A Parallel Algorithm for Change Detection

Mian M. Mubasher, M. Shahid Farid, Abdul Khaliq, Muhammad Murtaza Yousaf
Punjab University College of Information Technology
University of the Punjab, Lahore, Pakistan
Email: mian.mubasher@gmail.com, shahid.fareed, abdul.khaliq, murtaza@pucit.edu.pk

***Abstract*—Change detection is an important research area in image and video processing due to wide applications in medicine, surveillance, remote sensing, geographical information processing and in defense and security. This paper presents a real-time, distributed algorithm to detect changes in videos. The proposed algorithm may be used in many change detection applications that require real-time performance like CCTV Security Surveillance Camera Systems. In such applications fast change detection is very important. Many a times, we need to search a video captured by security camera for an event with significant change. Doing it manually is very time consuming and cumbersome, a fast automatic algorithm is desired in such cases. The proposed parallel algorithm runs on a cluster and detects the significant changes in real-time. The algorithm is implemented in C using MPICH2 in Linux environment and is tested over a number of videos for correctness, accuracy and execution time. The proposed approach is also compared with existing change detection techniques and gained speedup of approximately 3.4 on cluster with 4 average machines.**

***Index Terms*—Change detection, Distributed algorithm, Parallel algorithm, Security camera application.**

I. Introduction

Change detection is the measuring the changed area in a video over time. Change detection is a very important area in computer vision and image processing due to its widespread applications. It used to determine land erosion, crops cultivated in a region and other such GIS processes. Change detection is used in video surveillance [1] to monitor the activities performed in a particular area under observation. These activities may include object detection, identification and classification [2], [3], humane action analysis [4], [5], [6], [7], determining the number of vehicles on the road [8], [9], [10], [11]. Change detection is used in geographical information processing to detect the changes in a particular area of a region by comparing two time variant aerial photographs or satellite images [12], [13], [14].

Change detection may be used in security surveillance camera systems to detect the significant change in the scene. Based on this change, the system may take appropriate action like ring the alarm or call the security office. To use change detection in such real-time applications, a very fast, robust and accurate algorithm is required. Many excellent algorithms have been proposed in literature to detect change in video. These algorithm may not be as useful in security surveillance system where a real-time algorithm is required. This motivation led us to develop a parallel algorithm for change detection, that must be able to process at least 30 frames in a second so that it can be used with live camera streams.

The simplest way to detect change in two frames is by subtracting one from other and thresholding the result. This technique is too simple to handle complex cases like scene with varying illumination, subtle changes in some parts of the data etc. An extension of image differencing technique was presented in [15] that is based in super-pixel change detection. An class of algorithms computes a running mean background image and takes some detection measure like difference, variance, co-variance etc from the current frame to the mean background image. These techniques are better than simple subtraction but they too suffer from the above described problems. A pyramid based change detection approach was described in [16]. An other hierarchical approach is described in [17]. Stauffer and Grimson described an adaptive background subtraction model [18]. Another adaptive background model was proposed in [19].

Change detection in color images is a sub-area in change detection that is dedicated to background subtraction or change detection in color images, videos. Fisher described change detection techniques in RGB and HSI color models [20]. An illumination invariant change detection approach was described by Cavallaro [21]. The technique used color edge detection with image differencing to detect the change. Malik, R. et.al; used change detection in traffic applications [22]. They proposed an approach to detect and extract the road signs.

Change detection is used in many applications like detecting and tracking of human activities in indoor or outdoor. Much research is going on in this field like [23] in which a real time visual surveillance system was proposed. The technique works for infrared videos and detects the multiple people in a science, segment them and labels them to track their body parts like head, nose, feet etc. An other such system that detects multiple human and tracks them is presented by Tao Zhao et.al [24]. A Hidden Markov Model based technique for human behavior understanding was presented in [25] from live stream in a nursing environment. Keming Chen described Change detection based on adaptive Markov Random Fields in [26]. Various other methods have been developed and proposed for multiple human detection and tracking are presented like [27], [28], [29], [30]. A crowed segmentation from background image was presented in [31]. This technique is used active basis model for for change detection. A good study on change detection is described in [32], [33], [21], [34].

To the best of our knowledge, no parallel algorithm has been proposed yet for change detection except [35]. In this paper the authors proposed a distributed algorithm for change detection in satellite images of land to determine the changes. Their algorithm takes two multispectral images. Each band is processed by a node in the parallel environment.

In this manuscript a distributed algorithm for change detection is proposed that works upon a cluster. The proposed approach is based on the classical Gaussian change detection algorithm which is implemented in multi-processor environment. The algorithm works in two steps, first, the algorithm is trained on a set of background frames and in second step, the frames tested with the background model to detect the change. In the nest section, the proposed algorithm is described. In section III time complexity analysis of the proposed algorithm is presented and in section IV experimental results are presented. Section IV-A compares the results with existing classical technique and measure the speedup ratio. The paper is concluded in section V.

II. Proposed Distributed Algorithm

The proposed distributed change detection algorithm works in two phases. The first phase is computing the background model using Gaussian distribution model. In the second phase, for each pixel in a frame the probability of likelihood that this pixel belongs to the background model is computed and the pixel is classified as foreground if its value is greater than a threshold and background otherwise and this process is done in parallel way. The pixels categorized as fore ground form the changed region. The following two subsection explain the algorithm in detail.

A. Computing background model

We assume that the color variation of a particular pixel in a set of frames follows the Gaussian distribution. Based on this assumption, we construct a Gaussian background model, μ, Σ. The model is constructed from k number of background frames. Let the f_i be the i^{th} frame in the set and the size of each frame is $m \times n \times 3$. Using these k background frames, the mean μ and the covariance Σ are computed. The μ is the mean of red, green and blue components of every pixel in k frames. That is,

$$\mu = [\mu_r, \mu_g, \mu_b]^t$$

where μ_r, μ_g and μ_b mean of red, green and blue components with size $m \times n$ and are computed as follows:

$$\begin{aligned} \mu_r(x,y,1) &= \tfrac{1}{k}\sum_{i=1}^{k} f_i(x,y,1) \\ \mu_g(x,y,2) &= \tfrac{1}{k}\sum_{i=1}^{k} f_i(x,y,2) \\ \mu_b(x,y,3) &= \tfrac{1}{k}\sum_{i=1}^{k} f_i(x,y,3) \end{aligned} \tag{1}$$

Here, f_i is the i^{th} frame. The size of μ is $m \times n \times 3$. The covariance Σ for each pixel (x,y) is computed as:

$$\sum(x,y) = \begin{bmatrix} \sigma_{(r,r)}(x,y) & \sigma_{(r,g)}(x,y) & \sigma_{(r,b)}(x,y) \\ \sigma_{(g,r)}(x,y) & \sigma_{(g,g)}(x,y) & \sigma_{(g,b)}(x,y) \\ \sigma_{(b,r)}(x,y) & \sigma_{(b,g)}(x,y) & \sigma_{(b,b)}(x,y) \end{bmatrix} \tag{2}$$

where σ is variance between two color components and is given by Equation 3.

$$\begin{aligned} \sigma_{(r,r)}(x,y) &= \frac{1}{k-1}\sum_{i=1}^{k}((f_i(x,y,1)-\mu_r(x,y,1))^2 \\ \sigma_{(r,g)}(x,y) &= \frac{1}{k-1}\sum_{i=1}^{k}((f_i(x,y,1)-\mu_r(x,y,1) \\ &\qquad (f_i(x,y,2)-\mu_r(x,y,2))) \\ \sigma_{(r,b)}(x,y) &= \frac{1}{k-1}\sum_{i=1}^{k}((f_i(x,y,1)-\mu_r(x,y,1) \\ &\qquad (f_i(x,y,3)-\mu_g(x,y,3))) \\ \sigma_{(g,g)}(x,y) &= \frac{1}{k-1}\sum_{i=1}^{k}((f_i(x,y,2)-\mu_g(x,y,2))^2 \\ \sigma_{(g,b)}(x,y) &= \frac{1}{k-1}\sum_{i=1}^{k}((f_i(x,y,2)-\mu_g(x,y,2) \\ &\qquad (f_i(x,y,3)-\mu_b(x,y,3))) \\ \sigma_{(b,b)}(x,y) &= \frac{1}{k-1}\sum_{i=1}^{k}((f_i(x,y,3)-\mu_b(x,y,3))^2 \end{aligned} \tag{3}$$

The value of k is important for a good background model. It is found from the experiments that $k > 50$ results in a good background model. From Equation 3, you can see that a lot of computation is required. Since, this computation is done only once, it is not advantageous to parallelize this step. This completes the background modeling step. Based on μ and Σ, the change in each frame is detected in a parallel way, explained in the next section.

B. Change detection

After the construction of background model, the system is ready to detect the change. For each pixel (x,y) of an input frame, its likelihood $P(\mathbf{x}|\mu,\Sigma)$ with the background model is computed. This probability is compared with the threshold τ and pixel (x,y) is marked as background (unchanged) pixel if its probability is greater than the threshold τ and marked as foreground (changed) pixel otherwise. The likelihood of pixel $\mathbf{x} = [\mathbf{x}_r, \mathbf{x}_g, \mathbf{x}_b]^T$ is computed as:

$$P(\mathbf{x}|\mu,\Sigma) = \frac{1}{(2\pi)^{\frac{3}{2}}\sqrt{|\Sigma|}} e^{\frac{-1}{2}(\mathbf{x}-\mu)^T(\Sigma)^{-1}(\mathbf{x}-\mu)} \tag{4}$$

To save the computations, instead of computing the likelihood, we may compute the $(\mathbf{x}-\mu)^T(\Sigma)^{-1}(\mathbf{x}-\mu)$ only, called Mahalnobis distance. The operation of computing the likelihood involves the following computational steps:

- Evaluation of $(2\pi)^{\frac{3}{2}}$. Since, there is no variable in computation, it may be evaluated at the start of the process once.

- $\sqrt{|\Sigma|}$ has to be computed for each pixel as each pixel in the background model has a different value of Σ. This operation involves computing the determinant of $\Sigma(x, y)$ followed by a square root operation. Computing the determinant takes 12 multiplications and 5 addition and subtraction operation.
- $(\mathbf{x} - \mu)^T$ is subtraction of two matrices followed by a transpose operation.
- Σ^{-1} computes the inverse of Σ and requires $O(p^3)$ time for a matrix of dimension $p \times p$.
- $\mathbf{x} - \mu$ requires same cost as step 3.

This is evident that computing likelihood is a very costly operation and this computation is done for each pixel in the frame. In a live stream from camera, around 30 frames of size $m \times n$ are received in a second and this operation has to be done for each pixel. Common resolution of camera is 1280×1024, that is $1,310,720$ number of pixels in one frame and thus, the total number of pixels in frames received in one second are $39,321,600$. On the average, one likelihood computation requires around 60 operations, giving the total number of operation to be performed in one second are $2,359,296,000$ that is about 36 million operations. Performing such a huge number of complex computational operations in one second may not be feasible on a single machine in real time mode; massive parallelism is required in this case.

To achieve the real time change detection, the proposed algorithm runs on a cluster of computers. It divides each frame into a number of sub-frames and distributes the them amongst the available nodes in the cluster. Each node, then, computes the change in its received data and returns the results back to the manager node. The manager nodes collects the data from the nodes and saves the resultant frame. Computing the mean μ and covariance Σ is done as a pre-processing step in serial way.

Let k be the number of nodes in the cluster and $m \times n$ be the size of the frame. The cluster works in *master-slave* fashion. The master nodes, on receiving a frame from the camera, treats each of the m rows as an element and decomposes the frame into $\frac{m}{k}$ chunks where each chunk has k rows in it. The master node distributes each chunk amongst the k nodes by using `MPI_Scatter()` call as shown in Figure 1. Each node performs computation on the received data and sends the results back to the master node which uses `MPI_Gather()` to gather the results received from the slaves node. This process is repeated for every chunk in the frame and result is accumulated. The code doing this process is given in Appendix.

III. Time Complexity Analysis

The time complexity of computing the change detection in frame of size $m \times n$ in serial way takes $O(mn)$ time as there mn number of pixels in the frame and computation is done for each pixel. The proposed algorithm divides the total number of pixels into k buckets, each computed by a different processing node. Hence, the time it takes to detect change in

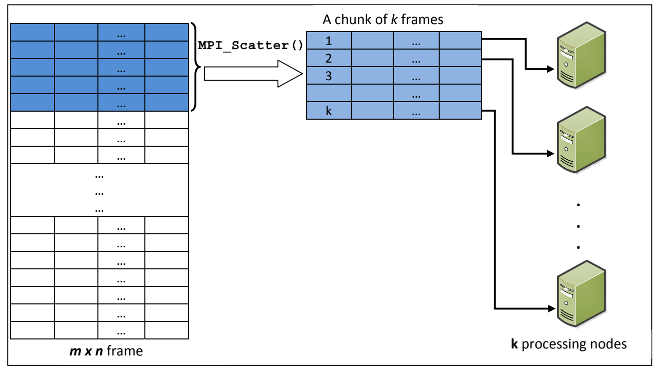

Fig. 1: Process of data distribution.

a frame is is decreased by a factor of k ideally, giving $O(\frac{mn}{k})$ time complexity. Though, theoretically this bound is correct but practically the running of any parallel algorithm may not be k times smaller than its serial time. This is because of the communication cost involved in the inter-node communication and the time spent in distributing the data and gather the results. The actual speed-up of the proposed algorithm over the traditional Gaussian change detection algorithm is given the next section.

IV. Experiments and Results

The proposed parallel algorithm is implemented in C with OpenCV image processing library using MPICH2 on Linux platform. The cluster used in experiments consists of 4 machines and is named 'beowulf' installed at P&DC LAB at PUCIT. A number of experiments are done to evaluate the effectiveness, correctness and robustness of the algorithm and to measure the speed-up gain. The same experiments were also carried out using the standard serial Gaussian change detection approach and execution time were noted against each experiment. Each machine in the cluster is Intel(R) Core(TM)2 Dou CPU with 2.19GHz processor with 1GB of RAM.

Results of five experiments are described in this section. In the first experiment, the video has 80 number of frames and took just 1.92 seconds to complete the change detection process. The results of the detection are shown in Figure Figure 2 (only few of the total frames are shown in these experiment). Table I summarizes the experiments. The statistics shows the effectiveness and speed of the algorithm. With only 4 average

TABLE I: Execution time for the given experiments. T_p is Execution Time is in seconds, τ is the threshold used in Mahalanobis and and γ is the number of frames processed per second.

Exp. No.	Frame Size	No. of frames	T_p	τ	γ
1	640×480	80	1.92	20	41.57
2	640×480	67	1.65	45	40.49
3	640×480	43	1.03	40	41.74
4	384×288	264	6.49	21	40.65
5	392×296	54	1.35	20	40.15

TABLE II: Execution time for the given experiments using serial approach. T_s is Execution Time is in seconds and γ is the frame rate achieved.

Exp. No.	Frame Size	No. of frames	T_s	γ
1	640×480	80	6.60	12.12
2	640×480	67	5.71	11.73
3	640×480	43	3.45	12.46
4	384×288	264	21.31	12.39
5	392×296	54	4.63	11.66

TABLE III: Execution time comparison of serial and parallel algorithms.

Exp. No.	T_s	T_p	Speed-up $\frac{T_s}{T_p}$
1	6.60	1.92	3.429
2	5.71	1.65	3.451
3	3.45	1.03	3.349
4	21.31	6.49	3.281
5	4.63	1.35	3.442

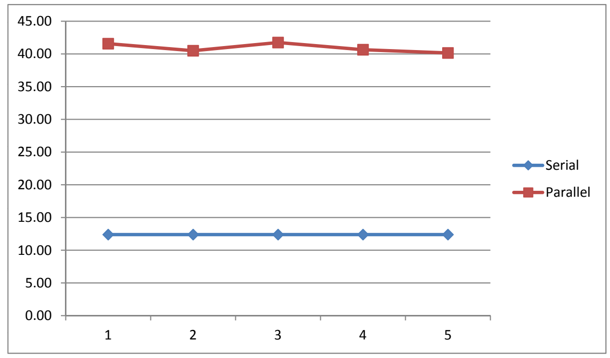

Fig. 7: Comparison of serial and parallel execution times of the 5 experiments.

machines, approximately 40 frames are processed per second which is enough to use it as in real time applications.

A. Speedup Comparison with Traditional Approach

The experiments reported in previous section were repeated with a single processing machine with same specifications and execution times were recorded. Table II shows the execution time with traditional change detection approach. A comparison of execution time of parallel approach and serial approach is presented in Table III. It is found from the the proposed parallel algorithm performs, on the average 3.391 times faster than serial algorithm (see Figure 7). The gain in speedup is not exactly 4 due data transmission and communication latency.

V. Conclusion

This paper presented a fast distributed algorithm for change detection in videos. Change detection in real time is an important task due to its vast and crucial applications in different fields of life. It is a challenging because it requires huge number of computations to done in a small amount of time on a huge data, almost 30 frames in a second. The proposed algorithm is based on Gaussian change detection approach and computes the mean and covariance matrices from a set of background frames. The master node decomposes the incoming frame into a number of sub-frames, each sent to a processing node. Using the mean and covariance, the processing node computes the change in its sub-frame and returns the results back to the master node. It is found from the experiment that the proposed algorithm performs more than 3 times faster than the traditional change detection algorithm on the cluster of 4 average computers.

References

[1] R. Collins, A. Lipton, and T. Kanade, "A System for Video Surveillance and Monitoring," in *American Nuclear Society 8th Internal Topical Meeting on Robotics and Remote Systems*, 1999.

[2] Feng Wang, Yu-Gang Jiang, and Chong-Wah Ngo, "Video event detection using motion relativity and visual relatedness," in *Proceedings of the 16th ACM international conference on Multimedia*, New York, NY, USA, 2008, MM '08, pp. 239–248, ACM.

[3] Christopher Wren, Ali Azarbayejani, Trevor Darrell, and Alex Pentland, "Pfinder: Real-time tracking of the human body," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, pp. 780–785, 1997.

[4] Larry S. Davis, Sandor Fejes, David Harwood, Yaser Yacoob, Ismail Haratoglu, and Michael J. Black, "Visual surveillance of human activity," in *Proceedings of the Third Asian Conference on Computer Vision-Volume II*, London, UK, UK, 1997, ACCV '98, pp. 267–274, Springer-Verlag.

[5] Rodrigo Cilla, Miguel A. Patricio, Antonio Berlanga, and José M. Molina, "A probabilistic, discriminative and distributed system for the recognition of human actions from multiple views," *Neurocomput.*, vol. 75, no. 1, pp. 78–87, Jan. 2012.

[6] Ronald Poppe, "A survey on vision-based human action recognition," *Image Vision Comput.*, vol. 28, no. 6, pp. 976–990, June 2010.

[7] P. Turaga, R. Chellappa, V.S. Subrahmanian, and O. Udrea, "Machine recognition of human activities: A survey," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 18, no. 11, pp. 1473 –1488, nov. 2008.

[8] Zehang Sun, G. Bebis, and R. Miller, "On-road vehicle detection: a review," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, no. 5, pp. 694 –711, may 2006.

[9] Hsu-Yung Cheng, Chih-Chia Weng, and Yi-Ying Chen, "Vehicle detection in aerial surveillance using dynamic bayesian networks," *Image Processing, IEEE Transactions on*, vol. 21, no. 4, pp. 2152 –2159, april 2012.

[10] Luo-Wei Tsai, Jun-Wei Hsieh, and Kuo-Chin Fan, "Vehicle detection using normalized color and edge map," *Image Processing, IEEE Transactions on*, vol. 16, no. 3, pp. 850 –864, march 2007.

[11] H. Jianming, M. Qiang, W. Qi, Z. Jiajie, and Z. Yi, "Traffic congestion identification based on image processing," *Intelligent Transport Systems, IET*, vol. 6, no. 2, pp. 153 –160, june 2012.

[12] F. Wang, "A knowledge-based vision system for detecting land changes at urban fringes," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 31, no. 1, pp. 136 –145, jan 1993.

[13] Sotirios Gyftakis, Peggy Agouris, and Anthony Stefanidis, "Image-based change detection of areal objects using differential snakes," in *Proceedings of the 13th annual ACM international workshop on Geographic information systems*, New York, NY, USA, 2005, GIS '05, pp. 135–142, ACM.

[14] Minghui Tian, Shouhong Wan, and Lihua Yue, "A novel approach for change detection in remote sensing image based on saliency map," in *Proceedings of the Computer Graphics, Imaging and Visualisation*, Washington, DC, USA, 2007, CGIV '07, pp. 397–402, IEEE Computer Society.

[15] Aniruddha Bose and Kunal Ray, "Fast change detection," *Defence Science Journal*, vol. 61, no. 1, 2010.

[16] C H Anderson, P J Burt, and G S Van Der Wal, "Change detection and tracking using pyramid transform techniques," *SPIE Conference on Intelligent Robotics and Computer Vision*, vol. 2, pp. 283–310, 1985.

Fig. 2: Experiment 1. The left most column shows a background frame on the top and its result after change detection at the bottom. The top row from second to the right shows a sequence of input frames. The bottom row shows its corresponding resultant frames with change detected in white.

Fig. 3: Experiment 2. The left most column shows a background frame on the top and its result after change detection at the bottom. The top row from second to the right shows a sequence of input frames of a CCTV photage. The bottom row shows its corresponding resultant frames with change detected in white.

Fig. 4: Experiment 3. The left most column shows a background frame on the top and its result after change detection at the bottom. The top row from second to the right shows a sequence of input frames of a CCTV photage. The bottom row shows its corresponding resultant frames with change detected in white.

Fig. 5: Experiment 4. The left most column shows a background frame on the top and its result after change detection at the bottom. The top row from second to the right shows a sequence of input frames of a CCTV photage. The bottom row shows its corresponding resultant frames with change detected in white.

Fig. 6: Experiment 5. The left most column shows a background frame on the top and its result after change detection at the bottom. The top row from second to the right shows a sequence of input frames of a CCTV photage. The bottom row shows its corresponding resultant frames with change detected in white.

[17] James R. Bergen, P. Anandan, Th J. Hanna, and Rajesh Hingorani, "Hierarchical model-based motion estimation," 1992, pp. 237–252, Springer-Verlag.

[18] Chris Stauffer and W. Eric L. Grimson, "Learning patterns of activity using real-time tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 747–757, Aug. 2000.

[19] J.M. McHugh, J. Konrad, V. Saligrama, and P.-M. Jodoin, "Foreground-adaptive background subtraction," *Signal Processing Letters, IEEE*, vol. 16, no. 5, pp. 390 –393, may 2009.

[20] R. B. Fisher, "Change detection in color images," 1999.

[21] A. Cavallaro and T. Ebrahimi, "Change detection based on color edges," in *Circuits and Systems, 2001. ISCAS 2001. The 2001 IEEE International Symposium on*, may 2001, vol. 2, pp. 141 –144 vol. 2.

[22] R. Malik, S. Nazir, and J. Khurshid, "Colour based road sign detection and extraction from still images," in *Cybernetic Intelligent Systems (CIS), 2010 IEEE 9th International Conference on*, sept. 2010, pp. 1 –6.

[23] Ismail Haritaoglu, Davis Harwood, and Larry S. David, "W4: Real-time surveillance of people and their activities," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 809–830, Aug. 2000.

[24] Tao Zhao, Manoj Aggarwal, Thomas Germano, Ian Roth, Alexandar Knowles, Rakesh Kumar, Harpreet Sawhney, and Supun Samarasekera, "Toward a sentient environment: real-time wide area multiple human tracking with identities," *Mach. Vision Appl.*, vol. 19, no. 5-6, pp. 301–314, Sept. 2008.

[25] Pau-Choo Chung and Chin-De Liu, "A daily behavior enabled hidden markov model for human behavior understanding," *Pattern Recogn.*, vol. 41, no. 5, pp. 1589–1597, May 2008.

[26] Keming Chen, Chunlei Huo, Jian Cheng, Zhixin Zhou, and Hanqing Lu, "Change detection based on adaptive markov random fields," in *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, dec. 2008, pp. 1 –4.

[27] Chin-De Liu, Yi-Nung Chung, and Pau-Choo Chung, "An interaction-embedded hmm framework for human behavior understanding: with nursing environments as examples," *Trans. Info. Tech. Biomed.*, vol. 14, no. 5, pp. 1236–1246, Sept. 2010.

[28] Mio Nishiyama and Tadashi Shibata, "Normalized scoring of hidden markov models by on-line learning and its application to gesture-sequence perception," in *Proceedings of the 16th IEEE international conference on Image processing*, Piscataway, NJ, USA, 2009, ICIP'09, pp. 3529–3532, IEEE Press.

[29] Lu Wang and Nelson H. C. Yung, "Extraction of moving objects from their background based on multiple adaptive thresholds and boundary evaluation," *Trans. Intell. Transport. Sys.*, vol. 11, no. 1, pp. 40–51, Mar. 2010.

[30] Yassine Benabbas, Nacim Ihaddadene, and Chaabane Djeraba, "Motion pattern extraction and event detection for automatic visual surveillance," *J. Image Video Process.*, vol. 2011, pp. 7:1–7:15, Jan. 2011.

[31] Bin Lai, Deng Yi Zhang, Zhi Yong Yuan, and Jian Hui Zhao, "Crowd segmentation from a static camera," in *Proceedings of the 4th international conference on Intelligent Computing: Advanced Intelligent Computing Theories and Applications - with Aspects of Theoretical and Methodological Issues*, Berlin, Heidelberg, 2008, ICIC '08, pp. 1134–1140, Springer-Verlag.

[32] R.J. Radke, S. Andra, O. Al-Kofahi, and B. Roysam, "Image change detection algorithms: a systematic survey," *Image Processing, IEEE Transactions on*, vol. 14, no. 3, pp. 294 –307, march 2005.

[33] Mohammad Imroze Khan, Bibhudendra Acharya, and Shrish Verm, "Comparison between different illumination independent change detection techniques," in *Proceedings of the 2011 International Conference on Communication, Computing & Security*, New York, NY, USA, 2011, ICCCS '11, pp. 347–350, ACM.

[34] Ying Wu and Ting Yu, "A field model for human detection and tracking," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, no. 5, pp. 753 –765, may 2006.

[35] F. Pop, "Distributed algorithm for change detection in satellite images for grid environments," in *Parallel and Distributed Computing, 2007. ISPDC '07. Sixth International Symposium on*, july 2007, p. 41.

APPENDIX
SOURCE CODE OF PROPOSED ALGORITHM

main()

```
int main(int argc, char* argv[])
{
/* Initializations, trainging part, computing covariance and mean are not given here */

if(rank==0){
        j=0;
        start = clock();
}

while(1){
 if(rank==0){
  hasFrame = 1;
  testFrame = cvQueryFrame( capture );
  if(!testFrame){
        hasFrame = 0;
  }
 }

 if(MPI_Bcast( &hasFrame, 1, MPI_INT, 0, MPI_COMM_WORLD) != MPI_SUCCESS)
        return EXIT_FAILURE;

 if(!hasFrame)
  break;

 for(i=0;i<frHeight/size;++i){
  if(rank==0){
         t = i *  blockSize;
         memcpy(rawTestingRows, testFrame->imageData+t, blockSize);
   }

  if(MPI_Scatter(rawTestingRows,rowSize,MPI_CHAR,rawTestRow,rowSize,MPI_CHAR,0,MPI_COMM_WORLD)!=MPI_SUCCESS)
         return EXIT_FAILURE;

  cvSetData(aTestFrame, rawTestRow, rowSize);

  resultFrame = computeMahalaNobis(aTestFrame, m, c, inputs.threshold, inputs.outputMode, 1, frWidth,(i*size)+rank);

  memcpy(rawTestRow, resultFrame->imageData, rowSize);
  cvReleaseImage(&resultFrame);

  if(MPI_Gather(rawTestRow, rowSize, MPI_CHAR, rawTestingRows, rowSize, MPI_CHAR, 0, MPI_COMM_WORLD)!= MPI_SUCCESS)
   return EXIT_FAILURE;

  if(rank==0)
   memcpy(finalResultFrame->imageData+t, rawTestingRows, blockSize);
 }

 if(rank==0){
        ++j;
        cvWriteFrame( writer, finalResultFrame);
 }
}

// Compute the execution time
if(rank==0){
 cvReleaseVideoWriter(&writer);
 end = clock();
 elapsed = (end - start) / (double)CLOCKS_PER_SEC;
 printf("computation time (seconds): %f\n", elapsed);
 printf("total number of frames tested: %d\n", j-1);
}
}
```

computeMahalaNobis()

```
IplImage* computeMahalaNobis(IplImage* testFrame, mean** m, cov** c, float threshold,  int outputMode,
                                                                 int frHeight, int frWidth, int currentRow)
{
IplImage* retval;
int i,j;
float d;
CvScalar p;
mean temp;
cov temp_cov;
```

```
retval = cvCreateImage(cvSize(testFrame->width,testFrame->height), testFrame->depth, testFrame->nChannels);
for(i=0;i<frHeight;++i){
 for(j=0;j<frWidth;j++){
  p = cvGet2D(testFrame,i,j);
        i+=currentRow;
        temp[0] = p.val[0]-m[i][j][0];
        temp[1] = p.val[1]-m[i][j][1];
        temp[2] = p.val[2]-m[i][j][2];

        inverseCov(c[i][j], temp_cov);

        d = mahalaNobis(temp, temp_cov);

        d = sqrt(d);

        if(!finite(d)){
                p.val[0] = p.val[1] = p.val[2] = 0;
        }else{
         if(!(d>threshold))
          p.val[0] = p.val[1] = p.val[2] = 0;
         else{
         if(outputMode==0){
                p.val[0] = p.val[1] = p.val[2] = 255;
                }
         }
        }
  i-=currentRow;
  cvSet2D(retval, i, j, p);
  }
 }
 return retval;
}
```