FOREGROUND DETECTION IN MULTI-CAMERA SURVEILLANCE SYSTEM

[1]Ming-Yi Lin (林明毅), [2]Tien-Hsu Lee (李天序), and [1]Pao-Chi Chang (張寶基)

[1]Dept. of Communication Engineering, National Central University
[2]Dept. of Electrical Engineering, National Chi Nan University

ABSTRACT

In multi-camera surveillance system, the importance of each camera differs from each other and needs to be identified. In this paper, we develop an Edge-based Foreground Block Detection (EFBD) method to find out changing (foreground) blocks and then determine the importance of cameras based on EFBD. We also use H.264 codec to develop a multi-camera surveillance system which provides functions of auto alarm, dynamic recording, and foreground detection, and makes important cameras obtain better visual quality. The experimental results demonstrate that the proposed scheme can extract the foreground blocks efficiently and can handle the variations of light conditions. The detected foreground blocks can provide clues to better rate allocation and coding efficiency in our implemented multi-camera surveillance system.

Keywords: edge detection, foreground block detection, region fill-in, H.264 video coding, multi-camera surveillance system.

1. INTRODUCTION

In multi-camera surveillance system, compared with other inactive cameras when one of cameras changes, the surveillance will focus on the changing camera. In other words, at the same time the importance of each camera does not necessarily equal to each other. Therefore, we hope that the important cameras can obtain higher video quality. We determine the important cameras based on the number of changing blocks. Also, in order to automatically identify the important cameras, we develop a foreground block detection method to detect the number of foreground blocks. Furthermore, we make use of the rate allocation method so that the important cameras can obtain higher video quality. And the detection method can be used to record dynamically and automatically for saving the storing space.

At present, most detection methods use either temporal or spatial information existed in the image sequence. Several conventional approaches for detection are outlined as follows [1].

1. Background subtraction: Background subtraction is the most common method for real-time segmentation of moving regions in image sequences. A simple and common background subtraction uses several seconds of frames to model each pixel of the background with a normal distribution. Then subtract the current image from the background image and pass the resulting difference image through the threshold to detect foreground pixels. Many systems use this method to detect pixels belonging to moving objects [2]-[10].
2. Temporal differencing: Temporal differencing makes use of the pixel-wise differences between two or three consecutive frames in an image sequence to extract moving regions. Temporal differencing is quite adaptive to dynamic environments, but generally does a poor job of extracting all the relevant pixels, e.g., there may be holes left inside moving entities. As an example of this method, Lipton et al. [11] detect moving targets in real video streams using temporal differencing. After the absolute difference between the current and the previous frame is obtained, a threshold function is used to determine changes. By using a connected component analysis, the extracted moving sections are clustered into motion regions. An improved version uses three-frame instead of two-frame differencing [12].
3. Optical flow: Optical-flow-based motion detection uses characteristics of flow vectors of moving objects over time to detect moving regions in an image sequence. For example, Meyer et al. [13] compute the displacement vector field to initialize a contour based tracking algorithm, called active rays, for the extraction of articulated objects. The results are used for gait analysis. Optical-flow-based methods can be used to detect independently moving objects even in the presence of camera motion. However, most flow computation methods are computationally complex and very sensitive to noise, and cannot be applied to video streams in real time without specialized hardware. More detailed discussion of optical flow can be found in Barron's work [14].

2. EDGE-BASED FOREGROUND BLOCK DETECTION (EFBD)

In order to coordinate the surveillance system, the proposed foreground block detection algorithm should conform to the following requirements:

1. In the surveillance system, any changes should be detected and recorded, such as caused by the shadow, human, vehicle, and so on.
2. When the moving object keeps static for a long time, it should be identified as the background.
3. In our surveillance system, the video codec uses H.264. Therefore, we need a relatively low complexity detection algorithm for real time implementation.
4. In order to match up H.264, we use the 4x4 block size as the basic unit of the developed foreground block detection.

2.1. Foreground edge map extraction

We use the edge-based background subtraction method for extracting foreground edges. While comparing with the temporal difference method, the background subtraction method would not be affected significantly by the frame rate. For example, when the frame rate is low, using the temporal difference method would result in appearing ghost. On the contrary, when the frame rate is high and objects move slowly, adopting the temporal difference method would lead to broken foreground objects. However, the background subtraction method is sensitive to the light variations. Therefore, we combine the concept of edge detection in order to eliminate the influences of light.

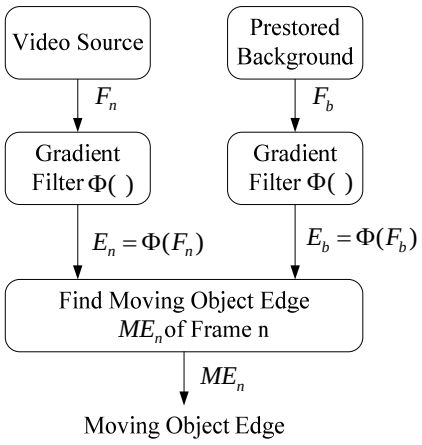

Fig. 1 Block diagram of the foreground edge map extraction.

Fig. 1 shows the block diagram of the foreground edge map extraction. After calculating the edge map difference of images using a Sobel edge detector, we extract the moving object edge ME_n of the current frame F_b based on the current frame's edge map $E_n = \Phi(F_n)$ and the background frame's edge map $E_b = \Phi(F_b)$. Here all operations are performed in the luminance component. We define the edge model $E_n = \{e_1, \cdots, e_k\}$ and $E_b = \{e_1, \cdots, e_{k_b}\}$ as the set of all edge points detected by the Sobel operator in the current frame F_n and background frame F_b, where $k \le w \times h$ and $k_b \le w \times h$, w is the image width, h is the image height. Similarly, we denote $ME_n = \{m_1, \cdots, m_l\}$ as the set of l moving object edge points, where $l \le k$ and $ME_n \subseteq E_n$.

$$ME_n(i,j) = \{e(i,j) \in E_n \mid \sum_{x=i-1}^{i+1} \sum_{y=j-1}^{j+1} |E_b(x,y) - E_n(x,y)| \ge TH_{foreground}\} \quad (1)$$

Where (i, j) represents a pixel location, $TH_{foreground}$ is threshold of the foreground edge points. The additional errors caused by the Sobel edge detector can be reduced by appropriately setting the threshold The edge points in ME_n are not restricted to the moving object's boundary, and can be in the interior of object boundary, as shown in Fig. 2.

Fig. 2 Edge maps resulting from (1).

2.2. Foreground block extraction

After extracting the edges of foreground objects, we should find out their corresponding regions. The common methods are 4-connected and 8-connected region filling. We hope that the system can handle twenty frames per second at least, but 4-connected and 8-connected region filling are too complicated to fit our system. On the other hand, in the video object segmentation, the Fill-in method [15], [16] is used to extract objects because it is efficient and low-complicated. But the method is mainly aimed at extracting one or two objects in a frame. In other words, multiple objects can not be applied. Therefore, we serve the Fill-in method as the basis to develop the proposed foreground block extraction algorithm.

The foreground block extraction includes three stages: The first step is to treat the edge pixel as the basis for the row filling. The second step is to transform the pixel-based frame into the 4x4 block-based frame. The third step is based on the results obtained from the second step and uses the 4x4 block-based frame for the column filling. The final results are foreground blocks. The flow chart is shown in Fig. 3, and the procedures are described as follows.

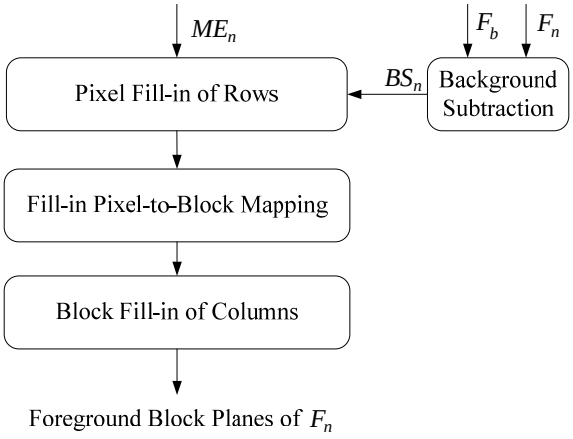

Fig. 3 Block diagram of the foreground block extraction.

1. We regard N_{Row} pixels as the unit for the fill-in of row direction. First of all, we find out the first edge pixel in ME_n and serve its coordinate as the start point for fill-in. And then from the start point we find out the coordinate of the last edge pixel or the coordinate conforming to (2) within N_{Row} pixels in the same row for identifying the stop point. Subsequently, we set the pixels from the start to the stop as foreground pixels, find out the next edge pixel as the start point for fill-in and also repeat the above procedures for all rows. TH_{Row} is a threshold for determining foreground pixels.

$$\begin{aligned} &\left|F_b[y][x-1] - F_n[y][x-1]\right| > TH_{Row} \cap \\ &\left|F_b[y][x] - F_n[y][x]\right| > TH_{Row} \cap \\ &\left|F_b[y][x+1] - F_n[y][x+1]\right| > TH_{Row} \end{aligned} \quad (2)$$

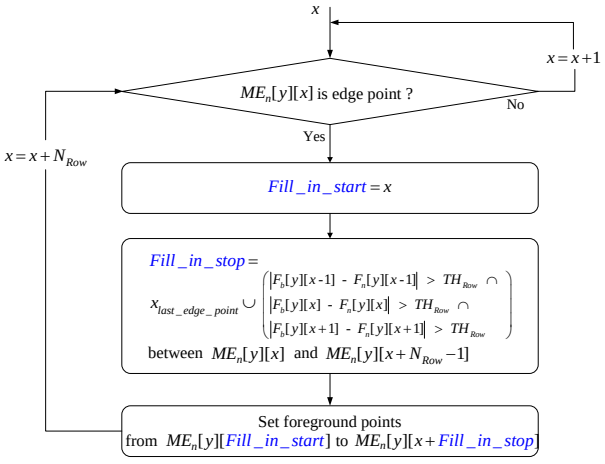

Fig. 4 Pixel fill-in of rows.

2. Calculate the number of foreground pixels in each 4x4 block. If the number is larger than TH_{Col} , set the block as a foreground block, where TH_{Col} is a threshold for determining foreground blocks.

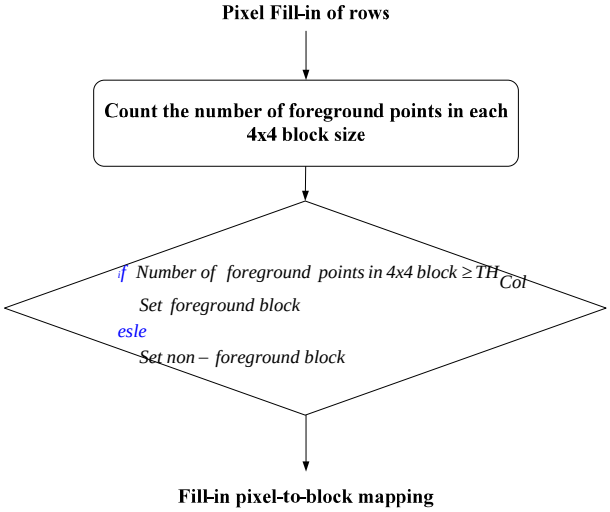

Fig. 5 Fill-in pixel-to-block mapping.

3. We use N_{Col} blocks as the unit for the fill-in of column direction. First of all, we find out the first foreground block obtained from the results of previous step and serve it as the start point for fill-in. And then from the start point we find out the last foreground block within N_{Col} blocks in the same column as the stop point (block). Subsequently, we set the blocks from the start to the stop as foreground blocks, find out the next foreground block as the start point for fill-in and also repeat the above procedures for all columns. Finally, we complete the extraction of foreground blocks.

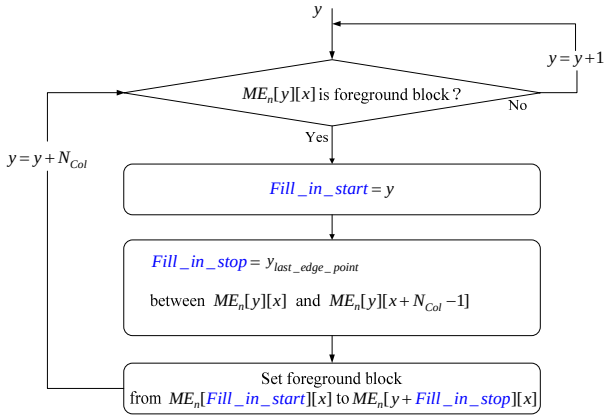

Fig. 6 Block fill-in of columns.

2.3. Background model update

We assume that the background frame model can be established by capturing a series of static images at the very start. However, the background subtraction method is sensitive to dynamic changes, such as accidents or light. Therefore, when the background changes with the passing of time, we need to update the background frame model continually to reduce errors.

As the proposed EFBD algorithm finds out the foreground blocks, if a certain foreground block keeps still in the same location for a long time, then the block should belong to the background model. For example, when a car is driven into a parking space and keeps still for a long time, then the car should be re-classified into the background frame. Under the long-time surveillance, with the passing of time the background frame should be also updated in order to maintain the accuracy of the foreground detection. Due to we adopt the block-based detection, we develop a block-based method to update our background model as well. At first, we classify blocks as follows:

- *FB* (foreground block): The foreground blocks obtained by applying EFBD.
- *NFB* (Non-foreground block): The blocks do not belong to the foreground blocks.
- *SFB* (Stationary *FB*) - The foreground blocks keep motionless in the same location.
- *SNFB* (Stationary *NFB*) - The non-foreground blocks keep changeless in the same location.

The background model needs to be updated for two kinds of blocks. They are *SFB* and *NFB* but not *SNFB*, respectively. In the first case, we define $FB_{t\times f_s}$ as the set of all foreground blocks detected by our algorithm in the $t\times f_s$ th frame, where t denotes the t th second, f_s denotes the source frame rate, and $NFB_{t\times f_s} \notin FB_{t\times f_s}$. $SFB_{t\times f_s}$ is defined as the set of k stationary foreground blocks, where $k \le l$ and $SFB_{t\times f_s} \subseteq FB_{t\times f_s}$, as follows:

$$SFB_{t\times f_s} = FB_{t\times f_s} \cap FB_{(t-1)\times f_s} \cap \cdots \cap FB_{(t-T_{FB})\times f_s} \quad (3)$$

where $T_{FB}\,(sec)$ denotes the foreground block's stay time. The pixels belong to $SFB_{t\times f_s}$ in the current frame F_n will be copied into the corresponding pixels of background frame F_b.

In the second case, we define $SNFB_{t\times f_s}$ as the set of k_b stationary NFBs in the $t\times f_s$ th frame, where $SNFB_{t\times f_s} \subseteq NFB_{t\times f_s}$, as follows:

$$SNFB_{t\times f_s} = NFB_{t\times f_s} \cap NFB_{(t-1)\times f_s} \cap \cdots \cap NFB_{(t-T_{NFB})\times f_s} \quad (3)$$

where $T_{NFB}\,(sec)$ denotes the non-foreground block's stay time. The pixels do not belong to *SNFB* in the current *NFB* frame will be copied into the corresponding pixels of background frame. Form the above two steps, we can dynamically update background model that can be adapted to still foreground objects and background light changes.

3. DEVELOPED MULTI-CAMERA SURVEILLANCE SYSTEM

The surveillance system includes multiple PC Cameras and a Central Location. PC Camera is used for detecting foreground blocks and encoding captured scenes. Central Location is used for receiving encoded videos and sending control messages to each camera. The system architecture shows in Fig. 7 and Fig. 8 and has the following characteristics:

1. Monitor more than three cameras simultaneously.
2. Provide high video quality and dynamic or full-time recording.
3. Achieve more than 20 fps processing speed in one camera.
4. Set the compression quality by oneself.
5. Monitor the video immediately through the network.
6. Provide different video quality based on the camera importance.
7. View the flow rate of each camera.
8. Support Intel Hyper-threading technology.

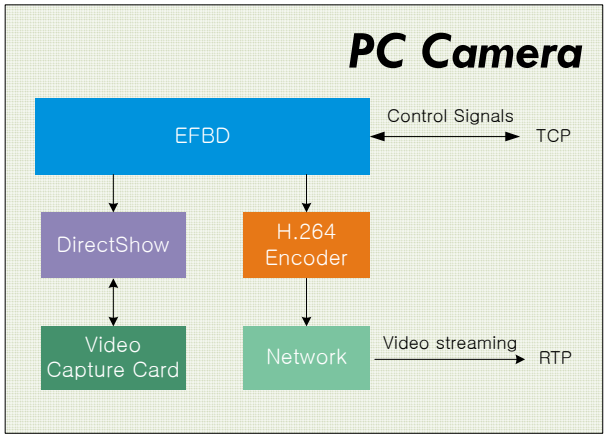

Fig. 7 Architecture of PC Camera.

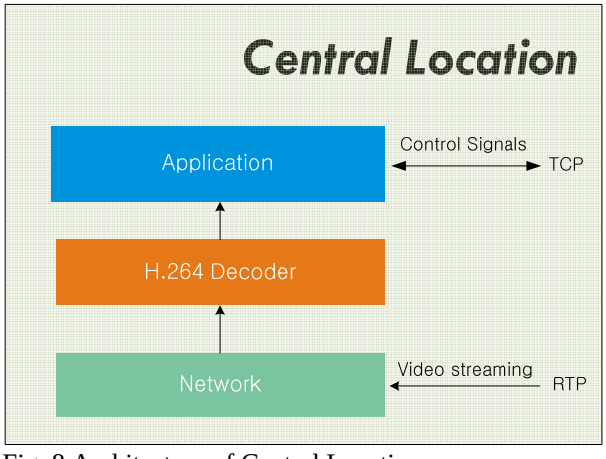

Fig. 8 Architecture of Central Location.

3.1. PC Camera

1. Capture different video sources:

The system uses DirectShow to connect with different video capture devices, as illustrated in Fig. 9. As long as the device specification conforms to WDM and VFW, the program can capture videos successfully.

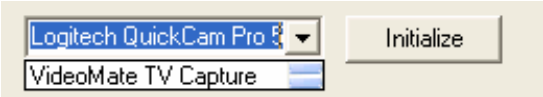

Fig. 9 Functions of capture device.

2. Provide the function of EFBD:

Under surveillance, Central Location can control the detection size of EFBD and the background model update period for PC Cameras, as shown in Fig. 10. The surveillance system can also deliver the alarm and even send the e-mail to inform remote what is happening and start recording automatically.

Fig. 10 Functions of EFBD.

3.2 Central Location

1. Provide dynamic or full-time recording:

According to users' preferences, they can adjust recording types for saving storage space. At present, we provide three types of recording: Motion, Detect, and Every Frame. Motion represents only to store frames with non-zero motion vectors. Detect means only to store frames in which foreground blocks are detected out. Every Frame is to store all captured frames. Fig. 11 shows above recording functions and the user interface.

Fig. 11 Functions of recording.

2. Monitor the flow rate of each camera:

We provide the flow rate of each camera in order to make users view the rate clearly. The target rate of each camera is dynamically allocated and controlled by Central Location for different visual quality. Fig. 12 shows an example.

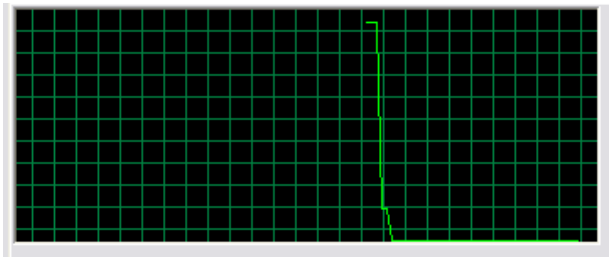

Fig. 12 Functions of monitoring flow rate.

4. SIMULATION RESULTS

This paper adopts the sequence *Hall* for the simulation and the results are described as follows.

Fig. 13 Original frame 35 and resulting EFBD.

Fig. 14 Original frame 120 and resulting EFBD.

Fig. 15 Original frame 160 and resulting EFBD.

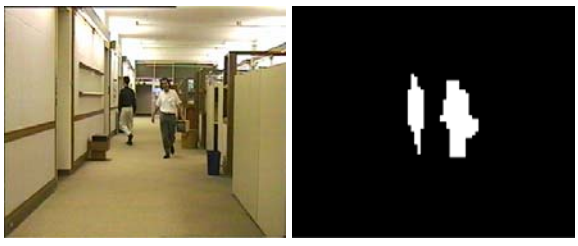

Fig. 16 Original frame 220 and resulting EFBD.

From Fig. 13 to Fig. 15, we observe that EFBD can extract the foreground blocks effectively. From Fig. 14 to Fig. 16, when the suitcase is put on the table for a long time, it will be updated into the background frame by EFBD, as shown in Fig. 17. Two persons in the film walk toward to or away from the camera, so parts of blocks will keep steady in the same location for a long time. As a result, these blocks are also updated into the background frame.

Fig. 17 Initial background frame and updated background frame.

We also capture some real surveillance videos for testing the performance of EFBD. Because of camera noises, light changes, and other factors, some additional errors could be involved. The dotted-line area depicted in Fig. 18 is caused by the shadow. We hope that the developed algorithm can detect any moving objects, of course, including the moving shadow. In Fig. 19, many persons move in the same scene simultaneously. EFBD can still extract the regions very well. When the moving object is close to the camera or in the scene that is over-exposed to the light, EFBD even extract the foreground objects successfully, as shown in Fig. 20 and Fig. 21.

In average, EFBD needs 10.12 ms for detecting a frame. After the code optimization, PC Camera can encode the video by H.264 at the speed of 26.84 fps with the proposed EFBD scheme.

Fig. 18 Original frame and resulting EFBD.

Fig. 19 Original frame and resulting EFBD.

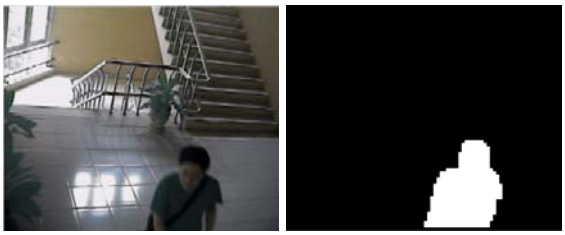

Fig. 20 Original frame and resulting EFBD.

Fig. 21 Original frame and resulting EFBD.

5. CONCLUSIONS

The proposed EFBD algorithm can extract the foreground blocks effectively and quickly. In addition, EFBD is not sensitive to the light variations. The background model can be updated dynamically and timely. The result of foreground block detection is helpful for determining the importance order of multiple surveillance cameras. However, a drawback of EFBD is that it can not accurately extract the foreground blocks with the similar luminance to the background parts.

6. REFERENCES

[1] W. Hu, T. Tan, L. Wang and S. Maybank, "A survey on visual surveillance of object motion and behaviors," IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews, vol. 34, pp. 334-352, 2004.

[2] K. Toyama, J. Krumm, B. Brumitt and B. Meyers, "Wallflower: Principles and practice of background maintenance," in Proceedings of the 1999 7th IEEE International Conference on Computer Vision (ICCV'99), Sep 20-Sep 27 1999, 1999, pp. 255-261.

[3] S. Kamijo, Y. Matsushita, K. Ikeuchi and M. Sakauchi, "Traffic monitoring and accident detection at intersections," Proceedings IEEE Conference on Intelligent Transportation Systems, pp. 703-708, 1999.

[4] D. Gutchess, M. Trajkovic, E. Cohen-Solal, D. Lyons and A. K. Jain, "A background model initialization algorithm for video surveillance," in 8th International Conference on Computer Vision, Jul 9-12 2001, 2001, pp. 733-740.

[5] R. Cucchiara, C. Grana, M. Piccardi and A. Prati, "Detecting moving objects, ghosts, and shadows in video streams," IEEE Trans. Pattern Anal. Mach. Intell., vol. 25, pp. 1337-1342, 2003.

[6] H. Lin, T. Liu and J. Chuang, "A probabilistic SVM approach for background scene initialization," in

International Conference on Image Processing (ICIP'02), Sep 22-25 2002, 2002, pp. 893-896.

[7] A. Makarov, "Comparison of background extraction based intrusion detection algorithms," in Proceedings of the 1996 IEEE International Conference on Image Processing, ICIP'96. Part 1 (of 3), Sep 16-19 1996, 1996, pp. 521-524.

[8] E. Durucan and T. Ebrahimi, "Change detection and background extraction by linear algebra," Proceedings of the IEEE, vol. 89, pp. 1368-1381, 2001.

[9] F. Ziliani and A. Cavallaro, "Image analysis for video surveillance based on spatial regularization of a statistical model-based change detection," Real Time Imaging, vol. 7, pp. 389-399, 2001.

[10] C. Kim and J. Hwang, "Fast and automatic video object segmentation and tracking for content-based applications," IEEE Transactions on Circuits and Systems for Video Technology, vol. 12, pp. 122-129, 2002.

[11] A. J. Lipton, H. Fujiyoshi, and R. S. Patil, “Moving target classification and tracking from real-time video,” in Proc. IEEE Workshop Applications of Computer Vision, 1998, pp. 8–14.

[12] C. Li, Y. Li, Q. Zhuang, Q. Li, R. Wu and Y. Li, “Moving object segmentation and tracking in video,” in Proc. Machine Learning and Cybernetics, vol.8, pp. 4957-4960, Aug. 2005

[13] D. Meyer, J. Denzler, and H. Niemann, “Model based extraction of articulated objects in image sequences for gait analysis,” in Proc. IEEE Int. Conf. Image Processing, 1998, pp. 78–81.D. Meyer, J. Psl, and H. Niemann, “Gait classification with HMM’s for trajectories of body parts extracted by mixture densities,” in Proc. British Machine Vision Conf., 1998, pp. 459–468.

[14] J. Barron, D. Fleet, and S. Beauchemin, “Performance of optical flow techniques,” Int. J. Comput.Vis., vol. 12, no. 1, pp. 42–77, 1994.

[15] C. Kim and J. Hwang, "Fast and automatic video object segmentation and tracking for content-based applications," IEEE Transactions on Circuits and Systems for Video Technology, vol. 12, pp. 122-129, 2002.

[16] T. Meier and K. N. Ngan, “Video segmentation for content-based coding,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 9, pp. 525-538, Dec. 1999.