Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2016, Article ID 3835952, 8 pages
http://dx.doi.org/10.1155/2016/3835952

Research Article

Improved Adaptive Vibe and the Application for Segmentation of Complex Background

Le Chang,[1] Zhenghua Liu,[1,2] and Yan Ren[1,3]

[1]*School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China*
[2]*Science and Technology on Aircraft Control Laboratory, Beihang University, Beijing 100191, China*
[3]*School of Information Engineering, Inner Mongolia University of Science and Technology, Baotou 014010, China*

Correspondence should be addressed to Le Chang; cl_buaa@126.com

Received 12 January 2016; Revised 17 February 2016; Accepted 19 April 2016

Academic Editor: Daniel Zaldivar

To solve the problems that basic Vibe algorithm cannot effectively eliminate the influence of background noise, follower shadow, and ghost under complex background effectively, an adaptive threshold algorithm, AdaVibe, based on the framework of basic Vibe is proposed. Aiming at the shortage of the basic algorithm, this paper puts forward some improvement measures in threshold setting, shadow eliminating, and ghost suppression. Firstly, judgment threshold takes adjustment with the changes of background. Secondly, a fast eliminating ghost algorithm depending on adaptive threshold is introduced. Finally, follower shadow is detected and inhibited effectively through the gray properties and texture characteristics. Experiments show that the proposed AdaVibe algorithm works well in complex environment without affecting computing speed and has stronger robustness and better adaptability than the basic algorithm. Meanwhile, the ghost and follower shadow can be absorbed quickly as well. Therefore, the accuracy of target detection is effectively improved.

1. Introduction

As the basis of the research on target tracking and behavior identification, moving target detection is one of the most popular directions in the field of computer vision [1–4]. Moving target detection algorithms can be divided into the following categories, frame difference method [5], optical flow method [6], and background subtraction method, which can further be divided into the Gaussian mixture model background subtraction method [7, 8], codebook model background subtraction method [9], pixel-level Vibe algorithm [10–12], and so on. Among the methods mentioned above, frame difference method with less computational complexity is easy to design but tends to misjudge slow moving target. Optical flow method has high accuracy, but it is not suitable for real-time detection because of large computation. The Gaussian mixture model with the slow parameter estimation has difficulty achieving real-time performance requirements. The fixed threshold value of codebook model can easily lead the code to diverge. Vibe algorithm can extract the target in the initial few frames, with good adaptability and real-time performance.

However, the basic Vibe algorithm also has a series of problems in practice. (1) The threshold set up by experience is a fixed value and it has no adaptive adjustment during the running process of the algorithm, so the result is not robust enough. (2) It is easy to create ghost and the suppression of the ghost is slow [13]. (3) Vibe algorithm does not deal with the follower shadow and directly recognizes the shadow as a part of the target [10]. The detection accuracy is reduced due to follower shadow. To solve these problems, this paper puts forward an improved algorithm combining adaptive algorithm with Vibe algorithm. A new suppression strategy of ghost that depends on the Vibe mode is introduced. At the same time, follower shadow is effectively detected through the gray properties and texture characteristics.

2. Fundamental Principle of Basic Vibe Algorithm

The Visual Background Extractor (Vibe) algorithm was proposed by Barnich and Van Droogenbroeck of Belgium in

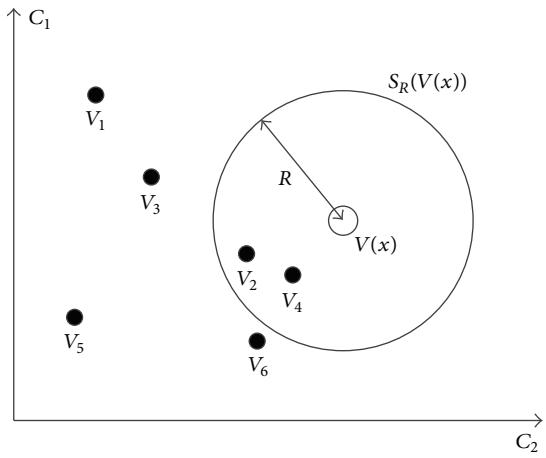

Figure 1: Vibe background model.

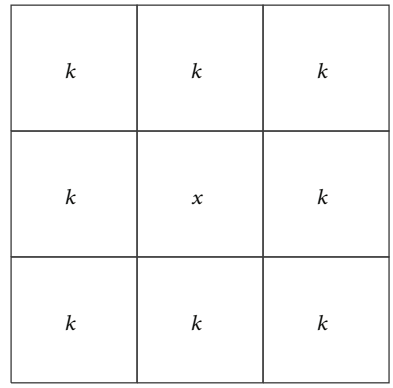

Figure 2: Eight-neighbor model.

2009 for quick background subtraction and the moving target detection [14].

Vibe algorithm uses the first frame to initialize the background model. It introduces the randomization into modeling for the first time and proposes a strategy to randomly update the background model. At the same time, according to the spatial consistency principle, each pixel of the background will be randomly updated into the neighbor model. The steps of the algorithm are pixel background modeling, model matching, and model updating.

As shown in Figure 1, each background pixel x is modeled by a collection of N background sample values. The background model formula is shown as

$$M(x) = \{V_1, V_2, V_3, \ldots, V_N\}. \quad (1)$$

All of the N samples have been judged as background pixels. As shown in Figure 2, these samples are randomly chosen from the eight pixels marked with k for N times.

In the next new frame, for the current pixel value $V(x)$ of x, set threshold R, and calculate the number of common samples of the $\{V(x) - R, V(x) + R\}$ and $M(x)$:

$$\#\{\{V(x) - R, V(x) + R\} \cap M(x)\}. \quad (2)$$

If the result is greater than value #min, x will be treated as a member of foreground. Otherwise, it will be treated as a member of background.

Usually, we need a certain length of video sequences to complete the initialization and build background model. However, Vibe algorithm only uses the first frame to initialize the background model. As one frame could not contain N samples of a specific pixel, according to the spatial consistency principle of neighbor pixels, we can fill the background model for N times with its neighbor pixels. These neighbor pixels are randomly chosen according to a uniform law:

$$M^0(x) = \left\{V^0\left(y \mid y \in N_G(x)\right)\right\}. \quad (3)$$

In formula (3), 0 represents the initial time, and $N_G(x)$ represents the eight-neighbor pixels. Vibe adopts a conservative model update policy. Foreground pixels will never be used to fill the background model. When $V(x)$ is determined as a background pixel, it has a probability of $1/\phi$ to update its own background model. At the same time, it also has the probability of $1/\phi$ to update the background model of its neighbor pixels, which are selected randomly from its eight-neighbor pixels.

The present time is t; in further time $(t + dt)$, the probability that a specific sample is still in the background model is equal to

$$P(t, t + dt) = e^{-\ln(N/(N-1))dt}. \quad (4)$$

3. Adaptive Threshold Strategy

In basic Vibe algorithm, model matching always uses a fixed global threshold R. Experiments show that the simple threshold policy ignores the complexity of the local environment and the uncertainty of changes. So it is difficult to detect the target effectively in complex environment.

For complex dynamic background, the threshold R should be increased appropriately, so that the background cannot be easily detected as foreground. On the other hand, for simple static background, R should be decreased to detect small changes of the foreground. In order to improve the robustness of basic Vibe algorithm, an improved method of adaptive threshold is proposed in this section.

Define a set of distances $D(x)$:

$$D(x) = \{D_1(x), \ldots, D_i(x), \ldots, D_N(x)\}. \quad (5)$$

In formula (5), $D_i(x)$ is the minimum Euclidean distance between current pixel $V(x)$ and its background samples $V_i(x)$ and N represents the number of samples. In this paper, N is set to be 20. The average value $d_{\text{mean}}(x)$ is used to characterize the complex situation of background change:

$$d_{\text{mean}}(x) = \sum_{i=1}^{N} D_i(x). \quad (6)$$

A new $d_{\text{mean}}(x)$ is recorded after each successful match between $V(x)$ and $V_i(x)$. For static background, $d_{\text{mean}}(x)$ tends to be steady, while $d_{\text{mean}}(x)$ increases gradually for

complex dynamic background. So, the threshold R is updated adaptively according to the value of $d_{\text{mean}}(x)$:

$$R = \begin{cases} R * (1 + \varepsilon_c), & \text{if } R < d_{\text{mean}}(x) * \delta, \\ R * (1 - \varepsilon_d), & \text{else.} \end{cases} \tag{7}$$

In formula (7), ε_c, ε_d, and δ are the fixed parameters; set $\varepsilon_c = 0.06$, $\varepsilon_d = 0.4$, and $\delta = 5$. It can be shown that R tends to be steady for static background. When the interference generated by dynamic background appears, R increases gradually to improve adaptability of the algorithm.

The upper and lower limits are set to prevent R from becoming too large or too small:

$$\begin{aligned} R_{\min} &= 10, \\ R_{\max} &= 30. \end{aligned} \tag{8}$$

4. The Adaptive Algorithm of Ghost Suppression

Different from other algorithms, Vibe algorithm initializes the background model from the first frame of the video, so the initialization is very quick. The drawback is that the presence of a moving object in the first frame will introduce an artifact commonly called a ghost. While initializing model, the moving target of the foreground in the first frame is considered as background pixels incorrectly. When the moving target in the following frames leaves the original position, the sampled real background grayscale values cannot match the initial background model. As a result, the background pixels are considered as foreground by mistake, and the ghost appears. Although in following frames the ghost can be eliminated by refreshing background models, this process is still relatively slow. Meanwhile, if the moving target goes through the ghost area, the detection accuracy will be decreased.

Considering the existence of the ghost, this paper deals with the detected foreground pixels according to properties of ghost to eliminate the ghost quickly.

4.1. Fundamental Idea. It can be seen from above that ghost pixels are background pixels, which are detected as foreground pixels incorrectly. According to the consistency principle of neighbor space, the grayscale values of ghost area are close to its neighbor pixels. So, a pixel can be treated as the ghost if this pixel and the background model of its neighbor pixels can match well. Usually, when we deal with a frame, we scan the frame from left to right and from top to bottom. As shown in Figure 3, following the left-right-top-bottom order, x stands for the suspicious ghost pixel to be judged, pixels marked with k are the background or treated ghost pixels, and pixels marked with u are the pixels to be dealt with.

When dealing with ghost, as the pixels marked with k_i have been scanned and dealt with, the background models of these four pixels are reliable. If pixel x is detected as a ghost pixel, update the background model of x in time. So background model of x is reliable and the ghost can be eliminated from outside to inside gradually to achieve the goal of quickly eliminating ghost. When updating the background model of x, we only choose its value from background models of four reliable neighbor pixels.

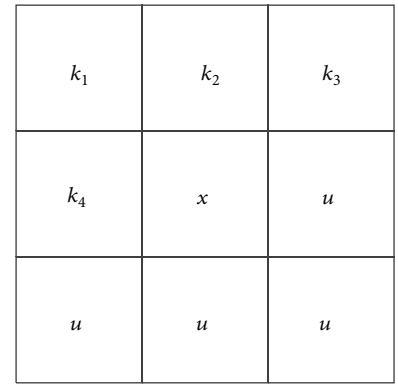

Figure 3: Ghost processing pattern.

4.2. Detailed Process. Steps to deal with ghost are listed below.

(1) Calculate the Euclidean distances in RGB space between x and pixels $\{I_1, I_2, I_3, I_4\}$ in positions $\{k_1, k_2, k_3, k_4\}$. Keep the distances as $\{d_1, d_2, d_3, d_4\}$, and get the minimum value of the four distances:

$$\begin{aligned} d_i &= \sqrt{(r_i - r_x)^2 + (g_i - g_x)^2 + (b_i - b_x)^2}, \\ d_m &= \min(d_1, d_2, d_3, d_4). \end{aligned} \tag{9}$$

(2) Using the corresponding pixel of d_m as reference pixel (i.e., $d_m = d_1$), calculate distances between the other three pixels and the reference. Find the largest distance D_m among the three distances following formula (10), and get the threshold by formula (11):

$$D_m = \max(D_{1,2}, D_{1,3}, D_{1,4}) \tag{10}$$

$$\text{th} = \min(D_m, D). \tag{11}$$

In formula (10), $D_{1,x}$ represents the distance between pixel on k_1 and pixels on $\{k_2, k_3, k_4\}$, respectively. It is to be noted that D in formula (11) is a fixed value to prevent the incidental fluctuations of threshold caused by occasional disturbances, and it is set to be 25 in this paper.

(3) Use the threshold th to judge ghost pixels. When $d_m < \text{th}$, the pixel x can easily match the background model of its neighbor pixels, so x is considered to be a ghost pixel. Classify the pixel x as background, randomly sample the pixel again, and establish a new background model. The background models of its neighbor pixels are updated at the same time. Otherwise, the pixel x belongs to foreground target indeed.

To reduce the computation cost, not all detected foreground pixels are judged whether they are ghost using the

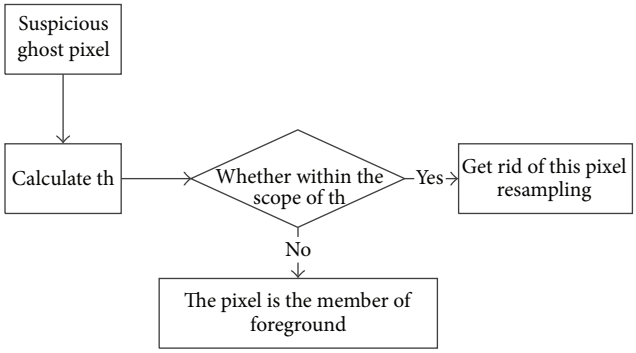

Figure 4: Flow chart of ghost suppression.

method above. Instead, a kind of classification strategy is used to filter the pixels before judging. In this paper, we count the times, N_g, that a pixel is continuously detected to be a foreground pixel. A pixel is determined to be a suspicious ghost pixel only when N_g exceeds a specific range. Then, take the method above to judge it again.

Figure 4 shows the flow chart of ghost suppression.

5. The Algorithm of Eliminating Shadow Based on Texture Characteristics

In general, shadow usually does not affect detection results. However, the follower shadow, which accompanies the target, is recognized as a part of the target by most algorithms. Basic Vibe algorithm does not take account of the follower shadow. That is to say, the follower shadow may be mixed into the detected foreground, which affects the result. To enhance the accuracy of the detected result, a new method of follower shadow eliminating is proposed.

Due to the characteristics of shadow, when a pixel is covered by shadow, its grayscale value is linear with that when not covered:

$$p(i,j) = \frac{B(i,j)}{O(i,j)} \quad (0.2 < p < 0.8). \tag{12}$$

According to the theory above, if the relationship between grayscale value of a foreground pixel and its original grayscale value satisfies formula (12), the pixel could locate in shadow area. In formula (12), $O(i, j)$ is the original grayscale value and $B(i, j)$ is the grayscale value when a pixel is covered by shadow. The whole formula is named proportion constraint formula.

Further experiments show that eliminating the shadow only with grayscale value may exaggerate the shadow area. That is, some parts of the target are also recognized as shadow. To solve this drawback, SILTP texture characteristics operator is used as additional constraint condition. The research shows that SILTP characteristics operator of a pixel cannot be changed a lot when covered by shadow.

Suppose the pixel location of a graph is (x_c, y_c); SILTP encoding is shown as

$$\mathrm{SILRP}^{\tau}_{N,R}(x_c, y_c) = \bigoplus_{k=0}^{N-1} s_\tau(I_c, I_k). \tag{13}$$

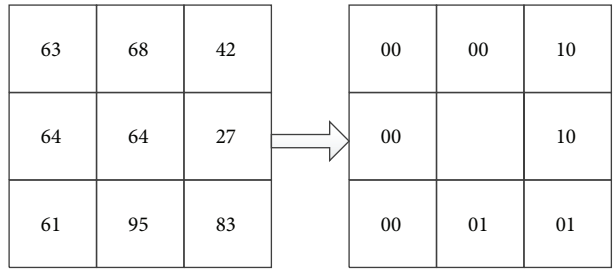

Figure 5: SILTP encoding.

In the formula, I_c is the grayscale value of the region center and I_k are the grayscale values of pixels in the N neighbor area whose radius is R. $\bigoplus$ means connecting all binary values to strings. τ is the variable threshold range. Piecewise function of $s_\tau(I_c, I_k)$ is shown as

$$s_\tau(I_c, I_k) = \begin{cases} 01, & I_k > (1+\tau) I_c, \\ 10, & I_k < (1-\tau) I_c, \\ 00, & \text{other}. \end{cases} \tag{14}$$

In this paper, region R is taken as the eight-neighbor area of the center pixel, and τ is taken as 0.1. Figure 5 is an example of SILTP encoding.

Take out codes clockwise, and then the eight binary values generated by four 2-bit binary values of four directions (top, left, bottom, and right) can be recognized as the characteristics operator of this window's center pixel. The eight binary values are also the texture information of this window's center pixel. In Figure 5, the SILTP characteristics operator is 00100100. SILTP characteristics operator has stronger robustness against brightness changes. In addition, the weak shadow area can also be detected well:

$$N_s = \mathrm{num}\left(\overrightarrow{o\mathrm{SILTP}} - \overrightarrow{b\mathrm{SILTP}}\right). \tag{15}$$

Take the foreground pixel which satisfies formula (12) as the foreground pixel to be detected. In formula (15), $\overrightarrow{o\mathrm{SILTP}}$ stands for the original characteristics operator of the pixel, and $\overrightarrow{b\mathrm{SILTP}}$ is the updated SILTP characteristic operator of the pixel. N_s is the number of positions whose $\overrightarrow{o\mathrm{SILTP}}$ and $\overrightarrow{b\mathrm{SILTP}}$ values are different among the eight positions:

$$N_s < N_{\max}. \tag{16}$$

Compare N_s and $N_{\max}$; if formula (16) works, the pixel locates in follower shadow area. Set $N_{\max} = 5$ in this paper.

In summary, the whole process of eliminating shadow is shown as below.

(1) Calculate initial characteristic operator $\overrightarrow{o\mathrm{SILTP}}$.

(2) Get foreground pixels by AdaVibe algorithm.

(3) Test grayscale proportion on foreground pixels.

(4) Test if the proportion satisfies formula (12). If it satisfies it, go to Step (5). Otherwise, the pixel is foreground.

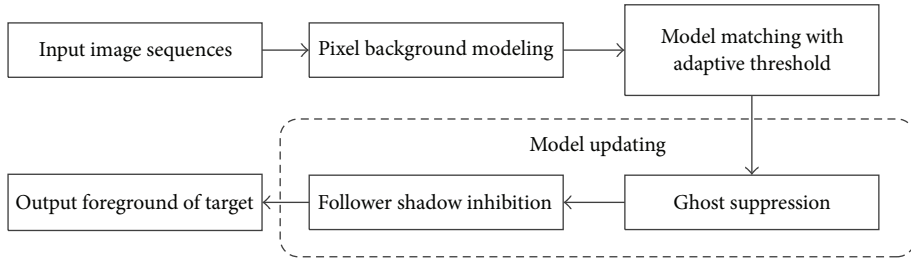

Figure 6: Flow chart of adaptive Vibe algorithm.

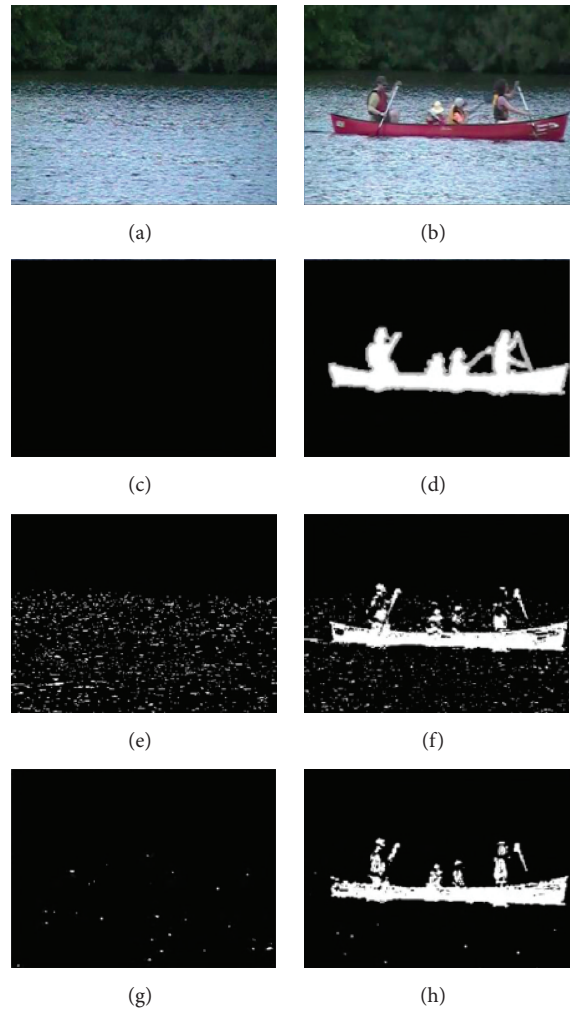

Figure 7: The effect figure by using Vibe and added adaptive R.

(5) Calculate $\overrightarrow{b\text{SILTP}}$ and N_s.

(6) If N_s satisfies formula (16), the pixel is shadow and can be removed from the foreground. Otherwise, it is foreground.

If the video background varies greatly, to ensure the robustness of the algorithm, SILTP characteristic operators of background pixels should be refreshed every T cycles to create new $\overrightarrow{o\text{SILTP}}$. Usually, the foreground only occupies a small part of the whole area. Consequently, adding SILTP characteristic operators does not affect the speed of the algorithm.

Above all, the flow chart of adaptive Vibe algorithm proposed in this paper is shown in Figure 6.

6. Simulations and Analyses

The experiment was done on a Lenovo PC, with Windows XP and Intel Celeron E3400 2.60 GHz CPU. The resolution of the video is $320 * 240$, and frame rate is 24 pfs, #min = 3, $\phi = 16$.

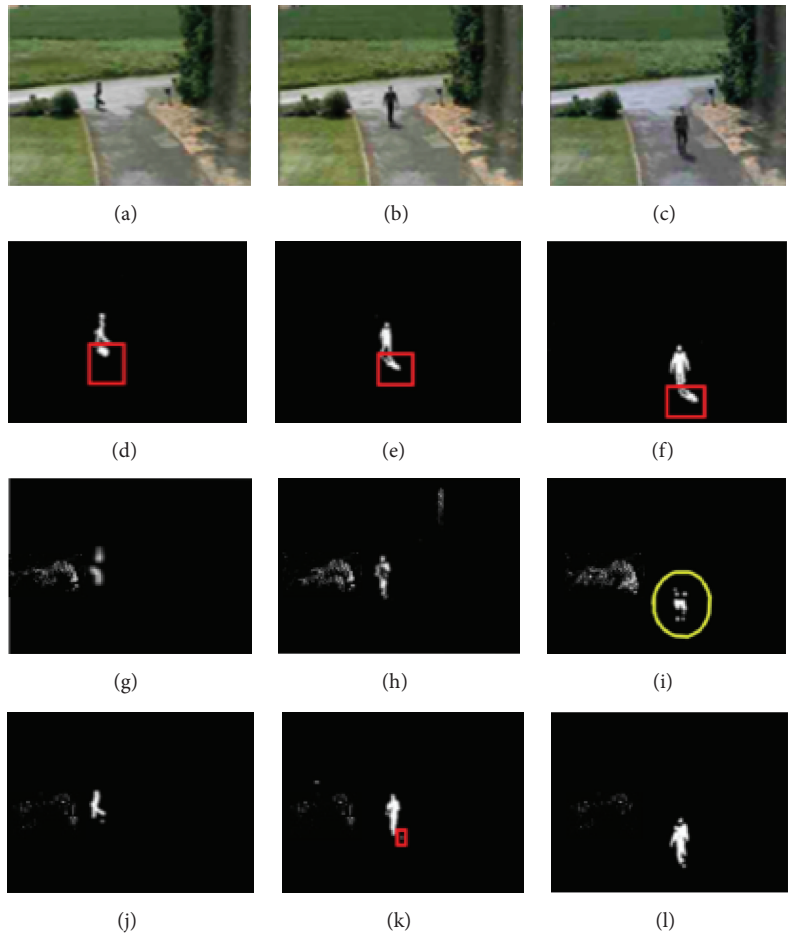

Figure 8: Effect figure of shadow eliminating.

Table 1: Performance of the algorithms.

	Calculation speed/frames	Error rate/%
Vibe (RGB)	20.24	18.3
AdaVibe	19.68	11.2

Calculation speed: average number of frames treated per second.
Error rate: percentage of error against ground truth.
Vibe (RGB): Vibe algorithm in RGB space.
AdaVibe: AdaVibe algorithm proposed in this paper.

Table 2: Comparison of shadow eliminating.

	Calculation speed/frames	Accuracy rate/%
Vibe (RGB)	19.56	N/A
SEGP	19.07	68.4
AdaVibe	18.98	94.5

Accuracy rate: percentage of correct shadow pixels detected by algorithm against practical shadow pixels.

6.1. Test of Adaptive Threshold. Figure 7 is the effect diagram of the first test video clip. Left to right are frames 20 and 968, and top to bottom are original graph, ground-truth, Vibe (RGB), and AdaVibe proposed in this paper. In this video, the interference is water fluctuations.

Comparison between the two algorithms is shown as Table 1.

It can be seen that, with the adaptive threshold adjustment strategy, the influence of water ripple on the foreground detection is reduced. AdaVibe algorithm is better suitable for the dynamic background. So AdaVibe has stronger robustness and better adaptability than the basic algorithm. Besides, AdaVibe has the same speed as Vibe (RGB).

6.2. Test of Follower Shadow Elimination Added Adaptive Threshold. Figure 8 is the treated result of the second video clip. In this video, there is interference such as overlaps, vibrations of leaves, and light changes. Besides, there is a shadow accompanying the moving target. Left to right are frames 132, 296, and 333, and top to bottom are original graph, ground-truth, the shadow eliminating algorithm based on grayscale properties (SEGP), and AdaVibe. Area inside the red rectangle is the follower shadow.

Comparison of shadow eliminating among these algorithms is shown as in Table 2.

From Table 2, AdaVibe gains a good result. It can suppress the follower shadow, without lowering algorithm speed and oversuppression (marked by yellow circle in Figure 8).

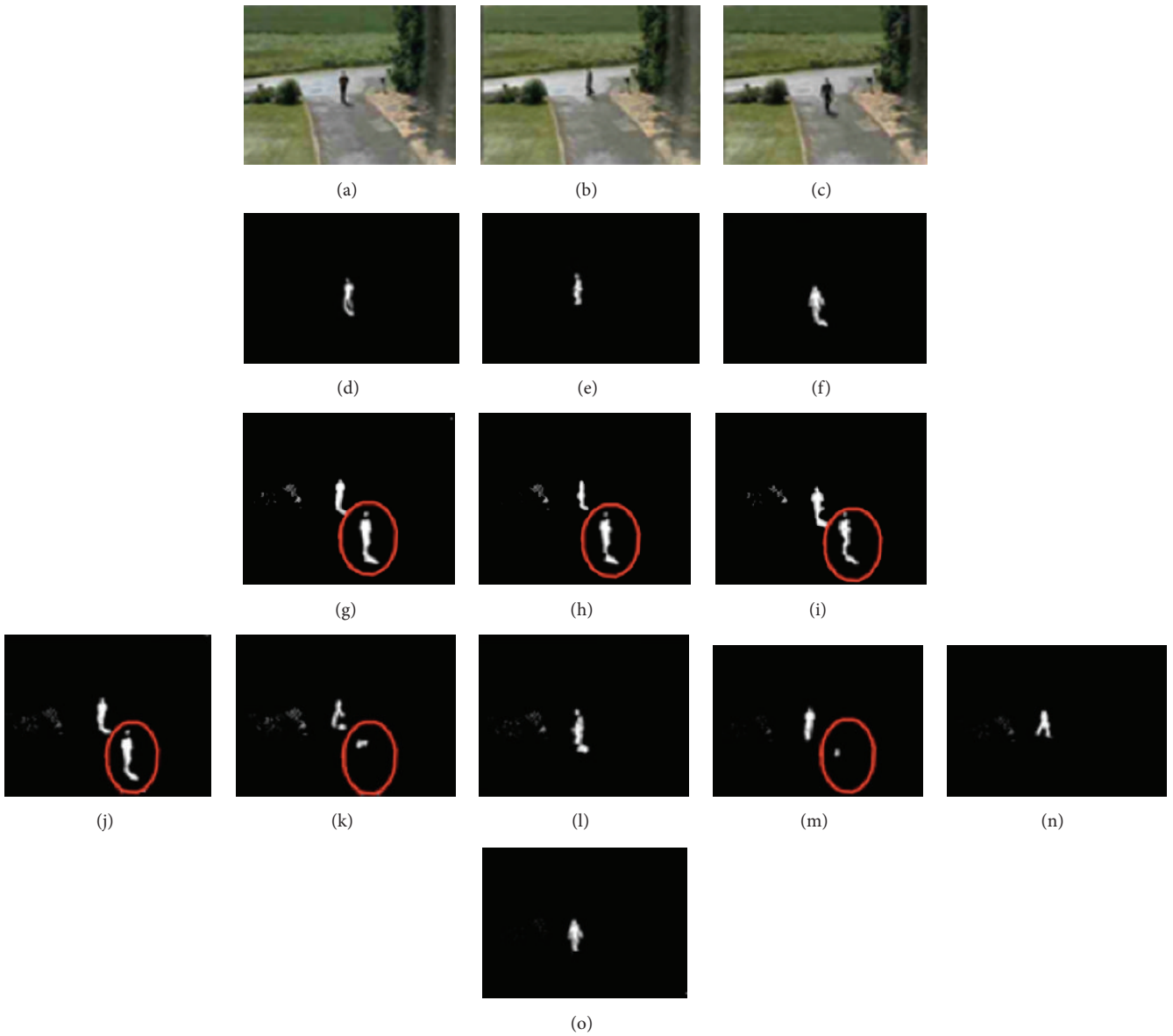

Figure 9: Effect figure of ghost and follower shadow suppression.

6.3. Test of Ghost and Follower Shadow Elimination Added Adaptive Threshold. In the third tested video, the moving target emerges from the first frame and there is follower shadow accompanying the target. That means ghost may exist in the basic Vibe algorithm. Apply different algorithms to the video, and compare the results.

Figure 9 is the effect diagram of ghost and follower shadow suppression. Left to right are frames 22, 117, and 303, and top to bottom are original graph, ground-truth, Vibe (RGB), algorithm proposed in [11], and AdaVibe. Area surrounded by red circle is ghost. The comparison of ghost suppression among three methods motioned above is listed in Table 3. Figure 10 shows the error rate of each frame.

From the test result, AdaVibe algorithm can eliminate ghost in less frames, which improves the computing speed. What is more, elimination of ghost and shadow also improves the accuracy and robustness greatly.

7. Conclusion

To solve the drawbacks in basic Vibe algorithm, the adaptive threshold is proposed in this paper to cover the shortage of fixed threshold, reduce the interference with test results, and then take actions to eliminate ghost. Finally, grayscale and texture information are introduced to suppress follower shadow effectively.

The results show that AdaVibe algorithm can effectively eliminate background noise, follower shadow, and ghost in complex background and improve the accuracy and robustness as well as stableness without affecting the computing speed.

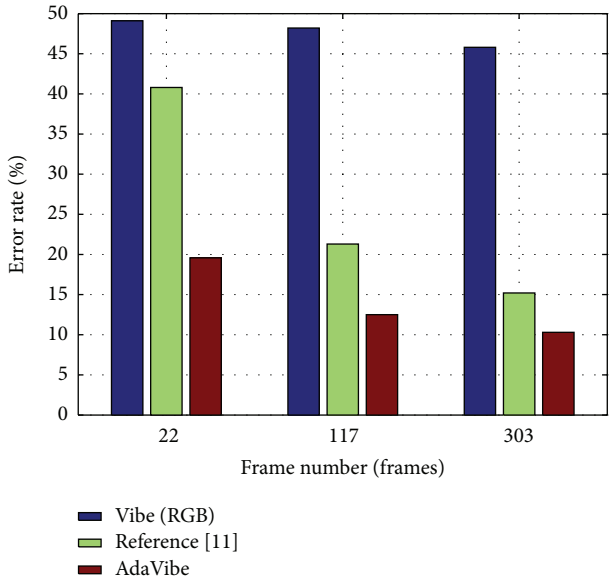

Figure 10: The error rate of each frame.

Table 3: Comparison of ghost inhibition.

	Frames needed for ghost suppression	Calculation speed/frames
Vibe (RGB)	763	19.58
Algorithm of [11]	125	18.60
AdaVibe	28	18.83

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments and thank the strong support provided by the National Natural Science Foundation of China (61563041), the National Natural Science Foundation of Inner Mongolia (2015MS0603), and Aviation Science Foundation Project (20135751040).

References

[1] L. Zhang, F. Zhizhong, and Z. Yueping, "Moving objects detection based on HSV colorspace and Vibe algorithm," *Computer Engineering and Applications*, vol. 50, no. 4, pp. 181–185, 2014.

[2] C. R. Jung, "Efficient background subtraction and shadow removal for monochromatic video sequences," *IEEE Transactions on Multimedia*, vol. 11, no. 3, pp. 571–577, 2009.

[3] S. Cohen, "Background estimation as a labeling problem," in *Proceedings of the 10th IEEE International Conference on Computer Vision (ICCV '05)*, vol. 2, pp. 1034–1041, Beijing, China, October 2005.

[4] R. G. Abbott and L. R. Williams, "Multiple target tracking with lazy background subtraction and connected components analysis," *Machine Vision and Applications*, vol. 20, no. 2, pp. 93–101, 2009.

[5] J. Zhao, *The Research of Moving Target Detection Method Based on Three-frame Difference*, Xidian University, Xi'an, China, 2013.

[6] B. K. P. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, no. 1–3, pp. 185–203, 1981.

[7] Z. Zivkovic, "Improved adaptive Gausian mixture model for background subtraction," in *Proceedings of the IEEE International Conference on Pattern Recognition (ICPR '04)*, vol. 2, pp. 28–31, Cambridge, UK, August 2004.

[8] D.-S. Lee, "Effective Gaussian mixture learning for video background subtraction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 5, pp. 827–832, 2005.

[9] C. Chen, X. Chen, and Z. Fan, "Detection algorithm based on block mode and codebook model," *Journal of China University of Metrology*, vol. 23, no. 2, pp. 125–130, 2012.

[10] O. Barnich and M. Van Droogenbroeck, "ViBe: a universal background subtraction algorithm for video sequences," *IEEE Transactions on Image Processing*, vol. 20, no. 6, pp. 1709–1724, 2011.

[11] C. Liang, X. Chen, and Z. Fan, "Ghost suppression algorithm based on Vibe," *Journal of China University of Metrology*, vol. 24, no. 4, pp. 425–429, 2013.

[12] M. Van Droogenbroeck and O. Paquot, "Background subtraction: experiments and improvements for ViBe," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 32–37, IEEE, Providence, RI, USA, June 2012.

[13] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati, "Detecting moving objects, ghosts, and shadows in video streams," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 10, pp. 1337–1342, 2003.

[14] O. Barnich and M. Van Droogenbroeck, "ViBe: a powerful random technique to estimate the background in video sequences," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '09)*, pp. 945–948, Taipei, Taiwan, April 2009.