$$w(\mathbf{x}_i) = \sum_{u=1}^{m} \delta[b(\mathbf{x}_i) - u] \sqrt{\frac{q_u(\hat{\mathbf{x}})}{p_u(\hat{\mathbf{x}}_{new})}}. \quad (2)$$

The similarity between target and candidate model is measured by the discrete formulation of the Bhattacharyya coefficient

$$\rho[\mathbf{p}(\hat{\mathbf{x}}_{new}), \mathbf{q}(\hat{\mathbf{x}})] = \sum_{u=1}^{m} \sqrt{p_u(\hat{\mathbf{x}}_{new}) q_u(\hat{\mathbf{x}})}. \quad (3)$$

The aim is to minimize the distance between the two color distributions $d(\hat{\mathbf{x}}_{new}) = \sqrt{1 - \rho[\mathbf{p}(\hat{\mathbf{x}}_{new}), \mathbf{q}(\hat{\mathbf{x}})]}$ as a function of $\hat{\mathbf{x}}_{new}$ in the neighborhood of a given position $\hat{\mathbf{x}}_0$. This can be achieved using the mean shift algorithm. By running this algorithm the kernel is recursively moved from $\hat{\mathbf{x}}_\mathbf{0}$ to $\hat{\mathbf{x}}_\mathbf{1}$ according to the mean shift vector.

3 4D KERNEL TRACKING

3.1 4D Kernel Definition

Usually a scaled Epanechnikov kernel is used for mean shift tracking which is defined as

$$K_e(\mathbf{x}) = \frac{1}{h^d} \cdot k_e\left(\frac{\|\mathbf{x}\|^2}{h^2}\right) \quad (4)$$

where h is the kernel bandwith and k_e the profile of the radially symmetric Epanechnikov kernel as defined in equation (12) in (Comaniciu et al., 2003).

Since a radially symmetric kernel is usually a bad approximation of the tracked object shape, we are using an elliptic kernel with varying bandwidths for both semi-axes, which is scaled by a scaling factor s and rotated by a rotation angle ϕ:

$$K'(\mathbf{x}, s, \phi) = \frac{1}{h_a \cdot h_b \cdot s^2} \cdot k'\left(\frac{\|\mathbf{H} \cdot \mathbf{R}(\varphi + \phi) \cdot \mathbf{x}\|^2}{s^2}\right) \quad (5)$$

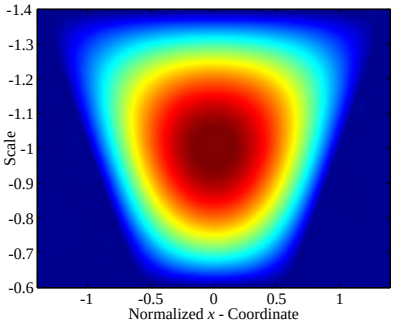

Figure 1: Cut surface of the adaptive kernel with the x-scale-plane when only scale adaptation is used. The colors correspond to the kernel-weights where dark-blue represents 0 and dark-red represents the maximum kernel-weight.

where k' is the kernel profile, h_a and h_b are the bandwidths for the semi-major and semi-minor axis, and φ being the rotation angle between the semi-major axis and the horizontal coordinate axis of the image. The scaling matrix $\mathbf{H}$ and the rotation matrix $\mathbf{R}$ are defined as follows:

$$\mathbf{H} = \begin{pmatrix} \frac{1}{h_a} & 0 \\ 0 & \frac{1}{h_b} \end{pmatrix} \quad (6)$$

$$\mathbf{R}(\varphi) = \begin{pmatrix} \cos(\varphi) & -\sin(\varphi) \\ \sin(\varphi) & \cos(\varphi) \end{pmatrix} \quad (7)$$

The scaled and rotated kernels $K'(\cdot)$ are considered to be the cut surfaces of a 4D tracking kernel with the 2D image plane. As position, scale and rotation are considered to be linearly independent, the scale and orientation adaptive 4D kernel is defined by:

$$K_a(\mathbf{x}, s, \phi) = K'(\mathbf{x}, s, \phi) \cdot K_e\left(\frac{s-1}{h_s}\right) \cdot K_e\left(\frac{\phi}{h_\phi}\right) \quad (8)$$

with 1D Epanechnikov kernels with the bandwidth h_s for the scale dimension and the bandwidth h_ϕ for the rotation dimension. Since the target scale is updated multiplicatively and the target rotation additively, the scale kernel is centered at one (neutral element for multiplication) and the rotation kernel at zero (neutral element for addition). Figure 1 shows the cut surface of the adaptive kernel with the plane spaned by the normalized x-coordinate and the scale dimension if only scale adaptation is used.

3.2 Tracking in the 4D Space

In order to run the mean shift tracking with the 4D kernel $K_a(\cdot)$, the kernel has to be sampled in the scale and rotation dimension and thus a set of $N_s \cdot N_\phi$ scaled and rotated kernels $K'(\cdot)$ is being constructed. An example of the resulting kernel-weights for an uniform

sampling with $N_s = 5$, $h_s = 0.4$, $N_\phi = 7$ and $h_\phi = \frac{\pi}{6}$ is shown in Figure 2. Of course, each of this kernels covers a different area and, therefore, each one has its own pixel set $\{\mathbf{x}_i\}_{i=1..n_h(s_k,\phi_m)}$ for the kernel density estimation (KDE).

Using the whole kernel set centered at $\mathbf{y}$, the candidate histogram is estimated by:

$$\hat{p}[\mathbf{u}](\mathbf{y}) = C_a \cdot \sum_{k=1}^{N_s} \sum_{m=1}^{N_\phi} \sum_{i=1}^{n_h(s_k,\phi_m)} K_a(\mathbf{y}-\mathbf{x_i}, s_k, \phi_m) \cdot \delta[\mathbf{b}(\mathbf{x_i}) - \mathbf{u}] \tag{9}$$

with the normalization constant

$$C_a = \frac{1}{\sum_{k=1}^{N_s} \sum_{m=1}^{N_\phi} \sum_{i=1}^{n_h(s_k,\phi_m)} K_a(\mathbf{y}-\mathbf{x_i}, s_k, \phi_m)} \tag{10}$$

Basically, the 4D KDE equals a series of 2D KDEs with the scaled and rotated kernels:

$$\hat{p}[\mathbf{u}](\mathbf{y}, s_k, \phi_m) = \frac{\sum_{i=1}^{n_h(s_k,\phi_m)} K'(\mathbf{y}-\mathbf{x_i}, s_k, \phi_m) \cdot \delta[\mathbf{b}(\mathbf{x_i}) - \mathbf{u}]}{\sum_{i=1}^{n_h(s_k,\phi_m)} K'(\mathbf{y}-\mathbf{x_i}, s_k, \phi_m)} \tag{11}$$

with a posterior averaging of all separately computed histograms:

$$\hat{p}[\mathbf{u}](\mathbf{y}) = \frac{\sum_{k=1}^{N_s} \sum_{m=1}^{N_\phi} \hat{p}[\mathbf{u}](\mathbf{y}, s_k, \phi_m) \cdot K_e(\frac{s_k - 1}{h_s}) \cdot K_e(\frac{\phi_m}{h_\phi})}{\sum_{k=1}^{N_s} \sum_{m=1}^{N_\phi} K_e(\frac{s_k - 1}{h_s}) \cdot K_e(\frac{\phi_m}{h_\phi})} \tag{12}$$

Since a high pixel-weight $w(\mathbf{x}_i)$ means a high probability of the pixel $\mathbf{x}_i$ belonging to the target, the mean pixel-weight

$$\bar{w}(s_k, \phi_m) := \frac{\sum_{i=1}^{n_h(s_k,\phi_m)} w(\mathbf{x}_i)}{n_h(s_k, \phi_m)} \tag{13}$$

inside the area covered by the kernel $K'(\mathbf{x}, s_k, \phi_m)$ depicts how well the target is approximated by this particular kernel.

The overall mean pixel-weight of the kernel set is then defined by:

$$\bar{w} := \frac{\sum_{k=1}^{N_s} \sum_{m=1}^{N_\phi} \bar{w}(s_k, \phi_m)}{N_s \cdot N_\phi} \tag{14}$$

Thus, the new candidate position is averaged over all kernels of the set, favoring these which approximate the target better.

$$\hat{\mathbf{y}}_\mathbf{1} = \frac{1}{\bar{w}} \cdot \sum_{k=1}^{N_s} \sum_{m=1}^{N_\phi} \bar{w}(s_k, \phi_m) \cdot \hat{\mathbf{y}}_\mathbf{1}(s_k, \phi_m) \tag{15}$$

with $\hat{\mathbf{y}}_\mathbf{1}(s_k, \phi_m)$ being the new candidate position computed with one particular kernel of the set:

$$\hat{\mathbf{y}}_\mathbf{1}(s_k, \phi_m) = \frac{\sum_{i=1}^{n_h(s_k,\phi_m)} \mathbf{x_i} \cdot w(\mathbf{x}_i)}{\sum_{i=1}^{n_h(s_k,\phi_m)} w(\mathbf{x}_i)} \tag{16}$$

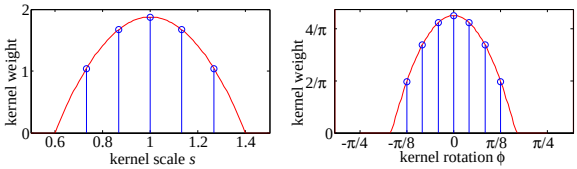

Figure 2: Sampling of scale (*left*) and rotation (*right*) dimension. The continuous kernel is depicted by the red curve.

Once the final target position has been found, the scale and rotation angle update-values are computed by one mean shift iteration in the proper dimensions:

$$\begin{aligned} \hat{s} &= (1/\bar{w}) \cdot \textstyle\sum_{k=1}^{N_s} s_k \cdot \sum_{m=1}^{N_\phi} \bar{w}(s_k, \phi_m) \\ \hat{\phi} &= (1/\bar{w}) \cdot \textstyle\sum_{m=1}^{N_\phi} \phi_m \cdot \sum_{k=1}^{N_s} \bar{w}(s_k, \phi_m) \end{aligned} \tag{17}$$

Usually, this should be done after each candidate position update, but it would require a reconstruction of the entire kernel set after each iteration. Since the linear approximation of the scale and rotation angle update-value has proven to be quite sufficient in the experiments, this compromise has been made in respect to computational efficiency.

Finally, the target scale and rotation angle are updated by:

$$\begin{aligned} h_a' &= h_a \cdot \hat{s} \\ h_b' &= h_b \cdot \hat{s} \\ \varphi' &= \varphi + \hat{\phi} \end{aligned} \tag{18}$$

4 LINEAR PREDICTION OF THE OBJECT SCENE PARAMETERS

Like all iterative solution techniques, the mean shift procedure requires the initial guess (target position in the last image) to be *sufficiently* close to the sought extremum (current target position) for convergence. Under perfect circumstances this means that the tracking kernels have to overlap, but if the tracked object is moving too fast or the scene is captured with a low frame rate that might not be the case. Fortunately, the changes of the object scene parameters are partly predictable. Due to the fact that the overall scene parameters (e.g. real-world position of object and camera) are not known in general, we concentrated on a basic linear prediction rather than on a prediction based on a physical movement model like the one using a Kalman-filter mentioned in (Comaniciu et al., 2003).

The simplest kind of linear prediction would be to assume that the current change of the object scene parameters equals to the last one. Usually, this would be a good guess since the velocity of the object does not change drastically during the sampling interval of the

camera, but this estimation would be highly susceptible to noisy input data, because only one data point is being used for the estimation. The influence of the input noise, however, can be minimized by computing the mean-value of the most recent data points, assuming that the object scene parameters do not change much during the considered interval.

For the computation of the mean-value, one has to distinguish between additively or multiplicatively updated parameters P. In general, if N_P previous parameter updates ΔP_i are being regarded for the mean-value computation, then a consecutive update by all ΔP_i must equal a N_P-fold update by the mean-value $\widehat{\Delta P}$. Let P_t be the parameter at time index t and ΔP_t the parameter update between the time indices t and $t+1$:

Additive updates are defined by

$$\hat{P}_t = P_{t-N_P} + \sum_{i=1}^{N_P} \Delta P_{t-i} = P_{t-N_P} + N_P \cdot \widehat{\Delta P}_t. \quad (19)$$

Thus, the predicted update-value equals the arithmetic mean:

$$\widehat{\Delta P}_t = \frac{1}{N_P} \cdot \sum_{i=1}^{N_P} \Delta P_{t-i} \quad (20)$$

Multiplicative updates, on the other hand, are defined by

$$\hat{P}_t = P_{t-N_P} \cdot \prod_{i=1}^{N_P} \Delta P_{t-i} = P_{t-N_P} \cdot \widehat{\Delta P}_t^{N_P} \quad (21)$$

resulting in the geometric mean being the predicted update-value:

$$\widehat{\Delta P}_t = \sqrt[N_P]{\prod_{i=1}^{N_P} \Delta P_{t-i}} \quad (22)$$

Applying the logarithm on both sides of equation (22) transforms the geometric mean of the update-value into an arithmetic mean of its logarithmic value:

$$\ln \widehat{\Delta P} = \frac{1}{N_P} \cdot \sum_{i=1}^{N_P} \ln \Delta P_{t-i} \quad (23)$$

Therefore, the same prediction method can be used for both types of parameter updates.

Since the position $\mathbf{y}$ as well as the rotation angle φ are updated additively, while the kernel bandwidth $(h_a, h_b)^T$ is updated multiplicatively by the scale factor s, the vector of the changes of the object scene parameters at the time index t is defined by

$$\mathbf{p}_t = \begin{pmatrix} \Delta y_1[t] \\ \Delta y_2[t] \\ \ln s[t] \\ \phi[t] \end{pmatrix} \quad (24)$$

and the parameter changes in the current image are estimated by computing the arithmetic mean of the preceding parameter changes:

$$\hat{\mathbf{p}}_t = \frac{1}{N_P} \cdot \sum_{i=1}^{N_P} \mathbf{p}_{t-i} \quad (25)$$

Before performing the mean shift iterations, the object scene parameters found in the last image are updated using the estimated changes:

$$\hat{\mathbf{y}}_0[t] = \mathbf{y}_0[t] + \begin{pmatrix} \widehat{\Delta y_1}[t] \\ \widehat{\Delta y_2}[t] \end{pmatrix} \quad (26)$$

$$\begin{pmatrix} \hat{h}_a[t] \\ \hat{h}_b[t] \end{pmatrix} = \begin{pmatrix} h_a[t] \\ h_b[t] \end{pmatrix} \cdot \mathrm{e}^{\ln \hat{s}[t]} \quad (27)$$

$$\hat{\varphi}[t] = \varphi[t] + \hat{\phi}[t] \quad (28)$$

5 ADDITIONAL FEATURES

Obviously, the color distribution is a very attractive feature, because it is usually very distinctive and it is offered to the tracker without the need of further image processing. Since the KDE works pixel-based, the tracker would benefit from any feature providing information about the correlation between neighboring pixels. However, using the oriented image gradients, computed by the Sobel filter, directly as additional features would be problematic. This would triple the number of image features, aggravating the *curse of dimensionality* (Scott, 1992) and disturbing the comparison between the target and the candidate histogram. Furthermore, the histogram would become highly rotation-variant and the tracker could be very easily mislead by changes of the object pose.

A possible solution would be to use only the norm of the combined image gradient vector as a new feature. Unlike one might think, this does not result in a huge loss of information as the individual color plane gradients are highly correlated anyway, because they appear mainly at object contours. In respect to computational efficiency, the L1 norm is being used.

$$I^{M+1}(\mathbf{x}) = \| \, [\, (\nabla I^1(\mathbf{x}))^T, .., (\nabla I^M(\mathbf{x}))^T \,]^T \, \|_1 \quad (29)$$

with I^l being the l-th feature plane of the image and

$$\|\mathbf{x}\|_1 := \sum_{i=1}^{\dim(\mathbf{x})} |x_i|. \quad (30)$$

6 EXPERIMENTAL RESULTS

For estimating the target histogram each color channel of the RGB space as well as the image gradient norm is quantized into 8 bins, leading to a total of $8^4 = 4096$ different histogram bins. The used

Figure 3: Results for tracking a police car in sequence *Airport* using the proposed method without gradient information (top) and with gradient information (bottom).

Figure 4: Results for tracking a white car in sequence *Airport* using the standard mean shift (green) and the proposed method with scale and rotation adaptation (blue) and with parameter prediction and scale and rotation adaptation (red).

adaptation parameters were set to $h_s = 0.4$, $h_\phi = 30°$, $N_s = 5$ and $N_\phi = 5$. Thus, the enhanced mean shift tracker was run in the 4D space with a kernel set of $N_s \cdot N_\phi = 25$ kernels.

In the sequence *Airport* (3 fps) vehicles which are moving on an airport apron were tracked using the standard mean shift tracker as well as the proposed enhanced mean shift tracker. In Figure 3 the results of the proposed tracker tracking a police car in sequence *Airport* with and without using the gradient information is shown. It can be seen, that the gradient information is an useful object feature, because the size of the police car is tracked much more reliably using the gradient information. Thus, for all other experiments the gradient information is used for all trackers.

In Figure 4 the results of the standard mean shift method are compared to the proposed method with and without using the linear prediction of the object scene parameters. While the standard mean shift tracker is not able to adapt to the orientation and size of the car (top row) in Figure 4, the new 4D kernel tracker is able to track the size as well as the orientation (middle row) of Figure 4. The results can even be further enhanced by using the linear prediction (bottom row) of Figure 4.

To demonstrate the strength of the adaptive mean shift tracker for tracking fast moving objects, the tracking performance is also evaluated using the sequence *Table Tennis* (30 fps). Since the orientation adaptation is not needed for tracking a circular object like a ball, N_ϕ was set to 1. The results for the standard

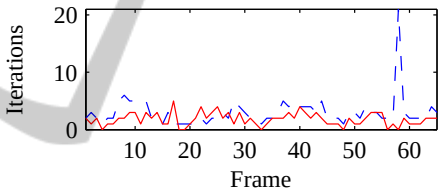

Figure 5: Mean shift iterations needed by the standard mean shift tracking (dashed blue) and by the proposed enhanced mean shift tracking (solid red) for the sequence *Table Tennis*.

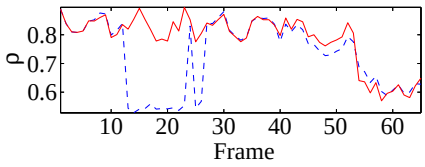

Figure 6: Bhattacharyya Coefficient ρ of the standard mean shift tracking (dashed blue) and of the proposed enhanced mean shift tracking (solid red) for the sequence *Table Tennis*.

mean shift tracker and the proposed tracker can be seen in Figure 7. The number of mean shift iterations needed is shown in Figure 5. Both methods do not require many iterations, but in most cases the proposed enhanced mean shift algorithm needs less iterations than the standard method. Especially around frame 60, when the ball is partly occluded, the proposed tracker needs less iterations than the standard tracker. The Bhattacharyya coefficient of the enhanced mean shift tracker is also more reliable than the one of

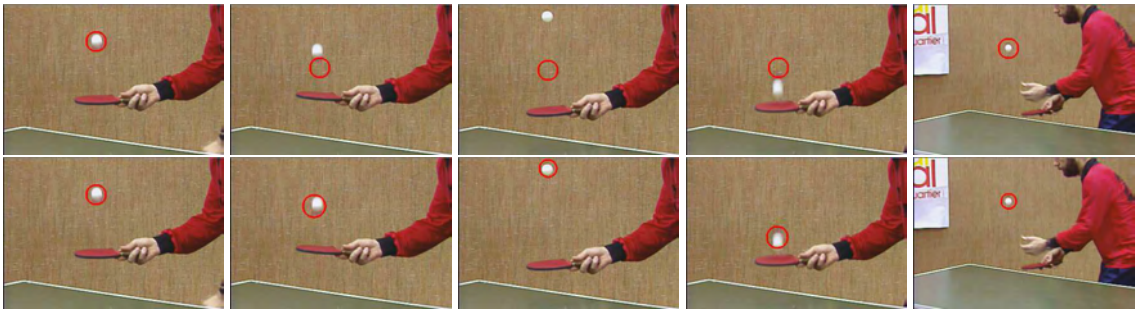

Figure 7: Tracking results for sequence *Table Tennis* of standard mean shift (top) and of the proposed method with scale and rotation adaptation (bottom).

the standard method, see Figure 6. Especially between the frames 13 to 27 the Bhattacharyya coefficient of the standard mean shift tracker decreases and becomes unreliable, because the standard mean shift tracker is not able to follow the fast movement of the ball. While, the proposed tracker has a high and therewith reliable Bhattacharyya coefficient.

Table 1: Computational performance for tracking the police car in sequence *Airport* and the ball in sequence *Table Tennis*.

Tracker	Target	Kernel size	Iterations	fps
standard	police car	97x45	6.63	76.77
proposed	police car	100.7x46.8	3.87	4.22
standard	ball	33x33	3.02	430.48
proposed	ball	31.4x31.4	1.82	97.81

In Table 1 the computational performance, the average kernel size in pixels and the average number of mean shift iterations of both trackers are given. Although the enhanced tracker runs with 25 kernels for sequence *Airport*, 5 for sequence *Table Tennis* respectively, it performs in in real-time. However, it is slower as the standard mean shift tracker.

7 CONCLUSIONS

A new mean shift tracking method using an adaptive 4D kernel to perform the mean shift iterations in an extended 4D search space has been proposed. Thus, the tracker adapts to the changing object scene parameters. Compared to the standard mean shift algorithm, which only tracks the position, the proposed tracker is able to track the position as well as the scale and the orientation of an object. The flexibility of the adaptation can be adjusted by the sampling scheme of the scale and rotation dimensions to match the individual requirements of each tracking scenario. By using the L1 norm as an additional feature the performance is further enhanced. Future work might concentrate on getting the tracker even more robust especially against background clutter.

ACKNOWLEDGEMENTS

This work has been supported by the Gesellschaft für Informatik, Automatisierung und Datenverarbeitung (iAd) and BMWi, ID 20V0801I.

REFERENCES

Bradski, G. (1998). Computer vision face tracking for use in a perceptual user interface. *Intel Technology Journal*, 2:12–21.

Collins, R. T. (2003). Mean-shift blob tracking through scale space. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 234–240.

Comaniciu, D. and Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:603–619.

Comaniciu, D., Ramesh, V., and Meer, P. (2003). Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25:564–575.

Kalal, Z., Matas, J., and Mikolajczyk, K. (2010). PN learning: Bootstrapping binary classifiers by structural constraints. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 49–56. IEEE.

Qifeng, Q., Zhang, D., and Peng, Y. (2007). An adaptive selection of the scale and orientation in kernel based tracking. In *Proc. IEEE Conference on Signal-Image Technologies and Internet-Based Systems*, pages 659–664.

Scott, D. W. (1992). *Multivariate Density Estimation: Theory, Practice, and Visualization*. Wiley-Interscience.