IJICTR International Journal of Information & Communication Technology Research
ITRC

Volume 4- Number 3- June 2012 (99-106)

Research Note

A Visual Tracking Algorithm Based on CAMShift and Motion Segmentation

Ebrahim Emami
Computer Engineering Department
Iran University of Science & Technology
Tehran, Iran
ebrahim_emami@comp.iust.ac.ir

Mahmood Fathy
Computer Engineering Department
Iran University of Science & Technology
Tehran, Iran
mahfathy@iust.ac.ir

Accepted: December 31, 2012- Accepted: May 1, 2012

Abstract—**Continuously adaptive mean-shift (CAMShift) is an efficient and light-weight tracking algorithm firstly developed based on mean-shift to track human face in a perceptual user interface. CAMShift tracks a target by searching for the most similar areas to the target region in frames of a video sequence in regard to its reference color histogram. While color based CAMShift is suitable for tracking targets in simple cases, it fails to track objects in more complex situations. In this paper we review our low cost extension to improve the traditional CAMShift algorithm. Combining the original algorithm with simple motion segmentation techniques, we proposed an improved CAMShift algorithm to cope with CAMShift`s tracking problems. We have evaluated the efficiency of our approach through analyzing our tracking results. We have compared our results with other tracking algorithms in various tracking scenarios.**

Keywords-Target tracking, CAMShift, motion segmentation, mean-shift, probability distribution image

I. Introduction

Object tracking is a key task in the field of computer vision. An efficient tracking algorithm will lead to the better performance of higher level vision tasks such as behavior analysis and activity recognition. Object tracking has a lot of potential applications in various fields of science and industry like automatic surveillance, human computer interaction, robotics, video compression and virtual reality [1-3]. Various approaches of object tracking have been proposed in the literature. Real time performance and low computation requirements are two key features of a practical tracking algorithm in real world applications [1].

Mean-shift is a kernel-based tracking method which uses density-based appearance models to represent targets. The method tracks a target by finding the most similar distribution pattern in a sequence of frames with its target reference pattern by iterative searching. Mean-shift has been widely used in computer vision applications because of its relative simplicity and low computational cost. But mean-shift would fail in changing the track window`s scale, as targets move toward or away from the camera [1,4].

Based on mean-shift, continuous adaptive mean-shift (CAMShift) was proposed to overcome the problem [5]. CAMShift adaptively adjusts the track window`s size and the distribution pattern of targets during tracking. CAMShift can be used to track the distribution of any kind of feature that represents the target in a lightweight, robust and efficient way. Most researchers though, use color data to represent targets for CAMShift [3], which give the method a low complexity and practical performance.

While CAMShift performs well with objects that have a simple and constant appearance, it is not robust in more complex scenes. For example, when background has similar color distribution with a target, or when a target moves in front of background objects with different colors, the tracker is very likely to fail. In another case, when the initial search window contains some parts of the background, due to poor object detection, it would normally cause the CAMShift`s search window to drift and diverge. This might be an intense problem because the traditional CAMShift algorithm usually starts with manually selecting a target region by a user [6], and a human user may not be able to fully segment the target region form the background. The problem of search window drift is inherent to many probability-based trackers, since these techniques only track the peak of a probability distribution – not taking into account the composition of probabilities [7].

Motion segmentation is a fundamental part of many intelligent video surveillance algorithms. Motion segmentation aims at separating out moving objects of a video frame from its static background. The only interesting parts in the frames of a video sequence to us, are usually the moving objects. Detecting the moving regions provides us a focus of attention for later processes on these regions of interest. Various motion segmentation methods have been proposed in the literature. Background subtraction, temporal differencing and optical flow are amongst the most common methods used in different video surveillance applications [8].

In this paper we present an improved CAMShift algorithm to overcome the CAMShift drawbacks mentioned above. Our extension combines motion segmentation with the traditional CAMShift algorithm to improve the tracking performance. Employing motion segmentation reduces the undesirable effects of background feature information on the tracker, and so helps us to solve the CAMShift`s problems related to background and target color interference.

The remainder of this paper is organized as follows. Related works are discussed in section 2. Section 3 describes our improved CAMShift algorithm, and in section 4, we discuss our approach`s experimental performance and compare our algorithm`s results with other tracking algorithms.

II. RELATED WORK

Various approaches of object tracking are proposed in the literature which can be put into three main categories of point tracking, kernel tracking and silhouette tracking. Kernel tracking refers to the tracking methods which track targets` shape or appearance features. For instance, a rectangle or elliptical shape with an associated feature histogram can be regarded as a kernel. A kernel tracking method then computes the motion of targets represented with the defined kernel. The computed motion is usually in the form of parametric motion types, such as transformation, conformal or affine. This is opposed to the point tracking methods like Kalman filter, which regard the detected objects as points and try to locate the estimated point in consecutive frames. The point tracking methods have a close relation with additional moving object detection methods [1, 4].

The simplest type of kernel tracking is template matching, in which the frames of a video sequence are searched with a brute force strategy to find a region similar to the target template. Visual features such as image intensity, color and image gradients might also be used as features in the process of target localization in a template matching method. The main drawback of template matching is its computational cost due the brute force search of the target`s template. Different template matching methods are discussed in [1].

Mean-shift tracker is probably the most well known kernel tracking method used in the literature. Instead of searching for the most similar area to the reference target as what is done in template matching, mean-shift tracker maximizes the appearance similarity between the target`s feature representation (usually color histogram) and the candidate target location by iteratively comparing and shifting the candidate target location. The main advantage of mean-shift over the standard template matching is its lower computation cost due to the elimination of the brute force search [1].

CAMShift was first introduced as a technique for face and head tracking in a perceptual user interface as a mean-shift extension in [5]. CAMShift`s introduction was the result of an effort to find a fast and simple algorithm for basic tracking [5].

Several variations of mean-shift tracking have been proposed in the literature. CAMShift has also been object to a variety of modifications and improvements to accommodate more general tracking applications since its introduction [5]. In the remaining part of this section, we review some of the various tracking algorithms proposed based on both tracking methods in the literature.

Reducing the target and background color interference has been a popular approach among the researchers to improve the performance of mean-shift and CAMShift object tracking. This purpose can be achieved by calculating the target color histogram with a weighted scheme. A weighted histogram calculation function gives higher weights to pixels closer to the object center, since the further pixels are more likely to be from background. Reference [9] for instance calculates the weighted color histogram of a circular target region in order to locate the target with mean-shift tracking. Similar weighted histogram calculation methods are also used in CAMShift tracking in [6] and [10].

Using multiple reference color histograms has also been studied in both mean-shift and CAMShift tracking algorithms. Reference [11] enhances the mean-shift`s tracking performance when multiple views of a target are available prior to the tracking. The approach proposed in [7] also uses multiple color histograms to model different appearances of a target and an accumulated histogram to compute the probability distribution of the corresponding target in every frame for CAMShift tracking. The authors evaluated their approach`s robustness in tracking objects which have diverse appearances like a multi-color cube. These methods can be helpful when multiple views of a single target are available.

Using Kalman filter in combined with mean-shift or CAMShift is an approach among the researchers to improve the tracking performance [2, 12-14]. Kalman filter basically builds a model of a system with a history of the system`s measurement [15]. This model is used to estimate the state of a linear system. Kalman filter has been widely used in the field of machine vision for the purpose of object tracking.

A combination of mean-shift and Kalman filter tracking has been proposed in [16], the method uses Kalman filter to predict the possible target position in a frame, and then uses mean-shift to search for the target in the predicted area. A similar method is proposed in [14] for CAMShift tracking in which Kalman filter is used to build a model of the target`s motion while tracking it with CAMShift. The model is used to predict the target`s position and the predicted area is searched by CAMShift to locate the target. If the search is successful CAMShift`s tracking result is used as the observed value of the target`s position to correct the Kalman filter, and the predicted value is considered as the target`s position. This model can track the target in the presence of occlusion or rotation. Reference [2] also uses Kalman filter to correct the target tracking whenever CAMShift fails.

CAMShift basically works with a single color histogram as the target representation [6]. Using extra information rather than color histograms to overcome algorithm`s inaccuracy has been a popular extension to CAMShift among researchers. In [3] for instance, an improved version of CAMShift is proposed which uses texture information along with color information to represent and track targets in video sequences. Reference [17] proposes a similar method as a mean-shift extension, mean-shift runs with color and texture feature histograms separately for each target, and the final location of the target is obtained with integration of both mean-shift trackers` results.

III. IMPROVED CAMSHIFT ALGORITHM

In this section we first review traditional CAMShift and its predecessor, the mean-shift algorithm, and then we explain our proposed approach in details.

A. *Mean-Shift and Camshift Overview*

The closest existing algorithm to CAMShift is the mean-shift algorithm. Mean-shift is a robust and non-parametric method of finding local maxima in the density distribution of a data set. It is essentially just hill climbing applied to a density histogram of the data. To locate a target, mean-shift uses iterative searching to find the extreme value of a probability distribution [4].

The main reason of mean-shift failure in tracking targets is the fact that a kernel size that works at one distribution scale may not be suitable for another scale as the targets may move toward or away from the camera [5]. A small fix sized kernel (window) may get lost in tracking large objects, while a large fixed sized kernel may include other foreground or background objects, which causes tracking failure as well.

The main difference of a CAMShift and mean-shift tracker is the fact that CAMShift uses continuously adaptive probability distributions, which means the target`s probability distribution may be recomputed in every frame. This lets the target`s size, shape and appearance change in every frame, while mean-shift uses static probability distributions which are not updated during tracking. But this advantage of CAMShift may become a problem and cause the search window to diverge when there are appearance similarities between targets and the background [5] [6].

To track targets, traditional CAMShift basically works as follows. First, target`s initial search window is selected and its color histogram is computed. Each frame of the sequence afterwards is converted to a probability distribution image relative to the target`s histogram. Then the new size and location of the target are computed via mean-shift from this converted image, and are used as the initial size and location of the target for the next iterations of the algorithm. In the next part we explain our tracking approach and discuss how it outperforms the traditional CAMShift.

B. *Improved CAMShift Algorithm Combined with Motion Segmentation*

Our tracking algorithm is primarily based on conventional CAMShift. The principles of our algorithm can be summarized in the following steps.

1. Initialize the search window`s location and size.
2. Segment the moving parts of the current frame.
3. Calculate the probability distribution image of the current frame.
4. Calculate the new location and size of the target search window using mean-shift.
5. Use the new location and size obtained in step 4 to re-initialize the search window in the new frame, and then jump to step 2.

1) *Search Window Initialization*

In the majority of works proposed on CAMShift, the initial location of a target is selected manually by a user [6] , we have the same approach in our algorithm. After manually locating the target by a surrounding rectangle, its two dimensional color histogram is calculated for further processing in the next steps.

2) *Color Histogram Generation*

To obtain robust results, we use HSV (Hue Saturation Value) color space in our algorithm for the color histogram generation. HSV color space separates out color(H) from its saturation(S) and brightness(V) values [18], which will improve our tracking performance. For target representation, we only choose hue(H) and saturation(S) color channels that represent the target`s main color features.

A color histogram of an image can be calculated as follows:

A histogram bin value $\{\hat{q}_u\}_{u=1\ldots m}$ of an m-bin histogram in an image with n pixels locations $\{x_i\}_{i=1\ldots n}$, is calculated as in (1), where

$c: R^2 \rightarrow \{1 \dots m\}$ is a function that gives the pixel at location x_i the histogram bin index $c(x_i)$.

$$\hat{q}_u = \sum_{i=1}^{n} \delta[c(x_i) - u] \qquad (1)$$

3) Motion Segmentation

To do the motion segmentation in our algorithm we use the image differencing method which is one of the simplest and most used techniques to detect moving objects [19]. The pixel by pixel intensity difference (D_k) of the current frame (f_k) and the reference background model (f_B) is computed from Equation (2), and the resulting image is thresholded to segment the frame`s foreground from its background (Equation (3)).

$$D_k(x,y) = |f_k(x,y) - f_B(x,y)| \qquad (2)$$

$$\text{Result}_k(x,y) = \begin{cases} 0 & \text{background if } D_k \le T \\ 1 & \text{foreground if } D_k > T \end{cases} \qquad (3)$$

The resultant image is a coarse map of temporal changes. Simple morphological operations are applied on this image to reduce noise and fill the holes inside the foreground moving objects. Then the main moving parts are extracted to generate the image of foreground objects. Figure 1 shows a sample resultant foreground image of our motion segmentation step and its corresponding foreground mask.

4) Probability Distribution Image Generation

A common method to generate a probability distribution image is histogram back-projection. Back-projection of an object histogram with a frame generates a probability distribution image in which each pixel`s value associates with the corresponding bin of the object histogram.

To produce the probability distribution image using histogram back-projection, for each pixel location x_i, the corresponding bin u (with the bin value of $\hat{q}_u$) in the image histogram is found, and the value $\hat{p}_u$ is stored in the resultant image at the specified x_i location. $\hat{p}_u$ is calculated from $\hat{q}_u$ as follows:

$$\left\{\hat{p}_u = \min\left(\frac{255}{\max(\hat{q})}\hat{q}_u, 255\right)\right\}_{u=1\dots m} \qquad (4)$$

That is how the histogram bin values are scaled from [0, max($\hat{q}$)] to the valid range [0, 255] of image pixel intensities.

In step 4 we calculate the back-projection of the target histogram with the resultant image of step 2. This will produce a probability distribution image which will be used by mean-shift to calculate a target`s new position.

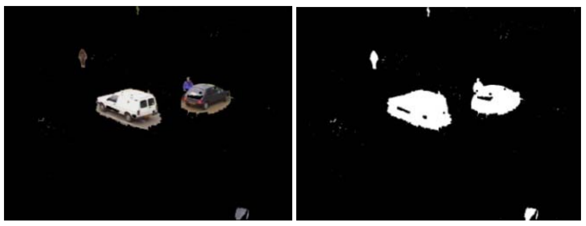


Figure 1. The foreground image(left) of a sample frame obtained from motion segmentation and its corresponding foreground mask(left)

5) Mean-shift Application

To calculate the target`s new size and location the mean-shift technique is used in our algorithm. Mean-shift searches for peaks of a data set in a local search window and ignores data points far away from this window. Meanshift gets a probability distribution image and the initial search window of the target as inputs and returns the target`s new location by iteratively searching the probability distribution image. The method runs in the following steps:

1. Choose a search window.
2. Compute the window's center of mass.
3. Center the window at the new center of mass
4. Return to step 2 until the window stops moving.

Window`s center of mass is computed in step 2, and then the window is re-centered to the computed center of mass. This movement will change what is under the window, and so the re-centering process is repeated until the movement vector converges to zero. The last calculated center of mass will be the new location of the target. It has been proved that the window finally converges and stops moving in step 4 [5].

The process of calculating the mean-shift vector can be simplified to the calculation of image pixel distributions, when dealing with a rectangular kernel [4].

The following equations are used in the calculation of the search window`s center of mass (y_c, x_c):

$$\begin{cases} x_c = \dfrac{M_{10}}{M_{00}} \\ y_c = \dfrac{M_{01}}{M_{00}} \end{cases} \qquad (5)$$

Here the zeroth and first moments are calculated as:

$$\begin{cases} M_{00} = \sum_x \sum_y I(x,y) \\ M_{10} = \sum_x \sum_y xI(x,y) \\ M_{01} = \sum_x \sum_y yI(x,y) \end{cases} \qquad (6)$$

where $I(x,y)$ is the intensity value of point (x,y) in the probability distribution image.

Moments can also be used to find the target`s new size, this gives CAMShift the ability to update the target`s scale during tracking. Search window`s new size can be computed as follows:

$$\begin{cases} l = \sqrt{\dfrac{(a+c)+\sqrt{b^2+(a-c)^2}}{2}} \\ w = \sqrt{\dfrac{(a+c)-\sqrt{b^2+(a-c)^2}}{2}} \end{cases} \qquad (7)$$

where l and w are the long and short axes of the search window respectively, and a, b, and c are obtained from Equation (8).

$$\begin{cases} a = \frac{M_{20}}{M_{00}} - x_c^2 \\ b = 2\left(\frac{M_{11}}{M_{00}} - x_c y_c\right) \\ c = \frac{M_{02}}{M_{00}} - y_c^2 \end{cases} \qquad (8)$$

And the second order moments are calculated from Equation (9):

$$\begin{cases} M_{20} = \sum_x \sum_y x^2 I(x,y) \\ M_{02} = \sum_x \sum_y y^2 I(x,y) \\ M_{11} = \sum_x \sum_y xyI(x,y) \end{cases} \qquad (9)$$

Finally the new size and location of the search window are used to iterate the algorithm from step 2.

C. The method`s Order of Complexity

Our improvements on CAMShift are computationally attractive. The order of complexity of conventional CAMShift is $O(\alpha N^2)$, where α is some constant, and the image is taken to be N×N [5]. Our motion segmentation phase`s order of complexity is $O(N^2)$ due to its background differencing step, which would not increase the whole algorithm`s order of complexity. For motion segmentation we used simple background differencing in our implementation, more efficient motion segmentation techniques may perform better depending on the video data and the scene complexity.

In the next section we discuss how our algorithm tracks targets in different tracking scenarios, and evaluate its performance by comparing our tracking results with the traditional CAMShift and mean-shift tracking results.

IV. Experimental Results

In order to show the improvements of our approach, we have implemented our algorithm with C++ code using OpenCV library, and applied our method on PETS2000 and PETS2001 standard datasets [20].

The ground truth data needed for our comparisons in the following experiments is manually generated for various targets available in the test video sequences with the help of Video Performance Evaluation Resource (ViPER) tool [21]. To compare different tracking algorithms, the binary measure O(GT, ST) of successful or unsuccessful tracker defined as in (10) is used for each frame of a sequence. We define a tracker to be unsuccessful as soon as the spatial overlap between its returned location of the target (ST) and the target`s ground truth bounding box (GT) falls below a predefined threshold value of T. The spatial overlap is computed as in (11).

$$O(GT, ST) = \begin{cases} 1 & \text{if } A(GT, ST) > T \\ 0 & \text{othrewise} \end{cases} \qquad (10)$$

$$A(GT, ST) = \mathrm{Area}(GT \cap ST)/\mathrm{Area}(GT \cup ST) \qquad (11)$$

In all of our experiments T is set to be 0.2. The threshold is constant for all the trackers and we can make fair comparisons.

We have used the following strategy to evaluate the trackers on each video sequence. For each target available in a video sequence, the number of successfully tracked frames and the approximate total number of frames in which the target is available are counted. The process is repeated three times for each target and the average number of successfully tracked frames is calculated. Finally the average counted number of frames for different targets is added together. The trackers are evaluated on all the targets available in PETS 2000 (3 vehicles and 3 pedestrians) and PETS 2001 Camera1 (4 pedestrians, 3 vehicles and 2 groups of pedestrians).

In the first experiment, we ran our proposed CAMShift algorithm, traditional CAMShift and also the mean-shift tracking algorithm simultaneously. We have compared our methods` results with the tracking results of both traditional CAMShift and mean-shift algorithms. Table I summarizes the results of this experiment.

Table I shows that our proposed CAMShift algorithm clearly performs better than the conventional CAMShift and mean-shift tracking algorithms. While all the evaluated trackers performed well in simple tracking scenarios such as tracking the single color vehicles available in the test data, our algorithm outperforms the other ones in more complex situations such as tracking small size pedestrians or groups of pedestrians. Targets with similar colors to the background and targets moving in front of changing color background also cause tracking failure in mean-shift and more significantly conventional CAMShift. But our algorithm performs much better than the other methods in these cases.

Figure 3. shows a case in which the background`s color behind a target is changed during tracking. As shown in the figure, when the target moves in front of the cars with different colors which also have similar colors to the target, traditional CAMShift`s tracking window drifts out of the target area. Our improved algorithm works fine in this case thanks to the motion segmentation step which prevents background color effects on the tracking performance.

As we discussed earlier, we believe our proposed method is robust in combination with a poor object detection method. In the next experiment, we deliberately set the initial search window of a target larger than its actual size. This would cause the existence of some background pixels in the targets` initial search window. A poor object detection in a real world application might be a result of shadows or sudden illumination changes. Table II shows the results of different tracking algorithms in this tracking scenario.

As it is interpreted from Table II, the tracking performances of mean-shift and conventional CAMShift have been considerably decreased in this experiment due to poor object detection. Our proposed algorithm in contrast, shows robustness in initializing with a weak object detection method.

Figure 4. shows a sample sequence of frames in this experiment, in which a small size target is being tracked. The initial size of the search window is deliberately set to be larger than the target`s size to contain some background pixels. As it is shown in the figure conventional CAMShift and the mean-shift algorithm quickly lose the targets` track several frame after the initialization. But our proposed algorithm tracks the target well.

The improvements which have been evaluated in this section can be justified with the fact that our algorithm ignores the undesirable effect of background pixels on the tracker due to the motion segmentation phase. In details, most of the background pixels that are included in the CAMShift`s search window are removed after motion segmentation. Therefore the pixels from the background which may decrease the tracking performance, are not considered in probability distribution image.

CAMShift`s robust ability to track targets with dynamically changing probability distributions also gives the method the ability to track targets in presence of noise. This idea has been evaluated in [5], in the first introduction of the algorithm. In order to evaluate our algorithm`s tolerance to noise in compared with the original CAMShift algorithm, we have created three new video sequences by adding three different amounts of uniform noise to each frame of PETS2000 video sequence. The test of correctly tracked frames is then applied on the new created datasets. The results of this experiment are summarized in Table III.

The results presented in Table III show that as the amount of added noise to the video frames increases, the tracking performances of both CAMShift algorithms degrade. The results discussed in [5] show the addition of noise to the video frames would slightly degrades the CAMShift performance in the case study of face tracking. Our results, in contrast, show in the case of tracking targets in general video surveillance applications, the effect of noise on the tracking performance is more significant.

It should be mentioned here that the presence of noise may considerably spoil the process of motion segmentation with the background subtraction method. The undesirable effect of the added noise on motion segmentation can be clearly seen in Figure 2. The figure shows the result of motion segmentation on a frame with the uniform added noise of 20%. In spite of the degradation of tracking performance in presence of noise for both tracking algorithms, Table III shows our CAMShift algorithm still keeps its superiority in compared with the original method.

This experiment shows that the proposed method preserves the ability of CAMShift in tracking targets in the noisy environments, and our method still outperforms the conventional CAMShift in the presence of noise. Figure 5. shows a case in which conventional CAMShift fails due to presence of noise, but our proposed method tracks the target well.

TABLE I. COMPARISON OF DIFFERENT TRACKING ALGORITHMS

DataSet	Algorithm	Successfully tracked frames (Total evaluated frames)	Success rate
PETS2000	Conventional CAMShift	1009(2750)	36%
	Mean-shift	1470(2750)	53%
	Proposed CAMShift	2613(2750)	95%
PETS2001	Conventional CAMShift	1623(4725)	34%
	Mean-shift	2503(4725)	52%
	Proposed CAMShift	4437(4725)	93%

TABLE II. COMPARISON OF DIFFERENT TRACKING ALGORITHMS INITIALIZED WITH POOR OBJECT DETECTION

DataSet	Algorithm	Successfully tracked frames (Total evaluated frames)	Success rate
PETS2000	Conventional CAMShift	346(2750)	12%
	Mean-shift	940(2750)	34%
	Proposed CAMShift	2487(2750)	90%
PETS2001	Conventional CAMShift	724(4725)	15%
	Mean-shift	1425(4725)	30%
	Proposed CAMShift	4269(4725)	90%

TABLE III. COMPARISON OF CAMSHIFT TRACKERS IN PRESENCE OF NOISE

Data Set	Algorithm	Uniform Added Noise (percent)	Successfully tracked frames (Total evaluated frames)	Success rate
PETS 2000	Conventional CAMShift	10	879(2750)	31%
		20	693(2750)	25%
		50	356(2750)	13%
	Proposed CAMShift	10	2413(2750)	87%
		20	1807(2750)	65%
		50	480(2750)	17%

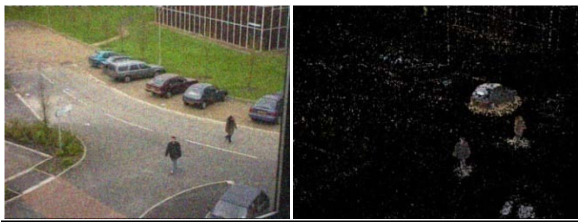


Figure 2. The effect of noise on motion segmentation. Foreground segmented image(right) of a sample frame(left) with 20% of added noise

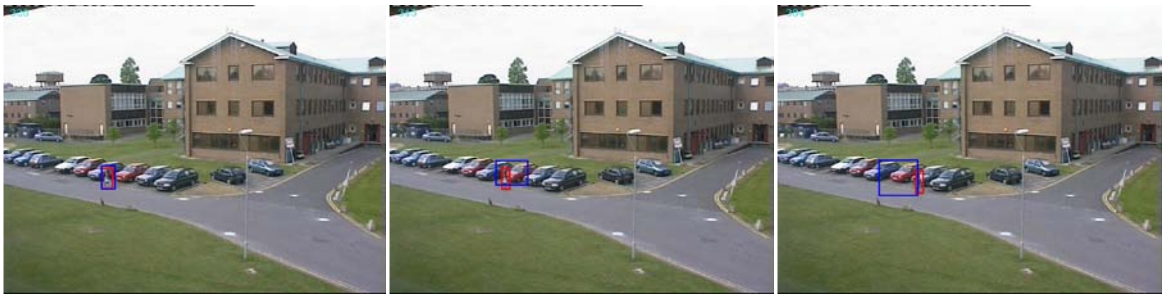


Figure 3. A sample sequence in which improved CAMShift outperforms the Traditional CAMShift(blue rectangle). Search widow is initialized several frames before the first shwon frame.

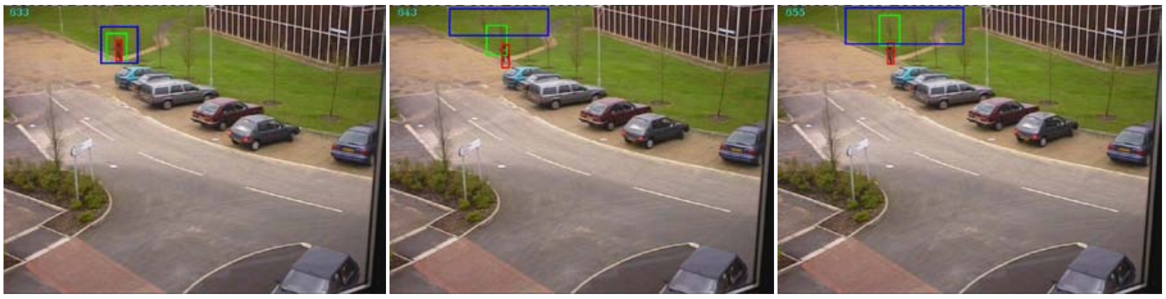


Figure 4. Traditional CAMShift(blue rectangle) and mean-shift(green rectangle) fail in tracking the target due to poor search window initialization, while the improved CAMShift tracks the target successfully (red rectangle). Search widow is initialized several frames before the first shwon frame.

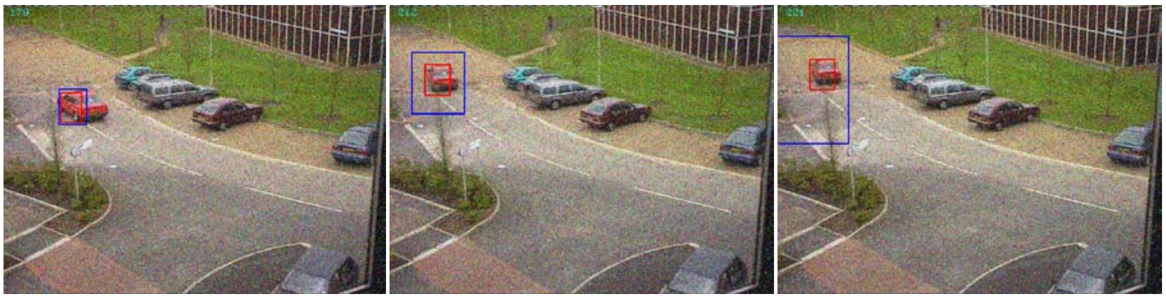


Figure 5. Traditional CAMShift(blue rectangle) fails in tracking the target in the presence 20% noise, while the improved CAMShift tracks the target successfully (red rectangle).

V. Conclusion and Future Works

We proposed an efficient color based CAMShift algorithm for target tracking in this paper. Combined with low-cost motion segmentation techniques, we improved the traditional CAMShift`s performance and showed how our approach can solve its drawbacks. Applying motion segmentation in the algorithm reduces the undesirable effects of background color information on the tracking performance and eases target localization in the probability distribution image. We finally evaluated our algorithm`s performance and compared it with mean-shift and conventional CAMShift in practice.

As we discussed in section 4, although our algorithm performs better than the conventional CAMShift in the presence of noise, but it doesn`t perform satisfactorily when the amount of noise increases. We believe working on the algorithm using more efficient motion segmentation methods such as mixture of gaussian may lead to promising results in the future works. One of CAMShift`s drawbacks is its inaccuracy in locating the exact area of a target. For example when dealing with generation of ground plane trajectory, the exact location of a target`s feet is needed, which can`t be obtained accurately using the conventional CAMShift. We believe using some modifications in our algorithm; we can handle these inaccuracies in future. We are also going to work on the occlusion handling of our algorithm. Our algorithm can handle partial occlusions exist in our test data. we are going to work on the method to handle more severe occlusions, especially when moving targets are occluded by static objects.

References

[1] J A. Yilmaz, O. Javed, and M. Shah, “Object Tracking: A Survey,” ACM Computing Surveys, vol. 38, p. 13 . 2006.

[2] S. Huang; and J. Hong; , "Moving object tracking system based on camshift and Kalman filter," International Conference on Consumer Electronics, Communications and Networks (CECNet), pp.1423-1426, 2011.

[3] J. Yin, Y. Han, J. Li, and A. Cao, "Research on Real-Time Object Tracking by Improved CAMShift," International Symposium on Computer Network and Multimedia Technology, pp.1-4, 2009.

[4] G. Bradski, and A. Kaehler, "Learning OpenCV", O`Reilly, 2008, pp. 337-341.

[5] G. R. Bradski. "Computer vision face tracking for use in a perceptual user interface". Intel Technology Journal, 2nd Quarter, 1998.

[6] G. J. Allen, Y. D. Richard Xu and S. Jin Jesse, “Object Tracking Using CAMShift Algorithm and Multiple Quantized Feature Spaces”, Inc. Australian Computer Society, vol.36, 2004.

[7] D. Exner, E. Bruns, D. Kurz, A. Grundhofer, and O. Bimber, "Fast and robust CAMShift tracking", IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp.9-16, 2010.

[8] W. Hu, T. Tan, L. Wang, and S. Maybank; , "A survey on visual surveillance of object motion and behaviors," Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on , vol.34, no.3, pp.334-352. 2004.

[9] D. Comaniciu, V. Ramesh, and P. Meer, , "Kernel-based object tracking," Pattern Analysis and Machine Intelligence, IEEE Transactions on , vol.25, no.5, pp. 564- 577, 2003.

[10] S. Dongcheng, L. Yinghuan, L. Chao, and Zh. Long; , "A modify target tracking algorithm based on anti-jamming CamShift in dynamic scene," 3rd International Congress on Image and Signal Processing (CISP), vol.3, pp.1482-1484, 2010.

[11] I. Leichter, M. Lindenbaum, and Ehud Rivlin, "Mean Shift tracking with multiple reference color histograms", Computer Vision and Image Understanding, v.114 n.3, p.400-408, 2010.

[12] D. Comaniciu, and V. Ramesh, "Mean shift and optimal prediction for efficient object tracking," International Conference on Image Processing, pp.70-73 vol.3, 2000.

[13] A. Ali, Aand K. Terada, "A framework for Human tracking using Kalman filter and fast mean shift algorithms," IEEE 12th International Conference on Computer Vision Workshops (ICCV Workshops), pp.1028-1033, 2009.

[14] W. Xiangyu, and L. Xiujuan, "The study of moving target tracking based on Kalman-CAMShift in the video," 2nd International Conference on Information Science and Engineering (ICISE), pp.1-4, 2010.

[15] G. Welsh, and G. Bishop, “An Introduction to the Kalman Filter” (Thechnical Report TR95-041), University oF North California, Chapel hill, NC, 1995.

[16] L. Wang, S. Hu, and X. Zhang, "Detecting and Tracking of Small Moving Target Under The Background of Sea", 9TH International Conference on Signal Processing , 989-992, 2008.

[17] X. Zhang, Y. Dai, Zh. Chen, and H. Zhang; , "An improved Mean Shift tracking algorithm based on color and texture feature," 2010 International Conference on Wavelet Analysis and Pattern Recognition (ICWAPR), pp.38-43, 2010.

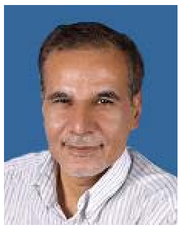
[18] A.R. Smith, "Color Gamut Transform Pairs," SIGGRAPH 78, pp. 12-19, 1978.

[19] L. Zappella, X. Lladó, and J. Salvi. "Motion Segmentation: A Review". 11th International Conference of the Catalan Association for Artificial Intelligence(CCIA'08), 2008.

[20] Ferryman, J., IEEE Int’l Workshops on Performance on Performance Evaluation of Tracking and Surveillance, 2000-2004.(URL: http://www.cvg.cs.rdg.ac.uk/VSPETS/).

[21] D. Doermann, and D. Mihalcik, , "Tools and techniques for video performance evaluation," Proceedings of 15th International Conference on Pattern Recognition, vol.4, pp.167-170, 2000.

Ebrahim Emami received the B.Sc. degree in computer engineering from Shahid Bohonar University of Kerman, Kerman, Iran, in 2009, and the M. Sc. degree in artificial intelligence from Iran University of Science & Technology, Tehran, Iran, in 2012. His research interests are in signal, image and video processing, computer vision particularly intelligent video surveillance systems, and machine learning.



Mahmood Fathy received his B.Sc. degree from Iran University of Science & Technology, Tehran, Iran, in 1984, his M.Sc. degree from Bradford University, West Yorkshire, U.K., in 1987, and his Ph.D. degree in image processing and computer architecture from the University of Manchester Institute of Science and Technology, Manchester, U.K., in 1991. Since 1991, he has been an Associate Professor with the Department of Computer Engineering, Iran University of Science & Technology. His research interests include the quality of service in computer networks, the applications of vehicular ad hoc networks in intelligent transportation systems, and real-time image processing, with particular interest in traffic engineering, bio informatics, and bio computers.