An Efficient Video Coding Algorithm Targeting Low Bitrate Stationary Cameras

Ngoc-Sinh Nguyen, Duy-Hieu Bui, and Xuan-Tu Tran

SIS Laboratory, VNU University of Engineering and Technology, Hanoi, Vietnam

E-mail: {sinhnn_55,hieubd,tutx}@vnu.edu.vn

Abstract One of the crucial factors in video monitoring systems or video surveillance systems is the bitrate while the complexity plays the most important part in the embedded system. Recently, people are trying to reduce the bitrate of these types of systems and the complexity of the algorithm for embedded systems. The bitrate can be reduced by using the current state-of-the-art video codecs such as H.264/AVC or the newest video standard H.265. However, these codecs have very high processing dependencies and complexity. By employing the properties of stationary cameras, we proposed an efficient video coding algorithm for embedded systems with low complexity, and good compression ratio. The algorithm is the combination of an simple but powerful motion detection, called $\Sigma - \Delta$, and JPEG algorithm. After the first frame in a sequence, the computation is reduced by processing only the moving blocks instead of the whole frame. The proposed algorithm has been evaluated and validated using SystemC.

Keywords JPEG, Sigma-Delta, Motion dectection, Zipfian estimation, video codec

1. Introduction

Nowadays, video cameras are applied widely for monitoring and recording events such as traffic monitoring and surveillance systems [7]. A monitoring system may contain hundreds of stationary cameras to record events then transmit them to the processing center or putting them in storage. One of the problems of this system is how to reduce the bitrate and the storage size of the encoded sequence without increasing the complexity of the embedded system.

A large number of cameras used in the monitoring system are stationary cameras. Stationary cameras are the cameras which do not change their positions throughout the recording process. The recorded scene of this kind of cameras can be divided into two parts: the foreground and the background. The background is the stationary scene which is almost unchanged or changed very slowly. The foreground is the part which contains motion or moving objects. The foreground can be extracted from video sequence using motion detection algorithms [14].

Many motion detection algorithms have been developed based on background subtraction such as [11] [13]. However, these methods are not efficient because of huge memory usage, computational complexity or the inaccurate model of the background which leads to the noise and the false detection [9]. Our algorithm use a new motion detection method with the dynamic background estimation algorithm, called $\Sigma - \Delta$ background estimation.

The storage size and the bitrate of the original videos can be reduced by using video codecs [6][10]. The current international codecs such as H.264/AVC or H.265 can highly reduce the bitrate of the system. However, these codecs have high complexity which leads to the difficulties of implementing them for embedded systems. The complexity of these codecs also leads to the inefficient in power consumption and the real-time feature of the embedded system. In contrast, JPEG is an efficient algorithm widely used for photographic still picture. It can also be used for motion pictures. The coding tools of JPEG are much more simpler than the one of the current codecs [3].

In this paper, we proposed a simple video compression and decompression algorithm which based on JPEG image compression standard combined with $\Sigma - \Delta$ motion detection algorithm. By doing motion detection, we can reduce the processing step and reducing the bitrate by encoding and sending only the moving blocks. Our algorithm are implemented using SystemC.

The rests of the paper are organized as follows. Section 2 in-

Copyright ©2013 by IEICE

troduces the basic of JPEG and motion detection algorithm used in our work. In section 3, our proposed algorithm which is the combination of JPEG compression with the help of motion detection is presented. Section 4 is the experimental results of the proposed algorithm. Finally, there are some conclusions and remarks in section 5.

2. Introduction to motion detection and JPEG compression

Our main work based on JPEG compression standard and the $\Sigma - \Delta$ motion detection technique. In this section, the basic of JPEG compression standard and the motion detection algorithm are introduced.

2.1. Motion detection

Motion detection is the process of detecting the change in position of an objects relative to its previous one. It is one of the most important steps in a video surveillance system. Thank to the motion detection algorithm, the foreground, the moving objects or events can be extracted from the stationary scene or the background. After that, the foreground or the moving object can be used for tracking, detecting abnormal events and so on.

In motion detection, each scene can be divided into three types of motion: still objects, the slow motion in clustering, and the foreground. The still objects do not change their positions throughout the time. The slow motion in clustering, for example the slow motion of the leaf because of the breeze, can be considered as noise. The most interest object in motion detection is the foreground – the moving objects. A good motion detection algorithm should be able to detect correctly the moving objects while to discard the slow motion in clustering and the noise.

There is a large number of motion detection algorithms such as optical flow [4], Gaussian estimation, frame difference or background subtraction [2], kernel density estimation [12] etc. However, these methods are complex and they consume a large amount of memory and processing power. This leads to the difficulties in implementing them into the embedded system, maintaining the real-time properties of the system, and the power consumption problems.

In this paper, we choose a simple algorithm called $\Sigma - \Delta$ which can dynamically estimate the background changes, discard the slow motion in clustering, and detecting the moving object correctly with simple computation and low resource requirement. The basic $\Sigma - \Delta$ is described in algorithm 1.

Algorithm 1: Basic $\Sigma - \Delta$ [9]

```
foreach pixel $x$ do
    if $M_{t-1}(x) < I_t(x)$ then $M_t(x) \leftarrow M_{t-1} + 1$
    if $M_{t-1}(x) > I_t(x)$ then $M_t(x) \leftarrow M_{t-1} - 1$
    otherwise $M_t(x) \leftarrow M_{t-1}$
foreach pixel $x$ do
    $O_t(x) = |M_t(x) - I_t(x)|$
foreach pixel $x$ do
    if $V_t(x) < N \times O_t(x)$ then $V_t(x) \leftarrow V_{t-1}(x) + 1$
    if $V_t(x) > N \times O_t(x)$ then $V_t(x) \leftarrow V_{t-1}(x) - 1$
    otherwise $V_t(x) \leftarrow V_{t-1}(x)$
foreach pixel $x$ do
    if $O_t(x) < V_t(x)$ then $E_t(x) \leftarrow 0$ else $E_t(x) \leftarrow 1$
```

The input to the algorithm is the pixel value $I_t(x)$. For the first frame, the first estimation value $M_t(x)$ is set to $I_t(x)$. After that, the estimation of the background $M_t(x)$ and the estimation of the variation $V_t(x)$ is dynamically calculated base on the difference $O_t(x)$ between the two consecutive frame. The parameter N is from 1 to 4 to discard the irrelevant motion. The smaller N is, the more the algorithm is sensitive to the slow motion. The motion $E_t(x)$ will be decided if the difference is greater than or equal to the estimated variance $V_t(x)$.

The $\Sigma - \Delta$ estimation algorithm is non-linear and recursive operation. The most advantage of this algorithm is low complexity and low resource requirement. The sensitive of the algorithm can be controlled by the parameter N in the algorithm. The $\Sigma - \Delta$ algorithm can be improved further by using the probabilistic model of Zipf-Mandelbrot. This method is called Zipfian estimation [8]. This method adds two additional parameter σ and T_V to update the background estimation $M_T(x)$ only for the moving pixels (the variation is greater than σ and the variation is updated only after a predefined time T_V. The detailed algorithm is presented in algorithm 2.

2.2. JPEG

JPEG has become the most popular image compression method from its announcement by Join Photographic Expert Group [5]. It is a source coding technique, applied for continuous-tone still images, vastly used in digital cameras, the storage of digital images on computers, as well as the transmission of these images in the network. Good performance and adjustable compression ratios are the main goals of this standards.

JPEG standard defines four modes of operation: sequential lossless mode, sequential DCT mode, progressive DCT-based mode, hi-

Algorithm 2: Zipfian estimation [9]

```
find the greatest $2^p$ that divides ($t$ mod $2^m$)
set $\sigma = 2^m/2^p$
foreach pixel $x$ do
    if $V_{t-1}(x) > \sigma$ then
        if $M_{t-1}(x) < I_t(x)$ then $M_t(x) \leftarrow M_{t-1} + 1$ if
        $M_{t-1}(x) > I_t(x)$ then $M_t(x) \leftarrow M_{t-1} - 1$
        otherwise $M_t(x) \leftarrow M_{t-1}$
    else
        $M_t(x) \leftarrow M_{t-1}$
foreach pixel $x$ do
    $O_t(x) = |M_t(x) - I_t(x)|$
foreach pixel $x$ do
    if $t$ mod $T_V = 0$ then
        if $V_t(x) < N \times O_t(x)$ then $V_t(x) \leftarrow V_{t-1}(x) + 1$
        if $V_t(x) > N \times O_t(x)$ then $V_t(x) \leftarrow V_{t-1}(x) - 1$
        otherwise $V_t(x) \leftarrow V_{t-1}(x)$
foreach pixel $x$ do
    if $O_t(x) < V_t(x)$ then $E_t(x) \leftarrow 0$ else $E_t(x) \leftarrow 1$
```

erarchical mode. Among these modes of operation, baseline profile which is sequential DCT-based mode with 8-bit input and Huffman entropy encoding is used in most applications. In this work, we also use the baseline profile of JPEG standard because of its popularity and good performance with low complexity.

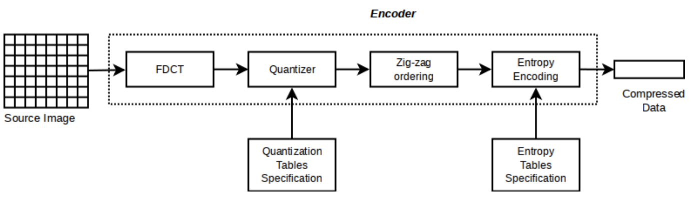

Figure 1: Baseline JPEG encoder

The basic coding tools of JPEG standard include Discrete Cosine Transform (DCT) 2-D transform, quantization, zigzag scan and entropy encoding. Figure 1 shows the coding process of a baseline profile JPEG compression process. At first, the source image is divided into 8×8 blocks. After that, these blocks are transformed using Forward Discrete Cosine Transform (FDCT). The transformed coefficients are quantized using a quantization table and read in a zigzag scan. Finally, the zigzag sequence are processed by entropy encoder and the coded sequence is obtained. The decoding process is just the reverse of the encoding process.

In our system, each frame is also divided into 8×8 blocks. After the background frame is sent, we only calculate the FDCT, quantization and entropy encoding for a moving blocks. Other static block is set as zero blocks. The coded bit of the static blocks are the code world EOB (End Of Block) as specified in the JPEG standard.

3. Proposed algorithm

Using the $\Sigma - \Delta$ algorithm and JPEG standard, the system is divided into two main parts. The first part is the motion detection and extraction module. The second part is the JPEG encoding/decoding module.The basic processing unit of the system is a 8×8 block. Figure 2 shows the block diagram of the proposed encoder while Figure 3 shows the structure of the decoder.

In the encoder, the first frame is encoded using the JPEG encoder only. After that, when there is the motion information, the system only encode the moving block. The source frame I_t is continuously fed into the motion detection module. After motion detection, the stationary block and the moving block will be extracted. In case of a stationary block, the DC value is set to 0 and the EOB symbol is sent directly to the entropy encoding process. If a block is a moving block, it will be subtracted by the corresponding block from the reconstructed image (frame I_{t-1}). The residuals are fed to the JPEG encoder and encoded using normal JPEG encoding process to reduce the bit-rate further.

The inputs fetched to the forward discrete cosine transform (FDCT) have to be in range -128 to 127 inclussively. Therefore, the input data is calculated as in equation 1, in which the $D_t(x)$ is the input data to the FDCT, $I_t(x)$ is the moving pixel in the moving block and $I_{t-1}(x)$ is the reconstructed pixel in the previous frame at the same location.

$$D_t(x) = \frac{I_t(x) - I'_{t-1}(x)}{2} \tag{1}$$

The decoder's structure is simpler than the encoder. The decoder does not need the motion detection process. Figure 3 shows the proposed decoding structure. Based on the information from the inverse entropy encoding, the stationary block and the moving blocks can be decided based on the DC value and the EOB symbol. Every block which have the DC value and the AC coefficients different from 0 is considered as the moving block, otherwise, it is a stationary block. For each stationary block, the pixels are directly copied from the previously decoded image. For the moving block, after the inverse zigzag scan, the inverse quantization and the inverse discrete cosine transform (IDCT), it will be rebuild using the equation 2. The $I'_t(x)$ is the decoded pixel, the $D_t(x)$ is the residual after the

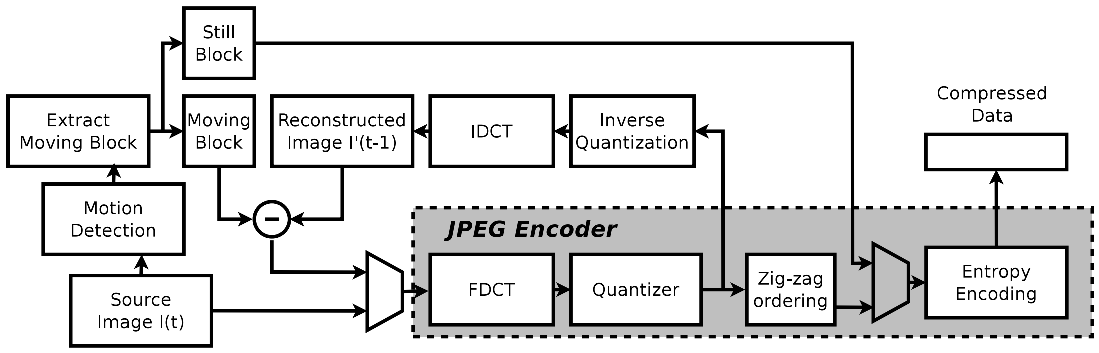

Figure 2: The proposed encoding structure

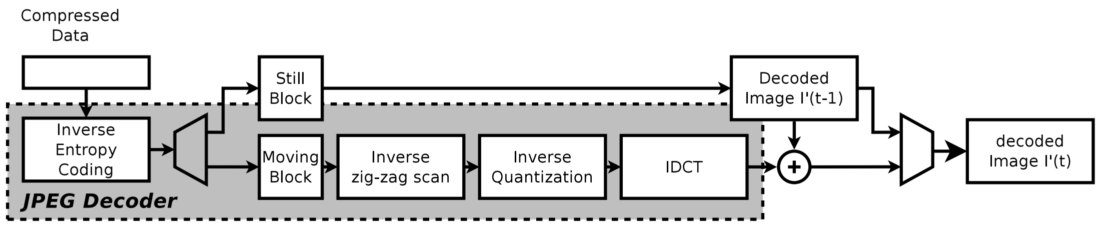

Figure 3: The proposed decoding structure

IDCT, and $I'_{t-1}(x)$ is the decoded pixel in the previous frame.

$$I'_t(x) = 2 \times D_t(x) + I'_{t-1}(x) \qquad (2)$$

4. Experiment Results

The proposed algorithm with the encoding and decoding structure described in section 3 is first implemented using pure C/C++. After that is refined with SystemC library to model the algorithm in parallel and the pipeline architecture. The test program reads the raw video and encode it into JPEG file. In our first version, it only encodes the Luma component Y.

The proposed algorithm is tested with difference Video sequences including "Coastguard", "Hall Monitor", "Akiyo", "Container", and "Deadline". All these sequences are downloaded from [1]. The "Coastguard" sequences are captured using moving cameras, while the others are captured using stationary cameras. The "Hall Monitor" sequence is an indoor video with background noise. The "Akiyo" sequence and the "Deadline" sequence are also captured by the indoor camera but with very little background noise. Difference from the "Hall" sequence, these two sequence also have the moving object from the first frame. The "Deadline" sequence has very fast motion with some sudden changes. The "Container" sequence is the outdoor stationary video with a few moving objects. Each sequence consists of 300 frames in the YUV4MPEG format. The resolution of each video is 352×288 (CIF). The objective visual quality is measured using the Peak Signal-to-Noise Ratio (PSNR) and the compression ratio for the Y component only.

The average number of the stationary blocks for each test sequence is shown in Table 1. The parameter N in the motion detection algorithm is used to discard the noise and the slow motion. Therefore, with the $N = 2$, there are many stationary blocks. For example, the "Akiyo" sequence and the "Deadline" sequence has more than 50% of the stationary blocks. The "Hall Monitor" sequence and the "Container" sequence have about 30% of the stationary blocks with $N = 2$ because of the noise and the slow motion in cluster. In these sequence, the more stationary blocks can be obtained by increasing N to 3 or 4. The "Coastguard" sequence have the least number of stationary blocks because the capturing camera is moving. This leads to the inaccurate motion detection. Figure 4 shows the effect of the parameter N to the compression ratio and the PSNR with the update period $T_v = 4$ and the quality factor $qp = 80$.

Table 1: The average number of the stationary blocks

Test sequence	$N = 1$	$N = 2$
Akyo	1004	1162
Hall	175	655
Container	121	560
Coastguard	2	133
Deadline	703	1095

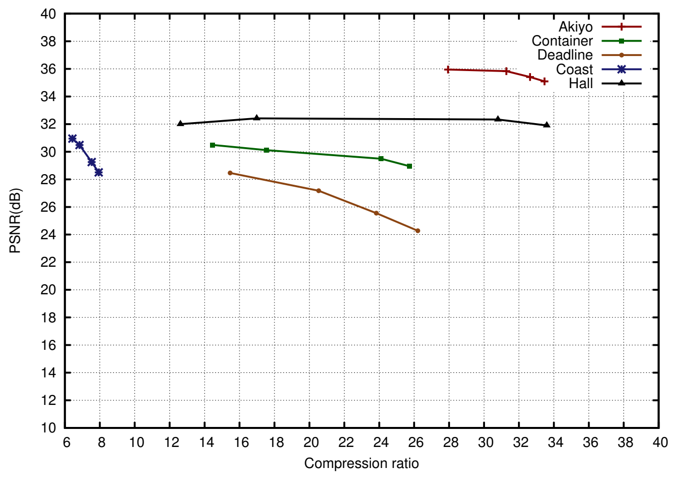

Figure 4: The Effect of the parameter N from 1 to 4 to the compression ratio and the quality with the fixed quality factor

Table 2 shows the number of computation for each block of the Zipfian estimation, FDCT and quantization process in our current implementation. The FDCT and the quantization required about 1000 additions and 200 multiplications for each blocks. The operations for motion detection are about 200 additions and about 80 shift operations for each block. With the optimized Zipfian estimation discussed in section 2.1, the saved operation is huge for video with more than 50% of the stationary blocks. For example, the "Akiyo" sequence can save up to 1 million multiplications. For the video with the 30% of the stationary blocks, the number of the operation is nearly the same as the original JPEG encoder, but the compression ratio is much better (doubling the compression ratio in comparison with the original JPEG encoder).

Table 2: The number of computation per blocks for Zipfian estimation, FDCT and quantization in our current implementation

Operation	Zipfian	FDCT	quantization
Addition	200	928	64
Multiplication	0	192	64
Shift	20	128	0

Figure 5 shows the comparison of our algorithm with the original baseline profile JPEG encoder with the quality factor ranging from 100 to 10. It is obvious that our algorithm have better compression ratio in all three test sequences with acceptable visual quality in comparison with the result of the original JPEG encoder. Our system can achieve up to 60 times in the compression ratio, while in the original JPEG encoder, the compression ratio is about 30. The visual quality in our system drops gradually with the increase in the compression ratio. In contrast, the original JPEG encoder decrease sharply with little increase in the compression ratio.

5. Conclusion

The embedded system requires efficient algorithms which can reduce the processing step and processing time as well as saving the bandwidth usage. In this paper, we presented an efficient algorithm for embedded system targeting stationary camera systems. The algorithm is the combination of a very fast, low resource and efficient motion detection algorithm, named $\Sigma - \Delta$, and JPEG – a well-known standard for image/video compression. The processing power is reduced by doing the compression and decompression only for the moving blocks. The evaluated results show that our algorithm is efficient in term of both performance and the visual quality of the stationary cameras. In the future, we would like to implement this algorithm into hardware to further evaluate it with other well-known codecs such as H.264/AVC.

References

[1] Test video sequences. `http://media.xiph.org/video/derf/`.

[2] O. Barnich and M.Van Droogenbroeck. Vibe: A universal background subtraction algorithm for video sequences. *Image processing, IEEE transactions*, 20, 2011.

[3] Charles Hollemeersch Bart Pieters, Jan De Cock. Ultra high definition video decoding with motion jpeg xr using the gpu. In *The 18th IEEE International Conference on Image Processing*, pages 377–380, 2011.

[4] S. Denman, C. Fookes, and S. Sridharan. Improved simultaneous computation of motion detection and optical flow for object tracking. In *Digital Image Computing: Techniques and Applications, 2009. DICTA '09.*, pages 175–182, 2009.

[5] The International Telegraph and Telephone Consultative Commitee. *ITU-T Recommendation T.81*, September 1992.

[6] S. kamath and J.R. Jackson. Low-bit rate motion jpeg using differential encoding. In *Signals, Systems and Computers, 2004. Conference Record of the Thirty-Eighth Asilomar Conference on*, volume 2, pages 1723–1726 Vol.2, 2004.

[7] Limin Liu, Zhen Li, and Edward J. Delp. Efficient and low-complexity surveillance video compression using backward-channel aware wyner-zip video coding. *IEEE Transactions on*

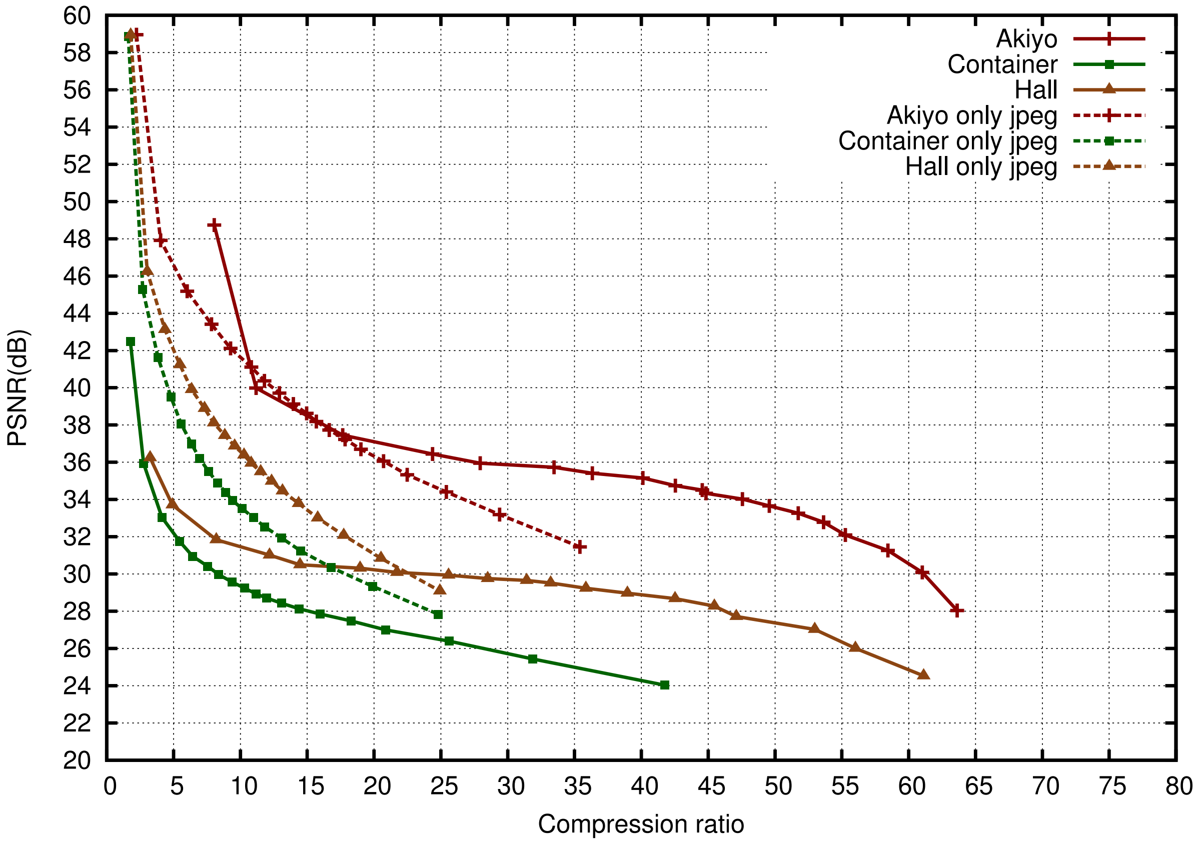

Figure 5: Comparision with the original JPEG baseline algorithm

Circuits and Systems for Video Technnology, 19(4):453–465, April 2009.

[8] A. Manzanera. Sigma-delta background subtraction and the zipf law. *CIARP.LNCS*, 28-2:42–51, 2007.

[9] A. Manzanera and J. C. Richefue. A robust and computationally efficient motion detection algorithm based on delta-sigma background estimation. *ICVGIP. IEEE*, 2004.

[10] J. Meessen, C. Parisot, X. Desurmont, and J.-F. Delaigle. Scene analysis for reducing motion jpeg 2000 video surveillance delivery bandwidth and complexity. In *The IEEE International Conference on Image Processing, 2005 (ICIP 2005)*, volume 1, pages I–577–80, 2005.

[11] A. Mittal and N. Paragios. Motion-based background subtraction using adaptive kernel density estimation. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–302–II–309 Vol.2, 2004.

[12] A. Mittal and N. Paragios. Motion-based background subtraction using adaptive kernel density estimation. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2004)*, volume 2, pages II–302–II–309 Vol.2, 2004.

[13] N.M. Oliver, B. Rosario, and A.P. Pentland. A bayesian computer vision system for modeling human interactions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):831–843, 2000.

[14] Liu Xuedong, Liu Jian, and Tan Yihua. A motion detection scheme for video surveillance. In *IET Conference on Wireless, Mobile and Sensor Networks (CCWMSN07)*, pages 705–708, 2007.