A Comparison of Mean Shift Tracking Methods*

Nicole M. Artner

Digital Media, Upper Austria University of Applied Sciences, Hagenberg, Austria
nicole.artner@gmail.com

Abstract

The mean shift algorithm is a well-known statistical method for finding local maxima in probability distributions. Besides filtering and segmentation it is applied in the field of object tracking. There are several approaches that use the mean shift method for locating target objects in video sequences. This paper compares three similar approaches and investigates their performance on different test videos.

Keywords: object tracking, mean shift, CAMShift, weighted histogram, ratio histogram, candidate model.

1 Introduction

Object tracking is an important task in computer vision. There are many different approaches to track an object in a video sequence. A detailed summary can be found in [15]. One possible approach is to use the mean shift algorithm to localize the target object.

The mean shift algorithm is a robust statistical method which finds local maxima in any probability distribution. It works with a search window that is positioned over a section of the distribution. Within this search window the maximum can be determined by a simple average computation. Then the search window is moved to the position of this maximum and the average computation is repeated again. This procedure is repeated until the mean shift algorithm finds a local maximum and converges.

To apply the mean shift algorithm in the field of object tracking it is necessary to represent the data of video frames as a probability distribution. Every pixel in a frame gets a probability value $P(u,v)$, depending on its color. P is a value which indicates how likely it is that the related pixel belongs to the target object. Using this probability values a frame can be represented as a 2D probability distribution and the mean shift algorithm can be applied. Mean shift is used in color-based object tracking because it is simple and robust. The best results can be achieved if the following conditions are fulfilled:

- The target object is mainly composed of one color.
- The target object does not change its color.
- Illumination does not change dramatically.
- There are no other objects in the scene similar to the target object.
- The color of the background differs from the target object.
- There is no full occlusion of the target object.

*Supported by the Austrian Science Fund under grant P18716-N13.

1.1 Related Work

There are numerous approaches employing the mean shift algorithm in object tracking. Bradski presents in [2] his so-called *CAMShift* (*C*ontinously *A*daptive *M*ean *S*hift) method, Allan et al. improve in [1] the CAMShift method by using a better target model, and Cominiciu et al. [6] propose to use a candidate model in addition to the target model (see Section 2). The mean shift algorithm is often combined with other methods to improve the tracking results. Han et al. [7] use the mean shift algorithm in combination with a double model filter to obtain robust results in scenes with abrupt and fast motion. In [8], Jeong et al. propose to use a Gaussian-cylindroid color model to increase robustness against illumination changes. Wang and Yagi [13] describe an approach using not only color but also shape features with mean shift. Besides object tracking, the mean shift algorithm can also be used for smoothing and segmentation. For more information see [5].

The rest of the paper is organized as follows: Section 2 describes three similar approaches using mean shift for object tracking. Section 3 compares the three approaches from Section 2 using different test sequences. In Section 4 the results and conclusions from Section 3 are summarized.

2 Tracking with mean shift

This section describes three possible variants of the mean shift object tracking method. They were chosen because they differ only by their target model and use no additional features besides the color of the target object. As these three approaches are similar, it is interesting to see how their results differ in the experiments in Section 3.

2.1 CAMShift by Bradski

The *CAMShift* algorithm [2] is an application of the mean shift algorithm for tracking objects in video sequences. The standard mean shift algorithm can only deal with static distributions (i.e., single images) because its search window has a fixed size. Bradski uses a dynamical search window that adapts its size after every video frame, depending on the size of the target object.

As described in Section 1, it is necessary to assign a probability value P to every image pixel $I(u,v)$. Therefore, a target model of the desired object (i.e., out of a user selection) is created in the form of a 1D histogram. The histogram is quantized into 16 bins which groups similar values and thereby improves performance. Bradski takes the H-channel of the HSV color space (*hue*, *saturation*, *brightness*) to describe the target object by a range of color hues. Depending on the occurrence of a hue in the histogram, the probability value lies in $[0,1]$. To increase performance, the probability distribution for the mean shift algorithm is created within a so-called *calculation region* which is, in most cases, smaller than the image. The probability values are assigned to the pixels depending on their hue and thereby the histogram is used like a lookup-table.

After determining the probability distribution $P(u,v)$, the maximum of the distribution is searched. The location of the maximum represents the position of the target object in the actual frame. To calculate the maximum within the search window, statistical moments of zeroth and first order are used.

A statistical moment of order p and q can be generally formulated as [3, p. 226]

$$m_{pq} = \sum_{(u,v)\in\Omega} P(u,v)\cdot u^p \cdot v^q,$$

where Ω is the distribution. The zero-order moment

$$m_{00} = \sum_{(u,v)\in\Omega} P(u,v),$$

corresponds to the integral over the distribution Ω. Similarly, the moments of first order are

$$m_{10} = \sum_{(u,v)\in\Omega} P(u,v)\cdot u \quad \text{and} \quad m_{01} = \sum_{(u,v)\in\Omega} P(u,v)\cdot v.$$

The position of the target object $\mathbf{c} = (c_x, c_y)$ is then calculated as [2]

$$c_x = \frac{m_{10}}{m_{00}}, \quad c_y = \frac{m_{01}}{m_{00}}.$$

After the position of the target object has been determined, the size of the search window is adapted for the next frame. This adaptation is done with the moment of zero order and the maximum value in the distribution P_{max} [2],

$$w_s = 2\cdot\sqrt{\frac{m_{00}}{P_{max}}} \quad \text{and} \quad h_s = 1.2\cdot w_s,$$

where w_s is the width and h_s is the height of the search window $\mathbf{s}$. Bradski multiplies the height with a factor of 1.2 since human faces being tracked are more or less elliptical. For the experiments in this paper we used different objects, not only faces, so a quadratic search window is used.

Besides the position of the target object, its width, height and orientation can also be derived from the statistical moments. The *width*

$$w_{target} = 2\cdot\left(\frac{(a+c)-\sqrt{b^2+(a-c)^2}}{2}\right)^{\frac{1}{2}}$$

and the *height*

$$h_{target} = 2\cdot\left(\frac{(a+c)+\sqrt{b^2+(a-c)^2}}{2}\right)^{\frac{1}{2}}$$

are calculated from the statistical moments m of first, second and zeroth order, with

$$\begin{aligned} a &= \frac{m_{20}}{m_{00}} - c_x^2, \\ b &= 2\cdot\left(\frac{m_{11}}{m_{00}} - c_x\cdot c_y\right), \\ c &= \frac{m_{02}}{m_{00}} - c_y^2. \end{aligned}$$

Using central moments μ, the orientation of the target object can be determined. Central moments are independent of the position of the distribution Ω. This is achieved by shifting the position of the distribution to the origin of the coordinate system [3, p. 227], i.e.,

$$\mu_{pq} = \sum_{(u,v)\in\Omega} P(u,v)\cdot(u-c_x)^p\cdot(v-c_y)^q.$$

Finally, the *orientation* is calculated from the central moments of first and second order as

$$\varphi_{target} = \frac{1}{2}\tan^{-1}\left(\frac{2\cdot\mu_{11}}{\mu_{20}-\mu_{02}}\right).$$

2.2 Method by Allan

In comparison to the original CAMShift method [2], Allan et al. [1] employ a 3D histogram for the target model, meaning that they use all three channels of the RGB color space. The aim is to improve the distinction between background, other objects and the target object. All implementations for the experiments in this paper use the HSV color space to do a fair comparison of the approaches.

2.2.1 Weighted histogram

Besides the three channels of the HSV color space, the target model is created from a selection (containing the target object) of weighted pixels. The weight of each pixel depends on its distance to the center of the selection. By weighting the pixels, the influence of background pixels on the target model should be reduced.

For the weighting, the profile

$$k(r) = \begin{cases} 1-r & \text{for} \quad r \le 1, \\ 0 & \text{for} \quad r > 1, \end{cases} \tag{1}$$

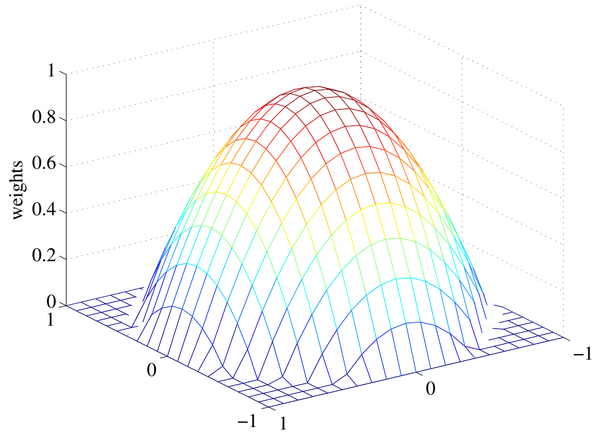


Figure 1: Weighting with the Epanechnikov kernel $K(r)$ for normalized pixel positions within $[-1,1] \times [-1,1]$.

of the *Epanechnikov* kernel is used, which yields the radially symmetric kernel

$$K(r) = \begin{cases} \frac{1}{2} \cdot c_d^{-1} \cdot (d+2) \cdot (1 - \|r\|^2) & \text{for} \quad \|r\| \le 1, \\ 0 & \text{for} \quad \|r\| > 1, \end{cases}$$

where c_d is the volume of the unit d-dimensional sphere [5].

In order to get a weighting that is independent of the size of the selected region, all pixel positions are normalized, such that their coordinates (x,y) are within $[-1,1] \times [-1,1]$ (see Figure 1). Thus the center $\mathbf{c}$ of the selection is moved to the origin of the coordinate system and x-positions are scaled with $(w_s - 1)/2$ and y-positions with $(h_s - 1)/2$, w_s and h_s being width and height of the selection.

In Equation (1), r stands for the squared distance of a pixel to $\mathbf{c}$ and is calculated as

$$r = \|\mathbf{x}_i^*\|^2 = \left(2 \cdot \frac{x - c_x}{w_s - 1}\right)^2 + \left(2 \cdot \frac{y - c_y}{h_s - 1}\right)^2,$$

where $\mathbf{x}_i^*$ is the normalized pixel position.

As in [2], the histogram is *binned*, but in this case with $32 \times 32 \times 16$ bins. The combined probability value for each bin (used later-on to create the probability distribution) is determined as [1]

$$\hat{q}_u = \sum_{i=1}^{n} k\left(\|\mathbf{x}_i^*\|^2\right) \cdot \delta\left(b(\mathbf{x}_i) - u\right). \tag{2}$$

In Equation (2), $\mathbf{x}_i$ is the pixel position in original coordinates (x,y). The function $b: R^2 \to \{1 \ldots m\}$ projects a pixel in 2D coordinates $\mathbf{x}_i$ into the 1D space of the histogram bin indices. $b(\mathbf{x}_i)$ returns, dependent on the HSV color value, the index of the fitting histogram bin for position $\mathbf{x}_i$. $\delta()$ denotes the Kronecker delta function, defined as

$$\delta(x) = \begin{cases} 1 & \text{for} \quad x = 0, \\ 0 & \text{for} \quad x \neq 0. \end{cases}$$

2.2.2 Ratio histogram

Allan et al. were not satisfied with the improvement of tracking by the weighted histogram so they also introduce a *ratio histogram* in [1]. Besides the weighting of the pixels depending on their distance to the center $\mathbf{c}$, they additionally propose a weighting with a so-called *ratio histogram* $\hat{o}$ (a model of the background). For the calculation of $\hat{o}$, a region three times larger than the search window is employed. $\hat{o}$ is determined as $\hat{q}$ with Equation (2), but with the kernel profile

$$k'(r) = \begin{cases} 0 & \text{for} \quad r \le 1 \\ r & \text{for} \quad r > 1. \end{cases}$$

With this profile, the pixels from which the target model was created are excluded (weighted with 0) and all other pixels are weighted higher the farther they are away from the center. Like $\hat{q}$, the ratio histogram consists of $1 \ldots m$ bins per color channel. For every bin, a *weighting factor*

$$w_u = \begin{cases} \frac{\hat{o}_{\min}}{\hat{o}_u} & \text{for} \quad \hat{o}_u > 0, \\ 1 & \text{for} \quad \hat{o}_u = 0, \end{cases}$$

is calculated, where $\hat{o}_{\min}$ is the smallest nonzero value in $\hat{o}$. The probability values of the bins of $\hat{q}$ are weighted with the corresponding w_u as

$$\hat{q}_{u_w} = \hat{q}_u \cdot w_u,$$

and the result is a histogram $\hat{q}_w$ that is additionally weighted with background information.

The properties of the target object (width, height and orientation) and the search window size are calculated as described in Section 2.1.

2.3 Method by Comaniciu

Comaniciu et al. [6] propose an approach using a target and a *candidate* model. They create the candidate model with the pixels of the target object of the actual frame to recognize changes in the target object. The formulation of the target model,

$$\hat{q}_u = C \cdot \sum_{i=1}^{n} k\left(\|\mathbf{x}_i^*\|^2\right) \cdot \delta\left(b(\mathbf{x}_i) - u\right),$$

is the same as the weighted histogram used by Allan et al., except for the constant C. C is required to fulfill the condition

$$\sum_{u=1}^{m} \hat{q}_u = 1$$

and is defined as

$$C = \frac{1}{\sum_{i=1}^{n} k\left(\|\mathbf{x}_i^*\|^2\right)}.$$

The candidate model is calculated at the current position $\mathbf{c}$ of the target object. In our implementation, the candidate model is calculated out of a so-called *candidate window*, which is smaller (70%) than the search window to keep the tracking process from including too many background pixels. To determine the candidate model

$$\hat{p}_u(\mathbf{c}) = C \cdot \sum_{i=1}^{n} k\big(\|\mathbf{x}_i^*\|^2\big) \cdot \delta\big(b(\mathbf{x}_i) - u\big), \qquad (3)$$

the pixels within the search window are used [6]. Equation (3) also contains the constant C to allow a comparison between the target and the candidate model.

To compare the two models, some kind of similarity measure is necessary. For example, Comaniciu et al. use the *Bhattacharyya* coefficient

$$B = \sum_{u=1}^{m} \sqrt{\hat{p}_u(\mathbf{c}) \cdot \hat{q}_u}$$

to calculate the difference between the two distributions $\hat{q}$ and $\hat{p}(\mathbf{c})$. Geometrically, one can interpret the Bhattacharyya coefficient as the cosine of the angle between the m-dimensional vectors $(\sqrt{\hat{q}_1}, \ldots, \sqrt{\hat{q}_m})^T$ and $(\sqrt{\hat{p}_1(\mathbf{c})}, \ldots, \sqrt{\hat{p}_m(\mathbf{c})})^T$. If $B = 1$, the angle between the two vectors is 0, and the two models are equal.

In [1] and [2], each pixel (within the calculation region) receives a probability value from the fitting histogram bin of the target model. In contrast, Comaniciu et al. assign a weight

$$w_i = \sum_{u=1}^{m} \sqrt{\frac{\hat{q}_u}{\hat{p}_u(\mathbf{c})}} \cdot \delta\big(b(\mathbf{x}_i) - u\big)$$

to every pixel in the search window, depending on both models. The probability values are only assigned within the search window, because in each iteration step of the mean shift algorithm, a new candidate model is created out of the search window at the actual position. So, it makes no sense to generate a bigger distribution, as it is used only for one iteration step. The target position is subsequently found as [6]

$$\mathbf{c} = \frac{\sum_{i=1}^{n} \mathbf{x}_i \cdot w_i}{\sum_{i=1}^{n} w_i}.$$

As the Bhattacharyya coefficient measures the similarity between the models, it could be used to adapt the target model. This would help to cope with illumination and appearance changes. For the experiments, a very simple method is used to adapt the target model, called "'parameterized neglecting"'. From the original target model $\hat{q}$ – obtained from the first frame –, the target model from the previous frame $\hat{q}_{t-1}$ and the candidate model $\hat{p}(\mathbf{c})$, we calculate the current target model

$$\hat{q}_t = \hat{q} \cdot \alpha_1 + \hat{q}_{t-1} \cdot \alpha_2 + \hat{p}(\mathbf{c}) \cdot \alpha_3,$$

where $0 \leq \alpha_1 \leq 1$, $0 \leq B \leq 1$, $\alpha_2 = ((1-\alpha_1) \cdot B)$, $\alpha_3 = (1-\alpha_1) - \alpha_2$ and $\alpha_1 + \alpha_2 + \alpha_3 = 1$. The constant α_1 specifies the influence of the original target model on the current target model. The properties of the target object are calculated as in [2]. The probability distribution for the properties is created with the probability values of the current target model $\hat{q}_t$.

	Method	Ref.	Seq. A	Seq. B	Seq. C
❶	CAMShift	[2]	24	19	29
❷	Weighted histogram	[1]	28	29	28
❸	Ratio histogram	[1]	29	28	29
❹	Candidate + target model	[6]	26	26	28

Table 1: Performance of the four tracking methods on test video sequences A, B, C (numbers are in frames/sec).

3 Experiments

In this section, the results of the experiments with four different methods are presented: CAMShift by Bradski [2], weighted histogram by Allan [1], ratio histogram by Allan [1], and Comaniciu's approach with candidate plus target model [6]. Within this section, the methods are referenced as ❶…❹. The target object was selected by the user in the first frame, and the initial target model was created out of this selection. In all experiments, the selection of the frames for the figures was dependent on their relevance. Therefore, there are different frames for different implementations in the figures. For every test sequence, the results of the different methods are compared to ground truth (position, width and height of object) using the Euclidean distance to allow a meaningful comparison. Ground truth was determined manually for every 10th frame.

The performance (in frames per second) of every method strongly depends on the number of iterations and the search window size. Table 1 lists the performance of the four methods on three video test sequences A, B, C. The tests were performed on a Pentium 4, 2.8 GHz, with 512 MB RAM. All algorithms were implemented in Java.

3.1 Test Sequence A (face)

In the first test sequence (Figure 2) a face should be tracked. The difficulty is that the background with the brown furniture has hues very similar to the face itself.

Figure 2(a) shows that the 1D histogram of Bradski's CAMShift method is not sufficient as target model. The background pixels are identified as target object, and that is why the search window is growing, the calculated positions are inaccurate, and the other properties (width, height and orientation) are useless (see curve (1) in Figure 6(a)).

With the weighted histogram as a target model, the result is already much better than with the CAMShift method (see Figure 2(b)). The properties of the target object are more accurate and usable. The corresponding curve (2) in Figure 6(a) shows that the maximum of the Euclidean distance is between the measurements 50–70. The reason for that is the hand movement, which leads to large jitter.

In Figure 2(c), the results of the ratio histogram method are shown, and they are better than the results with the weighted histogram. The disturbance through the hand passing the face is much smaller than with the weighted histogram (see curve (3) in Figure 6(a)).

The approach proposed by Comaniciu et al. shows promising results too (Figure 2(d)). The Euclidean distance is quite high during the occlusions through the hand (curve (4) in Figure 6(a)). A reason for that is the adaptation of the target model.

3.2 Test Sequence B (orange)

This test sequence (Figures 3, 4) shows a moving *orange*, where the challenge is the fast and abrupt motion of the orange, and its similarity to the background. The values calculated for the orientation in this test are not reliable, because the object is more or less circular, and thus small changes in the probability distribution can cause large changes in orientation.

Figure 3(a) shows that the CAMShift method has the same problems as with the first test sequence. The calculated properties are not feasible, and the Euclidean distance between the results and ground truth is very large (see curve (1) in Figure 6(b)).

The implementation with the weighted histogram handles this test very well and tracks the object successfully (see Figure 3(b)). The corresponding cirve (2) in Figure 6(b) shows that the Euclidean distances are within an acceptable range and do not vary strongly.

In Figure 4(a), the result with the ratio histogram can be seen. Surprisingly, the result is worse than in Figure 3(b), because the target object is lost between frames 40 and 45. The reason is that important colors were defined as background (in the background histogram) and are not in the target model.

The best result with this sequence was achieved with the approach of Comaniciu et al. As Figure 4(b) and the corrersponding plot (4) in Figure 6(b) show, the position, width, and height of the target object were determined quite accurately.

3.3 Test Sequence C (cup)

In the third test sequence (Figure 5), a violet cup has to be tracked. The challenges are the illumination changes and the changes of the object's appearance due to the rotation of the cup.

Surprisingly, the tracking with the CAMShift method delivered very good results. Figure 5(a) and curve (1) in Figure 6(c) show that the determined properties of the target object are accurate and reliable. Since the hues of the cup differ enough from the hues of the background, the 1D histogram is perfectly sufficient as target model. The 1D histogram only uses the hue information and so the brightness changes pose no major problem, because brightness is not considered in the model.

Figures 5(b, c) and the corresponding curves (2) and (3) in Figure 6(c) show the results for the approaches of Allan et al. (weighted histogram and ratio histogram). Both approaches delivered similar results. At the beginning, the tracking performs well, but then the target object is lost in frame 298 because of the illumination and appearance changes.

The results for Comaniciu's approach can be seen in Figure 5(d). As with the CAMShift method, the target object can be tracked through the whole sequence. This solution works well, even thought it takes into account the brightness channel of the HSV color space. The reason is the simple target model adaptation. The corresponding curve (4) in Figure 6(c) shows that the Euclidean distances around measurement 30 are the highest. The reason is the very difficult part of test sequence C, where the cup is tilted. In this part of the sequence, the methods of Allan et al. failed.

4 Discussion and Conclusions

The CAMShift tracking method is the simplest of the four methods and supplies reliable and robust results, if the colors in the background differ significantly from those in the target object. It can deal quite well with slight illumination and appearance changes.

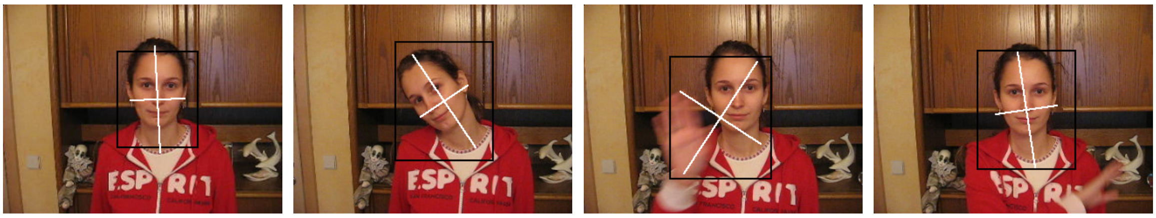
Extending the target model to a 3D histogram, as proposed in [1] and [6], leads to a better distinction of the target object from the background and other objects, even when they are similar. The main drawback is that tracking is less robust against illumination and appearance changes.

By using a weighted histogram, as proposed by Allan et al. [1], the target model becomes more reliable, since it contains fewer background pixels. The ratio histogram seems to be even more robust, but it also has its drawbacks. Imagine that the aim is to track a single face in a video showing several people. When the ratio histogram is created, it is possible that there is the color information of another face in the background histogram. This will lead to a failure in tracking, because important colors will be weighted very low or, in the worst case, set to 0.

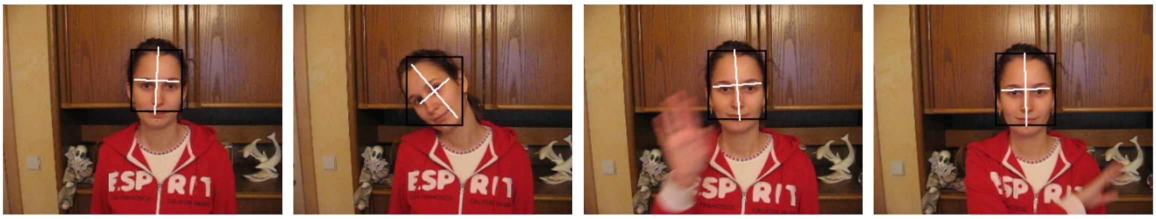
The approach proposed by Comaniciu et al. [6] appears to be the most promising, but also the approach with the highest computational costs (even though the difference is not very high, see Table 1). It might be a good idea to improve the adaptation of the target model, as the solution with parameterized neglecting is not the best, as in [12]. Another idea is to use an estimator like the Kalman filter [14, 11] to improve tracking with mean shift. It is possible to use the estimated position as the starting position for the mean shift algorithm (not the position from the last frame as before), and use the result of mean shift as a measurement for the correction process of the estimator. Another good effect of the Kalman filter is that it will smooth the results of tracking with mean shift. Of course, there are other appealing estimators besides the Kalman filter, such as the particle filter [4, 9, 10], for example.

All four methods have advantages and disadvantages under different circumstances. The CAMShift method, with its simplicity, can be a good choice if the scene is not demanding. If the illumination is constant but the background is similar to the object, the methods by Allan et al. should be applied. For more challenging scenes, the approach by Comaniciu et al. appears to be the best choice.

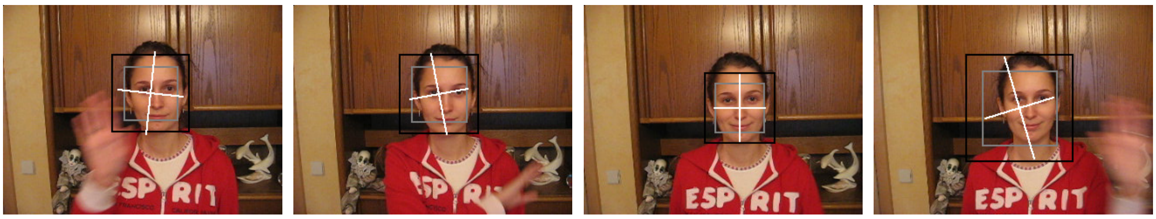
(a) Method ❶ (CAMShift)



(b) Method ❷ (weighted histogram)

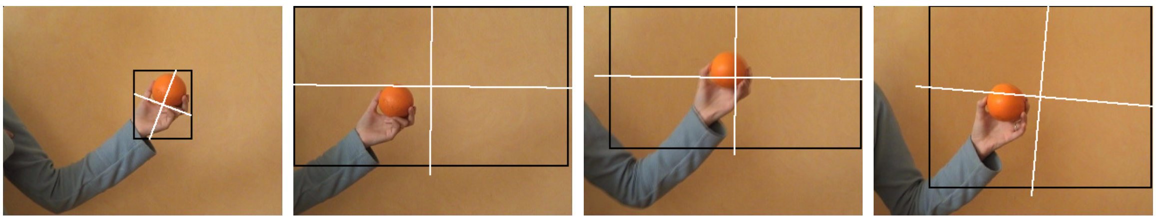


(c) Method ❸ (ratio histogram)

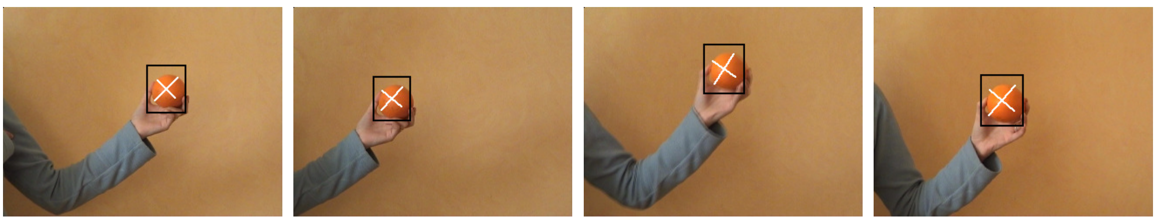


(d) Method ❹ (target plus candidate model)

Figure 2: Tracking results for Test Sequence A obtained with methods ❶. . . ❹.

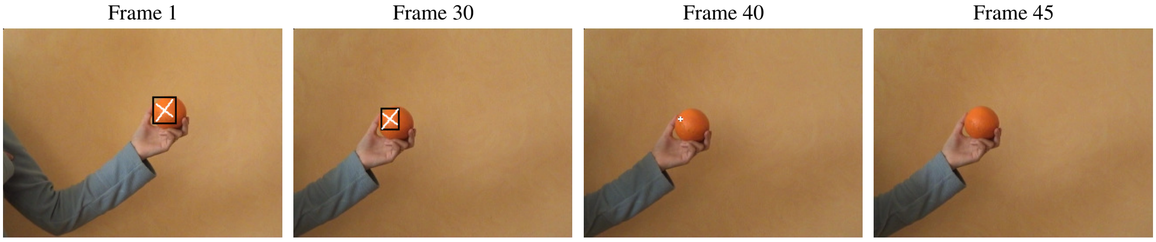


(a) Method ❶ (CAMShift)

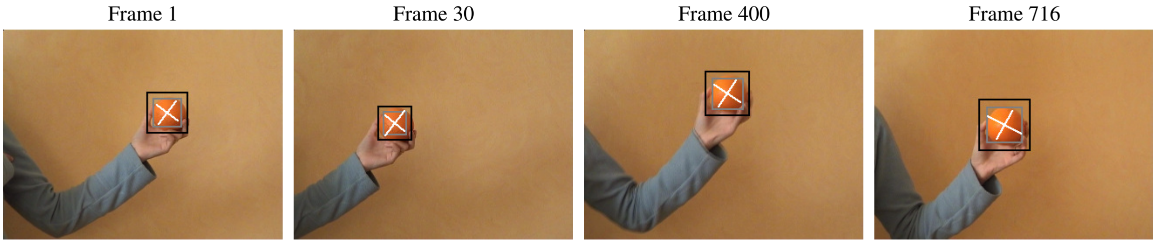


(b Method ❷ (weighted histogram)

Figure 3: Results for Test Sequence B obtained with methods ❶, ❷.

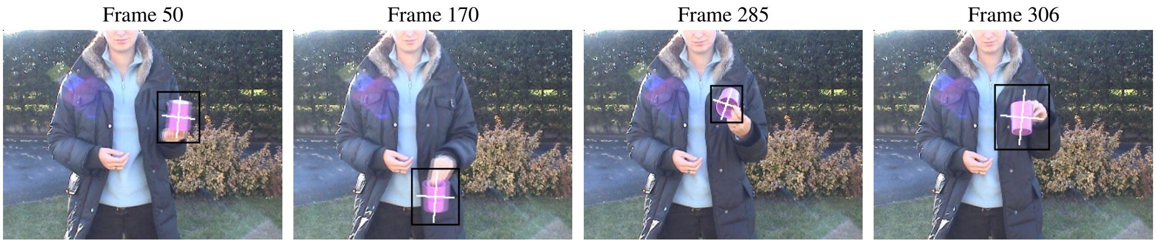


(c) Method ❸ (ratio histogram)

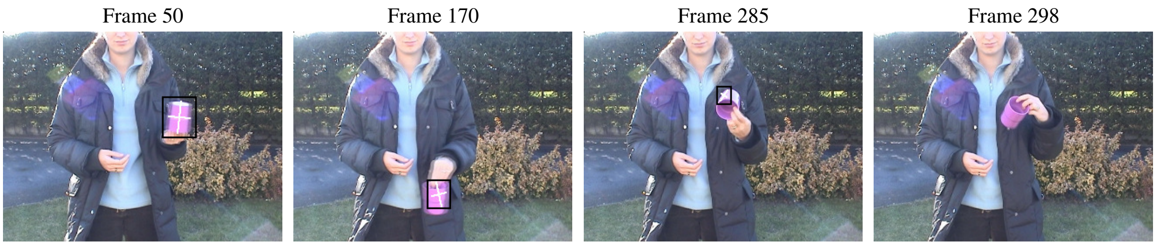


(d) Method ❹ (target plus candidate model)

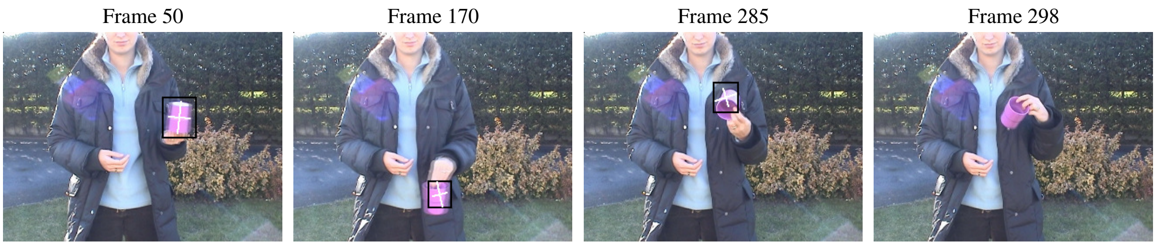
Figure 4: Tracking results for Test Sequence B obtained with methods ❸, ❹.



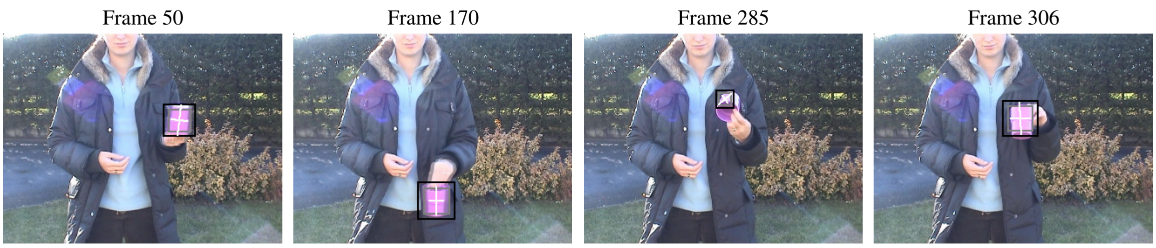
(a) Method ❶ (CAMShift)



(b Method ❷ (weighted histogram)

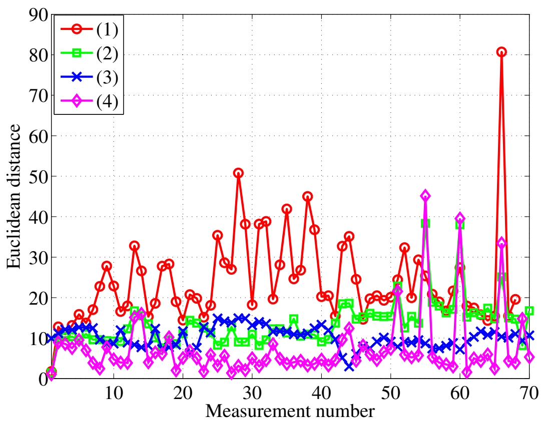


(c) Method ❸ (ratio histogram)

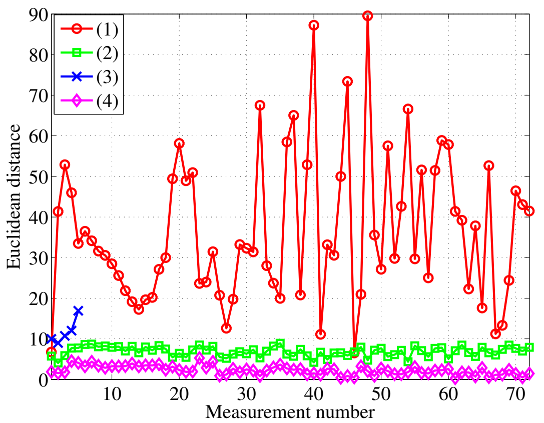


(d) Method ❹ (target plus candidate model)

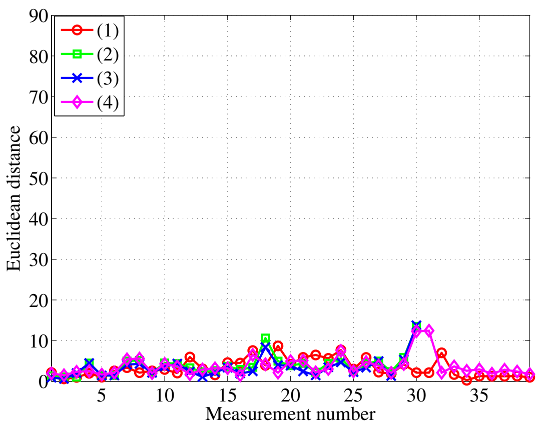
Figure 5: Tracking results for Test Sequence C obtained with methods ❶. . . ❹.



(a) Test Sequence A



(b) Test Sequence B



(c) Test Sequence C

Figure 6: Distance between detected target and ground truth positions for test sequences A, B, and C. Curves marked (1)...(4) plot the results for the four tracking methods ❶...❹ described in Section 2.

References

[1] J. G. Allen, R. Y. D. Xu, and J. S. Jin. Object tracking using camshift algorithm and multiple quantized feature spaces. In *Proc. Pan-Sydney area workshop on visual information processing*, pages 3–7, 2004.

[2] G. R. Bradski. Computer vision face tracking for use in a perceptual user interface. *Intel Technology Journal*, 2, April–June 1998.

[3] W. Burger and M. J. Burge. *Digital Image Processing – An Algorithmic Introduction using Java*. Springer-Verlag, 2007.

[4] C. Chang and R. Ansari. Kernel particle filter for visual tracking. *IEEE Signal Processing Letters*, 12(3):242–245, March 2005.

[5] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *PAMI*, 24(5):603–619, 2002.

[6] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *PAMI*, 25(5):564–575, 2003.

[7] R. Han, Z. Jing, and G. Xiao. Probabilistic motion switch tracking method based on mean shift and double model filters. In *Proc. International Symposium on Neural Networks*, volume 2, pages 705–714, Nanjing, China, June 2007.

[8] M.-H. Jeong, B.-J. You, and W.-H. Lee. Color region tracking against brightness changes. In *Proc. Australian Joint Conference on Artificial Intelligence*, volume 4304/2006, pages 536–545, Hobart, Australia, December 2006.

[9] K. Nummiaro, E. Koller-Meier, and L. J. Van Gool. Object tracking with an adaptive color-based particle filter. In *Proc. of the 24th DAGM Symposium on Pattern Recognition*, pages 353–360, London, UK, 2002. Springer-Verlag.

[10] P. Pérez, C. Hue, J. Vermaak, and M. Gangnet. Color-based probabilistic tracking. In *Proc. ECCV*, volume 1, pages 661–675, London, UK, May 2002. Springer-Verlag.

[11] H. W. Sorenson. Least-squares estimation: from Gauss to Kalman. *IEEE Spectrum*, 7:63–68, July 1970.

[12] H. Uemura, J. K. Tan, and S. Ishikawa. A robust color tracker employing a two-dimensional color histogram with an anisotropic asymmetric gaussian distribution model. In *Proc. IEEE Region 10 Conference*, pages 1–4, Hong Kong, China, November 2006. IEEE Computer Society.

[13] J. Wang and Y. Yagi. Integrating shape and color features for adaptive real-time object tracking. In *Proc. International Conference on Robotics and Biomimetics*, pages 1–6, Kunming, China, December 2006.

[14] G. Welch and G. Bishop. An introduction to the Kalman filter. Technical report, University of North Carolina, Chapel Hill, NC, USA, 1995.

[15] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Comput. Surv.*, 38(4):13, 2006.