Object Tracking Using Improved CAMShift Algorithm Combined with Motion Segmentation

Ebrahim Emami, Mahmood Fathy
Computer Engineering Department
Iran University of Science and Technology
Tehran, Iran
E-mail: ebrahim_emami@comp.iust.ac.ir, mahfathy@iust.ac.ir

Abstract—Continuously adaptive mean-shift(CAMShift) is an efficient and light-weight tracking algorithm developed based on mean-shift. 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 a motion segmentation phase, we proposed an improved CAMShift algorithm to cope with CAMShift`s tracking problems. We evaluated the efficiency of our approach by comparing our tracking results with the traditional algorithm`s results in several cases

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 automated surveillance, human computer interaction, behaviour analysis and activity recognition. 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 targets by finding the most similar distribution pattern in a frame sequences with its sample pattern by iterative searching. It has been widely used 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, 2].

Based on mean-shift, continuous adaptive mean-shift (CAMShift) was proposed to overcome the problem. CAMShift adaptively adjusts the track window`s size and the distribution pattern of targets during tracking. CAMShift algorithm can be used to track the distribution of any kind of feature that represents the target in a lightweight, robust and efficient way[2]. Most researchers though, use color data to represent targets for CAMShift [3], this would give a low complexity and practical performance to the method.

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 different color background objects, 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 [4], 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 [5].

In this paper we present an improved CAMShift algorithm to overcome the above mentioned drawbacks. 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 discussed our approach`s experimental performance and compare our algorithm`s results with the traditional CAMShift`s.

II. RELATED WORK

CAMShift was first introduced as a technique for face and head tracking in a perceptual user interface as a mean-shift extension [6]. Since then it has been object to a variety of modifications and improvements to accommodate other tracking applications. In this section we review some of these extensions related to our work.

As mentioned earlier, traditional CAMShift fails to track targets in several complex situations such as tracking objects which have similar colors with the background [4,7]. Several works have been done to improve CAMShift`s performance and accuracy in these cases. CAMShift basically works with a single color histogram as the target representation [4]. Using extra information rather than a single color histogram 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

978-1-4577-1535-8/11/$26.00 ©2011 IEEE

information along with color information to represent and track targets in frame sequences.

The approach proposed in [5], in the other hand, 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.

To reduce the target and background color interference, a popular approach is to generate the target color histogram with a weighted scheme [1]. 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.

Several approaches combine other simple tracking methods with CAMShift to improve the tracking performance, the approaches proposed in [7,8] for example, combine CAMShift algorithm with Kalman filter. In [7], the possible positions of a target are predicted by Kalman filter, and then CAMShift is used to search and match the target in the predicted areas. The approach proposed in [8], in contrast, uses Kalman filter only when CAMShift fails to track targets.

III. IMPROVED CAMSHIFT ALGORITHM

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

A. Mean-Shift and Camshift Overview

The closest existing algorithm to the CAMShift algorithm is the mean-shift algorithm [9]. 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 [2].

The main difference of a CAMShift and mean-shift tracker is 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 [4, 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[2,4]. 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 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[4], we also use 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[10], which would improved our tracking performance. For target representation, we only choose hue(H) and saturation(S) color channels that represent the target`s main color features.

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 [11]. 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 (1), and the resulting image is thresholded to segment the frame`s foreground from its background (Equation (2)).

$$D_k(x,y) = |f_k(x,y) - f_B(x,y)| \quad (1)$$

$$Result_k(x,y) = \begin{cases} 0 \text{ background if } D_k \le T \\ 1 \text{ foreground if } D_k > T \end{cases} \quad (2)$$

The result is a coarse map of temporal changes, after some simple morphological operations on this image, the main moving parts are extracted to generate the image of foreground objects.

4) Probability Distribution Image Generation

A common method to generate a probability distribution image is histogram back projection. Histogram back projection of the target histogram with a frame generates a probability distribution image in which each pixel`s value associates with the corresponding bin of the target histogram [4].

In step 3 we calculate the back-projection of the target histogram with the resulting image of step 2. This will produce

a probability distribution image which will be used by mean-shift to calculate the target`s new position.

5) Mean-shift Application

To calculate the new location of a target, the mean-shift algorithm is used. Mean-shift takes a probability distribution image and an initial search window, computes the window`s center of mass, and then re-centers the window at 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 [2].

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

$$\begin{cases} x_c = \dfrac{M_{10}}{M_{00}} \\ y_c = \dfrac{M_{01}}{M_{00}} \end{cases} \tag{3}$$

Here the zeroth and first moments are calculated as:

$$\begin{cases} M_{00} = \sum_x \sum_y I(x,y) \\ M_{10} = \sum_x \sum_y xI(x,y) \\ M_{01} = \sum_x \sum_y yI(x,y) \end{cases} \tag{4}$$

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

Search window`s new size can be computed as follows:

$$\begin{cases} l = \sqrt{\dfrac{(a+c)+\sqrt{b^2+(a-c)^2}}{2}} \\ w = \sqrt{\dfrac{(a+c)-\sqrt{b^2+(a-c)^2}}{2}} \end{cases} \tag{5}$$

where l and *w* are the long and short axes of the search window respectively, and a, b, and c are obtained from Equation (6).

$$\begin{cases} a = \dfrac{M_{20}}{M_{00}} - x_c^2 \\ b = 2\left(\dfrac{M_{11}}{M_{00}} - x_c y_c\right) \\ c = \dfrac{M_{02}}{M_{00}} - y_c^2 \end{cases} \tag{6}$$

And the second order moments are calculated from Equation (7):

$$\begin{cases} M_{20} = \sum_x \sum_y x^2 I(x,y) \\ M_{02} = \sum_x \sum_y y^2 I(x,y) \\ M_{11} = \sum_x \sum_y xyI(x,y) \end{cases} \tag{7}$$

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

In the next section we discuss how our algorithm tracks targets in several cases, and compare our tracking results with the traditional CAMShift`s.

IV. Experimental Results

In order to prove the improvements of our approach over traditional CAMShift, we implemented our algorithm with C++ code using OpenCV library, and applied our method on various video sequences. We compared our tracking results with the tracking results of traditional CAMShift available at OpenCV library. While both algorithms perform well in simple situations, our experiments showed that our improved CAMShift algorithm outperforms the traditional one in more complex situations. In the remaining part of this section we discuss several cases in which traditional CAMShift failed in tracking targets while our method tracked the targets successfully.

Figure 1 shows a case 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. Our experiments show that after several frames, traditional CAMShift quickly diverges to the background, while our proposed algorithm tracks the target very well. This shows our algorithm`s better performance and robustness in combination with a poor object detection module which normally causes traditional CAMShift`s failure.

Another case in which traditional CAMShift fails in tracking is shown in figure 2. This is the case when the background`s color behind the target is changed during tracking. As shown in the figure, when the target moves in front of the cars with different colors, 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 mentioned earlier one of the main drawbacks of CAMShift arises when targets have similar colors with the background. Our improved CAMShift algorithm shows promising results in such conditions. Figure 3 shows an example of this situation in which our algorithm outperforms the traditional CAMShift.

All the improvements which are discussed in some typical cases in this section, can be proved to work well with the fact that our algorithm ignores the undesirable background pixels` effects 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 and may decrease the tracking accuracy, are removed after the motion segmentation phase and are not considered in probability distribution image.

Figure 1. Traditional CAMShift diverges and fails in tracking the target due to poor search window initialization (blue rectangle), improved CAMShift tracks target successfully (red rectangle).

Figure 2. Background color changes affect traditional CAMShift`s performance (blue rectangle), improved CAMShift continues tracking correctly (red rectangle).

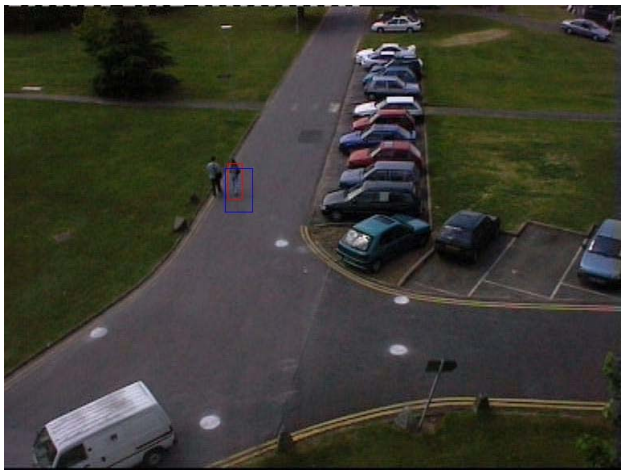

Figure 3. Background and target color similarities cause traditional CAMShift`s failure (blue rectangle), improved CAMShift continues tracking correctly (red rectangle).

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 in practice, and showed how our approach can stand tracking failures in various situations in which traditional CAMShift would normally fail.

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 $N \times N$ [6]. 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 future works we are going to work on the occlusion handling of our algorithm. We believe our algorithm, with a few changes, would promisingly be able to handle occlusions, especially when 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] G. Bradski, and A. Kaehler, "Learning OpenCV", O`Reilly, 2008, pp. 337-341.

[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. 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.

[5] 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.

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

[7] 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.

[8] 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.

[9] Y. Cheng, "Mean shift, mode seeking, and clustering," IEEE Trans. Pattern Anal. Machine Intell., 17:790-799, 1995.

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

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