Object tracking using improved Camshift with SURF method

Jianhong Li, Ji Zhang, Zhenhuan Zhou, Wei Guo, Bo Wang and Qingjie Zhao
Beijing Lab of Intelligent Information Technology
School of Computer Science, Beijing Institute of Technology, Beijing, P.R. China

***Abstract*—Camshift is an effective algorithm for real time dynamic target tracking applications, which only uses color features and is sensitive to illumination and some other environment factors. When similar color existing in the background, traditional Camshift algorithm may fail, that is the target getting lost. To solve the problem, an improved Camshift algorithm is firstly proposed in this paper to reduce the influence of illumination interference. Besides, a method judging whether the target is lost is also proposed. Once the target is judged lost, the Speeded Up Robust Features (SURF) is utilized to find it again and the improved Camshift keeps on tracking the target continuously. SURF is invariant to scale, rotation and translation of images. We program in C++ based on OpenCV. The results prove that the proposed method is more robust than the traditional Camshift and give better tracking performance than some other improved methods.**

***Keywords*—target tracking; improved Camshift; SURF**

I. INTRODUCTION

Due to the prospect of wide applications and significant role it plays in industrial applications, target tracking is attracting more and more attention in fields of video and image processing as well as human-computer interaction. Camshift(Continuously Adaptive Meanshift) is a target tracking algorithm proposed by Bradski [1], using color histogram as its target model. Since the histogram of a target picture is the recording of color probability, this algorithm may not be easily influenced by the shape changes of the target, and hence can effectively solve the problem that the target is moving or partly sheltered. Besides, this algorithm can achieve the purpose of target tracking fast, since it can reduce computation time by using the gradient descent method. Although Camshift is effective in real-time target tracking, it inevitably has the following two main defects, since it only uses the color characteristics of the target: First, the tracking result is not satisfactory when the target and background are alike in color; Second, the target might be lost in dynamic background. Camshift algorithm is very sensitive to illumination and some other environmental changes.

In this paper, we proposed an improved Camshift algorithm, aiming at making Camshift well-adapted to the environmental changes. Traditional Camshift algorithm cannot handle the problem when the target is interfered by those with similar color which lead to the loss of the target. The main reason is that traditional Camshift algorithm only uses the color information, while other factors, such as motion information and texture features, are not considered at all. Some studies present algorithms which combine Camshift and motion information [2,3]. We propose the addition of texture information, and such idea has been applied in other studies [4-7], to make target tracking more accurate. However, these studies all use SIFT(Scale-invariant feature transform) as matching algorithm which is slow in computation. We propose a method to judge whether the target in Camshift is lost. Once the target is lost, we can re-match the target using SURF(the Speeded Up Robust Features) algorithm. SURF feature-matching algorithm has the characteristic that it can remain invariant to target rotation, scale zoom and brightness variations. Our experiment result shows that the proposed method is able to accurately locate and track targets. This method is significantly superior to traditional Camshift algorithm, and has good robustness as well.

OpenCV [8] is an open-source computer vision library developed by Intel. It consists of a series of C functions and a few C++ classes, and can realizes many common algorithms in image processing and computer vision fields. OpenCV includes more than 300 cross-platforms, middle-level and high-level API written in forms of C functions. It does not depend on other external libraries, although it can refer to certain ones. Besides, OpenCV is free for both non-commercial and commercial applications. In addition, OpenCV provides transparent interfaces for Intel Integrated Performance Primitives (IPP), which means it can automatically load the IPP libraries when they are available for the optimization of some specific processors. And OpenCV provides the implementation code of traditonal Camshift.

The organization of this paper is as follows: Section II gives the brief overview of traditional Camshift method and also introduce our improved method in detail. The approach of how to judge the lost of the target and find it again using SURF Feature is presented in Section III. Finally, we provide extensive experimental results in Section IV.

II. AN IMPROVED CAMSHIFT ALGORITHM

Camshift was firstly proposed by Gary R. Bradski in 1988 [1]. It is based on a robust non-parametric technique for climbing gradient to find the mode(peak) of probability distributions called Meanshift. Since the Hue in HSV color model is not influenced by illumination, color is represented as Hue value.

This work was supported by the National Natural Science Foundation of China (61175096).

978-1- 61284-495-4/11/$26.00 ©2011IEEE

However, simply ignoring saturation and brightness(S and V values) brings a problem. As mentioned by Bradski, when brightness is low(V near 0), saturation is also low(S near 0), Hue then becomes quite noisy since in such a small hexcone, the small number of discrete hue pixels cannot adequately represent slight changes in RGB. It was also mentioned in many papers later [3,9], but most of them used the same solution that Bradski proposed: HSV brightness and saturation thresholds are employed since hue is not well defined for very low or high brightness or low saturation.

To solve such a problem, we improve the Camshift algorithm and propose a new method, which is able to adaptively change the threshold of brightness and saturation(S and V) during the initialization period of Camshift fitted to different environments and thus the robustness get improved.

A. Traditional Camshift Algorithm

Camshift uses color histogram as its target model. Color histogram calculation is one of basic image operations. Suppose there are n pixels in an image, each of which is made up of H, S and V values, and the histogram is an m-bin one. The H value of each pixel is counted into the histogram only if the S and V values of that pixel are under the initial thresholds, else the H is counted directly as 0. We define the H, S and V values of each pixel as $\{h_i\}_{i=1...n}$,$\{s_i\}_{i=1...n}$ and $\{v_i\}_{i=1...n}$, the histogram is $\{q_k\}_{k=1...m}$, and define a function $c : \Re^2 \rightarrow \{1...m\}$ that associates to the pixel at location x_i^* and the histogram bin index $c(x_i^*)$. Then we compute the color histogram:

$$q_k = \sum_{i=1}^{n} \delta[c(h_i)-k]u[s_i-s_{min}]u[v_i-v_{min}]u[v_{max}-v_i]. \quad (1)$$

$$u(t) = \begin{cases} 1 & t \geqslant 0 \\ 0 & t < 0 \end{cases}. \quad (2)$$

$$\delta(t) = \begin{cases} 1 & t \geqslant 0 \\ 0 & t < 0 \end{cases}. \quad (3)$$

The color probability distribution represents the probability of being the target. For better observation, every value in the function is multified by 255 to make up a gray image, which is defined as Back Projection. The back projection is generated by the color histogram. As is defined above, the H, S and V values of each pixel are $\{h_i\}_{i=1...n}$, $\{s_i\}_{i=1...n}$ and $\{v_i\}_{i=1...n}$, the histogram is $\{q_k\}_{k=1...m}$ and the histogram bin index $c(x_i^*)$. Now we define the back projection value of each pixel as $\{b_i\}_{i=1...n}$, then the formula of computing back projection is

$$b_i = \frac{q_c(h_i)}{n} * 255 \quad (4)$$

The search window of Camshift is an ellipse, and the calculation of window's location is a converging process. Each time the new window is computed according to the window of the last frame, and the new window is located when the difference of the two windows is below a pre-set threshold. The specific procedure in mathematical form is shown below, where $I(x, y)$ is each pixel's value in back projection.

0-order moment:

$$M_{00} = \sum_x \sum_y I(x, y). \quad (5)$$

1-order moments:

$$M_{01} = \sum_x \sum_y yI(x, y). \quad (6)$$

$$M_{10} = \sum_x \sum_y xI(x, y). \quad (7)$$

2-order moments:

$$M_{20} = \sum_x \sum_y x^2 I(x, y). \quad (8)$$

$$M_{11} = \sum_x \sum_y xyI(x, y). \quad (9)$$

$$M_{02} = \sum_x \sum_y y^2 I(x, y). \quad (10)$$

Now the center's coordinate of the search window is

$$x_c = \frac{M_{10}}{M_{00}}, y_c = \frac{M_{01}}{M_{00}}. \quad (11)$$

Since the search window is ellipse-sized, the length l and the width w is

$$l = \sqrt{\frac{(a+c) + \sqrt{b^2 + (a-c)^2}}{2}}. \quad (12)$$

$$w = \sqrt{\frac{(a+c) - \sqrt{b^2 + (a-c)^2}}{2}}. \quad (13)$$

where $a = \frac{M_{20}}{M_{00}} - x_c^2, b = 2(\frac{M_{11}}{M_{00}} - x_c y_c), c = \frac{M_{02}}{M_{00}} - y_c^2$.

The above procedure continues until the window converges.

The procedure of Camshift is:

(1) Choose the initial search window size and location to set the calculation region. Set the thresholds of S and V values, and only Hue under the thresholds is effective. Then calculate the color histogram of such region.

(2) Calculate the color probability distribution function of the current image's color histogram.

(3) Find the mass center within the search window, then center search window at the mass center, and compute the area.

(4) If the window converges, return to (2), else return to (3).

The procedure is summarized as Fig.1 below.

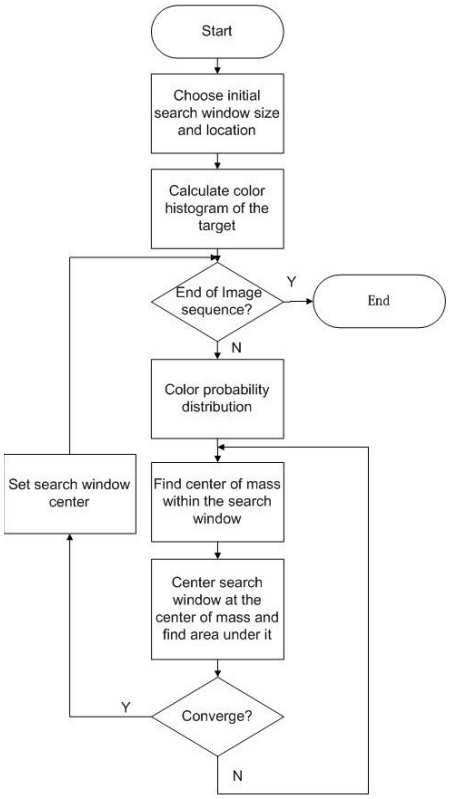

Fig. 1. Block diagram of Camshift

B. Improved Camshift Algorithm

Traditional Camshift only computes Hue value. As to S and V, it sets thresholds for them, and only those pixels whose S and V are below the thresholds are judged effective. For different illumination, it is necessary to artificially set the thresholds of S and V in the initializing period in order to maximize the effective pixels. However, when illumination changes during the tracking, traditional Camshift is not robustic against the background's noise. Thus, it cannot do the tracking automatically, let alone guarantee a good tracking quality.

In this paper we propose an improved Camshift Algorithm to solve this problem. Based on the fact that Hue value is very easy to be influenced by illumination fluctuation, we consider the new algorithm being able to adjust the thresholds of S and V adaptively against the environment changes. Specifically, when under a strong illumination, the algorithm ignores those pixels with high S values, and when under weak illumination it ignores those with low S.

Under such consideration, we respectively change the thresholds of S and V after the search window is initialized. When the ratio of back projection value between the whole image and the search window reaches the minimum, the adjustment finishes. However, many experiments prove that the performance of this method is not as good as expected, sometimes only part of the object is recognized. This is because both the pixels inside and outside the search window would decrease when the thresholds of S and V change, then there is too few pixels to represent the whole object. To solve this problem we find that the ratio of the effective pixels to all the pixels in the search window is a good evaluation of the object's completeness, thus we set its minimum to 0.3 to ensure there are enough pixels in the search window.

In practice, the change of S and V only by one unit may cost lots of time. To achieve a better performance, we change the value of S and V by ten units for one step. We also find that when either S or V is below 30, there is usually much noise. And when either of them is larger than 120, the object is usually incomplete. So we set both S and V between 30 and 120. And our algorithm can find the best value to reach the best tracking performance.

III. SURF Feature

SIFT [10] is an image feature proposed by Lowe that is invariant to rotation and scale. It is often used in image matching and connection, but the complexity and data are usually very large. SURF [11,12] is the speeded up and robust SIFT, which can offer a better perfomance than SIFT. The calculation of SURF feature consists of keypoints detection, descriptor generation and keypoints matching.

A. Keypoints Detection

Using the integral image, the task of calculating the area of an upright rectangular region is reduced to four operations, and the calculation of first-order Haar wavelet response will be six operations.

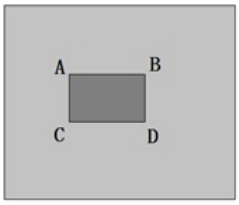

Fig. 2. Area computation using integral images

The integral image of image $I(x,y)(0 \leqslant x \leqslant M, 0 \leqslant y \leqslant N)$ can be defined by the formula:

$$I_{\sum}(x,y) = \sum_{i=0}^{i \leqslant x} \sum_{j=0}^{j \leqslant y} I(x,y). \tag{14}$$

Fig.2 shows how to perform fast pixel intensities, which can be calculated by:

$$\sum_{ABCD} = I_{\sum}(A) + I_{\sum}(D) - [I_{\sum}(B) + I_{\sum}(C)]. \tag{15}$$

The image's scale-space, which is also called Gaussian pyramid, is mainly used to find interest points in different scales. In SIFT, the image is iteratively convolved with Gaussian kernel and repeatedly sub-sampled. This method results in that each layer relies on the previous, and thus, the complexity is very

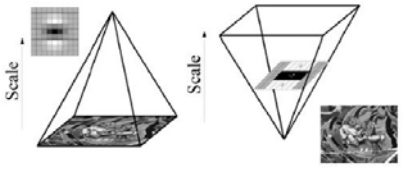

Fig. 3. Gaussian pyramid

large. In SURF, kernels can be changed in size to create the Gaussian pyramid. As Fig.3 shows, Laplacian of Gaussian is approximated to the box filter. This allows for multiple layers of the scale-space pyramid to be processed simultaneously and negates the need to subsample the image, and hence it has a better performance.

The determinant of Hessian is used at the interest points localization to determine whether a point is extremum. Suppose $f(x, y)$ is a continous function with two variables, then the Hessian matrix is:

$$H(f(x,y)) = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix}. \tag{16}$$

Its determinant is:

$$H(f(x,y)) = detH = \frac{\partial^2 f}{\partial x^2}\frac{\partial^2 f}{\partial y^2} - \frac{\partial^2 f}{\partial x \partial y}^2. \tag{17}$$

if $detH < 0$, which means the eigenvalues of H have different signs, and then the point is not a local extremum. Otherwise it is an extremum.

Replacing $f(x, y)$ with $I(x, y)$, then the Hessian matrix of the image is:

$$H(x,\sigma) = \begin{bmatrix} L_{xx}(x,\sigma) & L_{xy}(x,\sigma) \\ L_{xy}(x,\sigma) & L_{yy}(x,\sigma) \end{bmatrix}. \tag{18}$$

Here $L_{xx}(x,\sigma) = \frac{\partial^2 g}{\partial x^2} * I(x,y)$, $L_{xy}(x,\sigma) = \frac{\partial^2 g}{\partial x \partial y} * I(x,y)$, $L_{yy}(x,\sigma) = \frac{\partial^2 g}{\partial y^2} * I(x,y)$. The derivative is calculated as Laplacian of Gaussian. After they are picked out, each of the extremums is compared with the 26 points adjacent to it (Shown in Fig.4). If the value of any extremum point is above or below all of the 26 adjacent points, then the extremum point becomes an interest point.

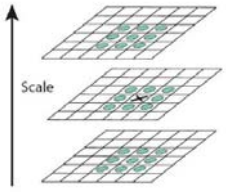

Fig. 4. Comparing the value of an extremum point with the 26 adjacent points

B. SURF Interest Point Descriptor

SURF interest point descriptor relies on the dominant orientations of all the interest points. The descriptor component is then builded. The task of calculating dominant orientation

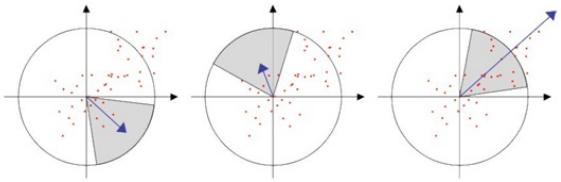

Fig. 5. Orientation Assignment

is based on Haar wavelet responese. It calculates the Haar response in both X and Y coordinates in the circle region centered at interest points with a radius of 6σ. The size of Haar wavelet is 4σ, and the sum of vectors is calculated in every 60 degrees in the circle. Finally, the orientation with the largest sum of vectors is the dominant orientation. The process is shown in Fig.5.

After the dominant orientation is determined, a square window is constructed centered at each interest point with a side length of 20σ. Then it is divide into 4×4 sub-region and the wavelet response is calculated in both the dominant orientation and the orientation vertical to it. If we define the wavelet of x and y as dx and dy, then there will be 4 values $\sum dx$,$\sum dy$,$\sum |dx|$,$\sum |dy|$, and totally it will be a 64-length vector for each interest point. Normalizing it, then we gain the descriptor component. The SURF descriptor is invariant to scale, rotation and translation of images.

IV. Improved Camshift with SURF method

Improved Camshift is able to automatically change its parameters to fit different illumination cases. However, some problems still remain to be solved. First, tracking performance is not ideal when the background is very similar to an object. Second, an object is easy to be lost under dynamic background because Camshift uses Hue as the only feature in tracking. SURF uses a much more robust feature. When it is added to Camshift, the accuracy of the whole tracking system is highly improved.

A. Determination of Object Lost

The object lost can usually be divided into 2 kinds of situation. On the one hand, the confusion of an object with background when they are much similar in Hue; On the other, when the object moves too fast, the object region of 2 continous frames do not overlap so that the search window may not converge to catch up the movement. To solve such two problems, we propose two methods respectively.

For the first situation, we consider evaluating the similarity between the target and the candidate object by Bhattacharrya distance. Suppose that we use m-bin color diagram. The target's normalized color diagram is $p(u)(u = 1...m)$, and

the candidate object's normalized color diagram is $q(u)(u = 1...m)$, then the Bhattacharrya distance is:

$$\ell(p,q) = \sum_{u=1}^{m} \sqrt{p(u)q(u)}. \quad (19)$$

Ideally when the candidate object is all the same with the target, $\ell(p,q)$ reaches maximum, but actually it can hardly be reached. We set 0.8 as the threshold. When $\ell(p,q)$ is below 0.8, we think the object lost.

B. Improved Camshift with SURF Method

Algorithm 1: Improved Camshift with SURF method

Input: Template image
Output: Location of the target

```
1  begin
2      Read the Template image and the first frame of the
       Image sequence, calculate the location of the target
       using SURF method;
3      Calculate the color histogram of the Template image,
       calculate the thresholds of S and V, filter the noise of
       the background;
4      while next frame do
5          Update the size and the location of the search
           window using Camshift;
6          Calculate Bhattacharrya distance according to
           Eq.(19);
7          If Bhattacharrya distance < 0.8 then
8              lost the target, recalculate the location of the
               target using SURF method;
9          end if
10     end while
11 end
```

Algorithm 1 shows the process of our algorithm. When the object is lost, SURF is used to search in the whole image to match the target, When matching points are found, the center of the points is calculated and the search window is located around it. Then Camshift continues to work. Under such a method, the system can gain good real time characteristic as well as robustness.

V. Experiment and Analysis

In this chapter, we carry the following experiments to show our algorithm's performance. In the experiments, we improve the traditional Camshift code in OpenCV library.

A. Larget target tracking

Fig.6 shows the contrast performance between traditional Camshift and our improved Camshift. In Fig.6, subfigure (a) and (d) are two normal color images at the beginning of tracking in two different environments. Subfigure (b) and (e) are the back projections of subfigure (a) and (d) using traditional Camshift, and subfigure (c) and (e) show the results using the improved Camshift. We can see that, because of the similarity between the Hues of the object's face and the curtain, there is a strong noise that could badly influence the tracking in the (b) and (e). Such influence lies in that the noise makes the system easily confuse the object with the background. In (c) and (f), using our improved Camshift, the noise is largely reduced and almost disappears. From these contrast groups we can see that, because of the ability to automatically adapt to the illumination, the system becomes more robust to noise.

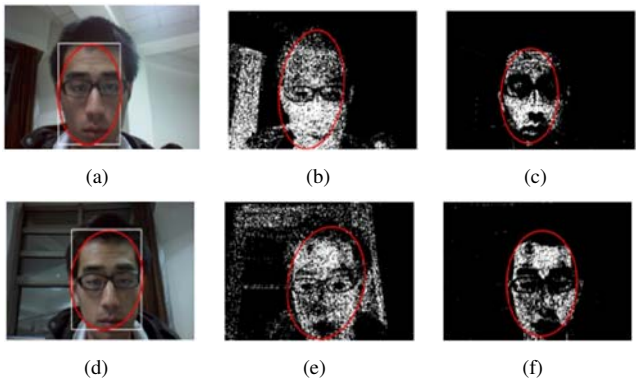

Fig. 6. (a) and (d) are normal color images of two different environments, (b) and (e) are their back projections of traditional Camshift, and (c) and (f)are the back projections of the improved Camshift which is able to automatically adapt to the environments.

B. Small target tracking

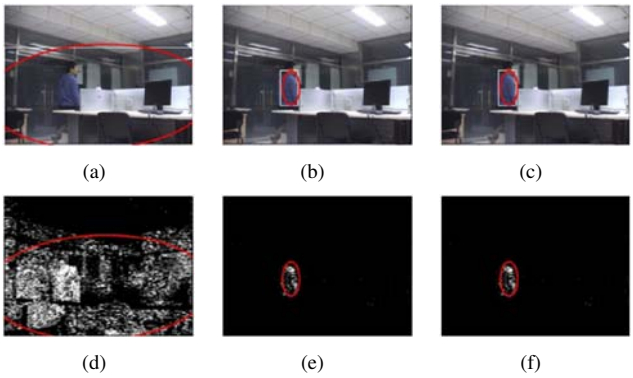

Fig. 7. The image of 160th frame. (a) and (d) shows the result of traditional Camshift, (b) and (e) shows that of improved Camshift, and (c) and (f) shows that of improved Camshift with SURF.

In Fig.7, the ellipse locates the target recognized by the system while the box represents the predicted position of the target in the next frame. We can see that since there is much noise in the background which traditional Camshift is unable to avoid, the located region is too large for the target. In contrast, with the ability of automatically reducing the noise to the minimum, improved Camshift and improved Camshift with SURF accurately capture the target and thus give better performance.

(a) (b) (c)

Fig. 8. The image of 300th frame. (a) shows the result using traditional Camshift, (b) shows that of improved Camshift, and (c) shows that of improved Camshift with SURF.

C. Performance of Improved Canshift with SURF

When an object with a similar Hue covers the target or the target moves very fast, the target is easy to be lost. After it is lost, traditional and improved Camshift is unable to refind the target. But it is easy to do that by matching the target using SURF feature. Fig.8 shows the three experimental results. In (a), traditional Camshift regards the large area of noise as the target and locates search window at it; In (b), improved Camshift is able to disregard noise, but since the target has been behind the background in several frames, it recognizes no target in the following frames and locates nothing; In (c), when the system judges the object is lost, matching program is activated to use SURF to refind the target, and then tracking program continues to work. We use Bhattacharrya distance to

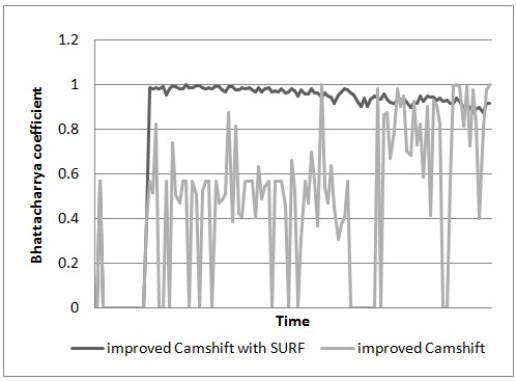

Fig. 9. Bhatacharrya coefficient graph

evaluate the similarity of the candidate object and the target. Fig.9 shows the Bhattacharrya distances after a lost happens. Improved Camshift regards no target in the image after losing happens. So the curve severely fluctuates, which means the tracking system is not stable. Improved Camshift with SURF has the same unstability with the former but quickly goes to a steady value that is close to 1. This is because the system succcessfully refinds the target and locates it. As a result, we consider the improved Camshift with SURF gives the best performance.

In summary, traditional Camshfit performs well under a simple background with a proper sized target, but it is not robust when the target is small and there is much confusing noise. Besides, object losing is also a problem. Improved Camshift can solve the former, but can do nothing on the latter. Improved Camshift with SURF not only removes most of noise but also refinds the target when it is lost. Thus, it is an accurate and robust algorithm.

VI. Conclusion

In this paper, we propose an improved Camshift algorithm with SURF method. It is an effective and robust algorithm. It is able to automatically adjust to illumination by changing parameters in the algorithm. When the target is lost, the algorithm can refind it quickly and continue tracking. The flaw of our algorithm lies in that the target must have strong texture feature, which means, the matching performance is not very good if the object is much smooth. Besides, a better method is needed to determine the lost more effectively. Therefore, our future work could focus on how to improve the lost judgement algorithm and add texture feature for the object to raise the accuracy of the system.

References

[1] G. R. Bradski, “Computer vision face tracking for use in a perceptual user interface,” Intel Technology Journal, 2nd Quarter, 1998.
[2] Gang Tian, Ruimin Hu, Zhongyuan Wang, Li Zhu, “Object Tracking Algorithm Based on Meanshift Algorithm Combining with Motion Vector analysis,” 2009 First International Workshop on Education Technology and Computer Science, pp.987-990, 2009.
[3] Yingying Yue, Yun Gao, Xuejie Zhang, “An Improved Camshift Algorithm Based on Dynamic Background,” The 1st International Conference on Information Science and Engineering (ICISE2009), pp.1141-1144, 2009.
[4] Xuena Qiu, Shirong Liu, Fei Liu, “Kernel-based Target Tracking with Multiple Features Fusion,” 48th IEEE Conference on Decision and Control and 28th Chinese Control Conference, pp.3112-3117, 2009.
[5] Huiyu Zhou , Yuan Yuan, Chunmei Shi, “Object tracking using SIFT features and mean shift,” Computer Vision and Image Understanding, pp.345-352, 2009.
[6] Xuena Qiu, Qiang Lu, “Target Tracking and Localization of Binocular Mobile Robot using Camshift and SIFT,” the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation, pp.483-488, 2009.
[7] Jieyu Zhang et al., “The Target Tracking Method Based on Camshift Algorithm Combined with SIFT,” Advanced Materials Research, Vol.186, pp.281-286, 2011.
[8] Gady Agam, “Introduction to programming with OpenCV,” http://www.cs.iit.edu/∼agam/cs512/lect-notes/opencv-intro/opencv-intro.html, January 27, 2006.
[9] J.G. Allen, R.Y.D. Xu, J.S. Jin, “Object Tracking Using CamShift Algorithm and Multiple Quantized Feature Spaces,” the Pan-Sydney area workshop on Visual information processing, pp.3-7, 2004.
[10] D.G. LOWE, “Distinctive Image Features from Scale-Invariant Keypoints, International Journal of Computer Vision,” vol.60, No.2, pp.91-110, 2004.
[11] H. Bay, T. Tuytelaars and L. Van Gool, “SURF: Speeded Up Robust Features,” Computer Vision–ECCV 2006, pp. 404-417, 2006.
[12] C. Evans, “Notes on OpenSURF library,” University of Bristol, Tech. Rep. CSTR-09-001, January 2009.