© October 2017 | IJIRT | Volume 4 Issue 5 | ISSN: 2349-6002

Implementation of Real Time Moving Object Detection for Video Systems

Kiran Chaudhari[1], Santhosh Banoth[2],
[1]*Student, Wainganga College of Engineering & Management*
[2]*Faculty, Wainganga College of Engineering & Management*

Abstract-**In every real time object detection video system, pre-processing step includes moving object detection algorithm which identifies (extract) useful information of moving objects present in a video. Most algorithms of moving object detection require large memory space for storage of background related information, therefore the implementation of such algorithms becomes a difficult task as there are limited resources for embedded systems. Therefore, to overcome this limitation, in this paper we present an algorithm which optimizes memory use along with increasing speed and therefore performance and reliability of moving object detection scheme for video systems. The scheme being modified from original clustering-based moving object detection algorithm and has coded in C sharp is presented here. Results of the same were compared with the original clustering-based moving object detection and analyzed thoroughly on qualitative and quantitative basis.**

The experimental results revealed that there is 11.66% reduction in memory requirement, hence speed has increased by 2% and therefore performance and reliability has increased by 4%, as compared to original without affecting accuracy and robustness.

Index Terms- **Real Time Moving Object Detection, Video Surveillance System, Cluster Based Algorithm.**

I. INTRODUCTION

In today's world, due to terrorist activities and other general social problems, the design of an real time object detection for video systems has become an important aspect of research and development in the field of automation of video surveillance systems for safety and security. These are also motivated by the constant increase in the number of cameras which naturally demands elimination of the human interaction within the video monitoring systems. Typically, the first step in any automated video surveillance system is the detection of moving objects, the outputs of which are used in the further processing steps. Thus, the efficiency and performance of the complete object detection video system solely depends on the effectiveness of the moving object detection algorithm.

Therefore, over the time a number of moving object detection algorithms have been proposed - each trying to compete their counterpart in terms of performance, reliability and speed. In addition to this, the algorithm should also be computationally efficient. The objective of the moving object detection algorithm is to identify the set of pixels that are significantly different from the previous image of the sequence, for sequence of images taken from the scene at different time interval[1]. There are a number of algorithms present in the literature for moving object detection. The most simplest approach used for moving object detection is based on background subtraction methods, where a background model is first built using images from a sequence, which is then used for the purpose of segmentation (to find the moving objects) by subtracting the current frame pixel-by-pixel from the build background model. Thus, the accuracy of moving object detection process depends on how well the background is modeled. Researchers have reported several moving object detection methods that are closely related to background subtraction e.g. change vector analysis [3]-[5], image rationing [6], and frame differencing using sub-sampled gradient images [7].

The simplicity of background subtraction based approaches comes at the cost of moving object detection quality and these approaches are unlikely to outperform the more advanced algorithms proposed for real-world surveillance applications such as predictive models [8]-[12], adaptive neural network [13], and shading models [14]-[16]. A comprehensive description and comparative analysis of these methods has been presented by Radke et al. [1].

Even for stationary background situations, there may be changes like light intensity changes in day time which must be a part of the stationary background. To address this problem, the researchers have designed adaptive background subtraction techniques [17]-[19] for moving object detection using Gaussian Density Function which are capable of handling light intensity/illumination changes in a scene with stationary background scenarios. But the backgrounds in the real-world scenes are pseudo-stationary and hence for real-world surveillance applications background cannot be assumed perfectly stationary. The pseudo-stationary background in the natural scenes are the consequences of the events like swaying branches of trees, moving tree leaves in windows of rooms, moving clouds, the ripples of water on a lake, or moving fan in the room. These small perturbations in the scenes are not desirable and should be incorporated into background. The statistical background modeling scheme using a single Gaussian is not capable of correctly modeling such pseudo-stationary backgrounds. Realizing this, Stauffer and Grimson [20] proposed an Adaptive Background Mixture Models using mixture of Gaussians to model such pseudo-stationary backgrounds.

However, maintaining these mixtures for every pixel in the frame is computational expensive and results in low frame rates. To overcome this limitation Butler et al. [21] proposed a new approach, similar to that of Stauffer and Grimson [20]. The processing, in this approach, is performed on YCrCb video data format which still requires many computations and a large amount of memory for storing the background models. In any automated video surveillance system, moving object detection is one of the important component which acts as a pre-processing step and on its outputs many other processing modules are dependent. Therefore, in addition to being accurate and robust, a moving object detection technique must also be efficient in terms of computational resources and memory requirement. This is because many other complex algorithms of an automated video surveillance system also runs on the same embedded or FPGA platform if standalone implementation is desired which is actually the case required for any automated video surveillance system. Thus, in order to address the problem of reducing the computational complexity, Chutani and Chaudhury [22] proposed a block-based clustering scheme with a very low complexity for moving object detection. On one hand this scheme is robust enough for handling pseudo-stationary nature of background, and on the other it significantly lowers the computational complexity and is well suited for designing standalone systems. However, the algorithm is still not much efficient in terms of memory requirements.

Therefore, to optimize the memory requirements of the clustering based moving object detection algorithm, we have presented a memory efficient moving object detection algorithm which also will eventually result in increase in performance and reliability. The rest of the paper is organized as follows: in the next section, we detail the original clustering based moving object detection algorithm.

In third section, we present the memory analysis of original clustering based moving object detection algorithm and certain important observations based on which the memory efficient algorithm is designed. The designed memory efficient moving object detection algorithm with its pseudo-code is described in the fourth section. Verification results and memory reduction results are reported in the fifth section. Finally, we conclude this paper with a short summary.

II. CONCEPT OF ORIGINAL CLUSTERING-BASED MOVING OBJECT DETECTION ALGORITHM

In this section, the clustering based moving object detection scheme is briefly described. For more detailed description we refer to [22] and [21]. Clustering based moving object detection uses a block-based similarity computation scheme.

To start with, each incoming video frame is partitioned into 4x4 pixel blocks.

Each 4x4 pixel block is modeled by a group of four clusters where each cluster consists of a block centroid (in RGB) and a frame number which updated the cluster most recently. Optionally, for each block there may be a motion flag field. The group of four clusters is necessary to correctly model the pseudo-stationary background, as a single cluster is incapable of modeling multiple modes that can be present in pseudo-stationary backgrounds.

The group size is selected as four because it has been reported by Chutani and Chaudhury [22] that four clusters per group yield a good balance between accuracy and computational complexity. The basic computational scheme is shown in Fig. 1 and the pseudo-code is shown in Fig. 2. The sequence of steps for moving object detection using original clustering-based scheme is given below.

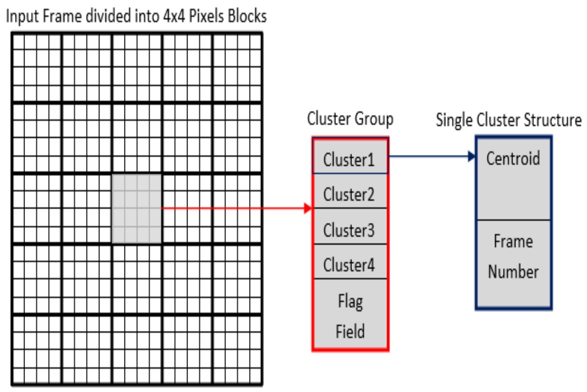


Fig.1. *Clustering-based Moving Object Detection Scheme.*

Block Centroid Computation: Each incoming frame is partitioned into 4x4 blocks. For each block, the block centroid for grayscale image is computed by taking the average intensity value of the 16 pixels of that block. The block centroid is of 8-bits.

Cluster Group Initialization: During the initial four frames, initialization is performed. In the first frame, the first cluster of each block is initialized with its centroid set to the block centroid of corresponding block of the first frame and its frame number is set to 1. In the second frame, the second cluster of each block is initialized with its centroid set to block centroid of corresponding block of the second frame and its frame number is set to 2. In the third frame, the third cluster of each block is initialized with its centroid set to the block centroid of corresponding block of the third frame and its frame number is set to 3. Similarly, fourth frame is initialized. In this way, all four clusters of the cluster group are initialized.

Cluster Matching: After initialization, the next step for moving object detection in incoming frames is to compare each of the incoming blocks against the corresponding cluster group. The goal is to find a matching cluster within the cluster group. For finding a matching cluster, for each cluster in the cluster group, the difference between its centroid and the incoming current block centroid is computed. The cluster with minimum centroid difference below the user defined threshold is considered as a matching cluster. In order to simplify this computation, Manhattan distance (sum of absolute differences) is used which avoids the overheads of multiplication in difference computation [21]. Eliminating multiplications is very beneficial in terms of reducing computational complexity of the algorithm as multiplications are costly in hardware.

Cluster Update: If, for a given block, a matching cluster is found within the cluster group, then the matching cluster is updated. The frame number of the matching cluster is replaced by the current frame number and the centroid of the matching cluster is replaced by the average value of matching cluster centroid and the incoming current block centroid.

Cluster Replace: If, for a given block, no matching cluster could be found within the group, then the oldest cluster which has not been updated for the longest period of time (cluster with minimum frame number) is deleted and a new cluster is created having the current block centroid as its centroid and the current frame number as its frame number.

Classification: For a given block, if no matching cluster is found and the oldest cluster is replaced, then it implies that the incoming current block is not matching with the background models and it is marked as moving object detected block by setting the motion flag field of the block to =1‘. If a matching cluster is found within the cluster group and the matching cluster is updated, then the incoming current block belongs to the background and therefore, the motion flag field of the block is set to =0‘ (i.e. no moving object detected).

I. Algorithm Analysis And Observations

In this algorithm, there are two main parameters (i.e. Centroid and Frame Number) associated with each 4x4 pixels block for storing the background related information. For grayscale images, the Centroid value is of 8-bits and Frame Number is stored using 16-bits data format. Therefore, for each 4x4 pixels block it would require 24-bits to store one background model information.

As there are four background models used in the algorithm, it requires 96-bits memory space for storing background information for each 4x4 pixel block. For PAL (720x576) resolution video, the total number of 4x4 pixels blocks are 25920 (= 720/4 * 576/4). Therefore, total memory space required to store the background information for PAL resolution videos is 25920x100 bits = 2592000 bits = 2530 Kbits = 2.373 Mbits. For achieving real-time performance, it is very difficult to implement this algorithm on limited resources standalone embedded platforms like low-cost low-end FPGAs having very less on-chip memory (Block RAMs). Therefore, for this reason, further emphasis needs to be given to the minimization of memory requirements of clustering-based moving object detection scheme proposed by [22] without compromising on accuracy and robustness of moving object detection.

Accordingly, the clustering-based moving object detection algorithm was re-looked at and the memory analysis was carried out. The memory size required for storing the background information is directly proportional to the size of the cluster group, block size, and video frame size.

Therefore, for given standard PAL (720x576) size color video streams, memory size can be reduced either by reducing the number of clusters from four to three or by increasing the 4x4 pixels block size to a larger block size. But Chutani and Chaudhury [22] had chosen to select a cluster size of 4 clusters and block size of 4x4 pixels because empirically they had found that these values yielded a good balance between accuracy and computational complexity.

In the first case, if the number of clusters is reduced to three then the algorithm‘s background model used to capture pseudo-stationary changes/movements becomes weak and the algorithm becomes more sensitive to pseudo-stationary background changes, resulting in false relevant moving object detection outputs for pseudo-stationary background changes. In the second case, for larger block sizes, the system becomes less sensitive to relevant motions in smaller areas in a video scene. Therefore, none of the above two techniques can be used to reduce memory size as the objective is to reduce memory size without compromising on the accuracy and the robustness of moving object detection. For this reason, we re-analyzed the original clustering based moving object detection algorithm and the following observations were resulted.

The important observation is that during the cluster updating (in case a matching cluster is found) or cluster replacement (in case no matching cluster is found) the actual time or frame number when the cluster is updated or replaced is not necessarily required. During cluster update, the matching cluster label is required (i.e. first or second or third or fourth), not the actual value of the Frame Number.

In case of cluster replacement, the oldest cluster label (which has not been updated for the longest period of time) is required. The Frame Number is stored and used to get this required information only. This implies that there is no need of storing the complete Frame Number value. An index value is sufficient to maintain the cluster updating or replacement history, which implies that it is the newest cluster (most recently updated or replaced), the second newest cluster, the second oldest cluster, or the oldest cluster (which has not been updated for the longest period of time).

As there are four clusters, therefore, a 2-bit index value is sufficient to record this information (i.e. the newest cluster, the second newest cluster, the second oldest cluster, or the oldest cluster). This reduces the 16-bit wide Frame Number memory to 2-bit wide memory and results in significant reduction in memory requirements.

Based on this observation and associated modifications in the original clustering-based moving object detection scheme, a memory efficient moving object detection scheme is proposed and detailed in the next section.

Proposed Moving Object Detection Algorithm

There are many approaches for motion detection of objects in a continuous video stream. The concept more or less remains same of comparing the current video frame from previous frames or with something that we called background. In this algorithm we use the AForge.NET framework.

The application supports the following types of video sources:

- AVI files(using Video for Windows);
- Updating JPEG from internet cameras;
- MJPEG(motion JPEG) streams from different internet cameras;
- Local capture device(USB cameras or other capture devices)

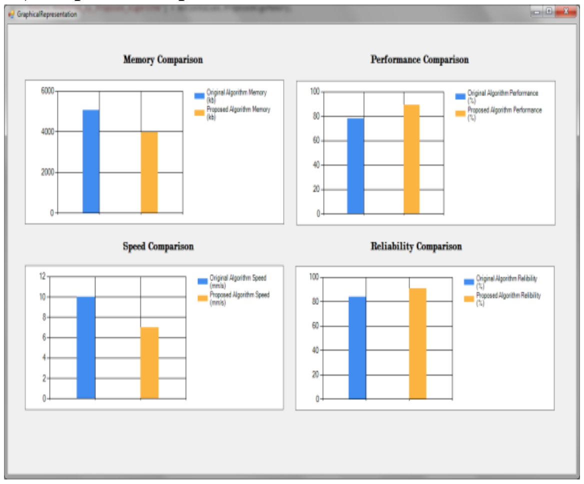
Assume that we have an original 24bpp RGB image called current frame(image), a grayscale copy of it(current frame) and previous video frame also gray scaled(background frame). To find the regions where these two frames are differing a bit, we use difference and threshold filters.

It is possible to count the pixels, and if the amount of it will be greater than a predefined alarm level we can signal about a motion event. As most cameras produce a noisy image, we will get motion in such

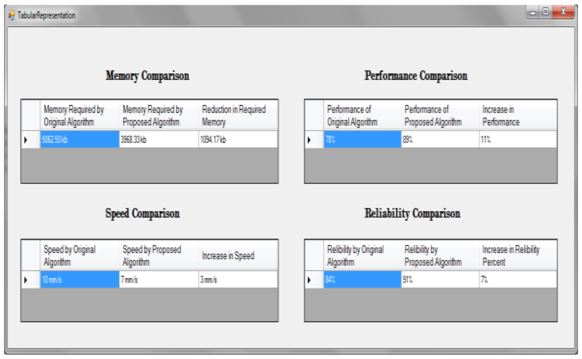
places, where there is no motion at all. To remove random noisy pixels, we use erosion filter.
First frame of the video sequence is used as background frame. Then the current frame is compared with background frame. Our approach is to "move" the background frame to the current frame on the specified amount (we have use level 1 per frame).
We move the background frame slightly in the direction of the current frame; we are changing colors of pixels in the background frame by one level per frame.
We apply pixel ate filter to the current frame and to the background before further processing as we can see a small numbers on the objects.
The idea of morph filter is to preserve specified percentage of the source filter and to add missing percentage from overlay image. We know that a binary image contains a difference between current frame and the background frame. To add motion alarm feature, we need to calculate the amount of white pixels on this difference image. Looking at picture2, we can see that objects are highlighted with a curve, which represents the moving object's boundary. By using blob counter we can get the number of objects, their position and the dimension on a binary image. For simplicity in the blob counting approach we can accumulate not the white pixels count, but the area of each detected object. Then if the computed amount of changes is greater than a predefined value, an alarm event occurred. There are different ways to process motion alarm event, the most useful one is video saving on motion detection.

III. RESULT AND CONCLUSION

A) Graphical Representation for results



B) Tabular Representation for results


TabularRepresentation

Memory Comparison

	Memory Required by Original Algorithm	Memory Required by Proposed Algorithm	Reduction in Required Memory
▸	5062.50 kb	3968.33 kb	1094.17 kb

Performance Comparison

	Performance of Original Algorithm	Performance of Proposed Algorithm	Increase in Performance
▸	78%	89%	11%

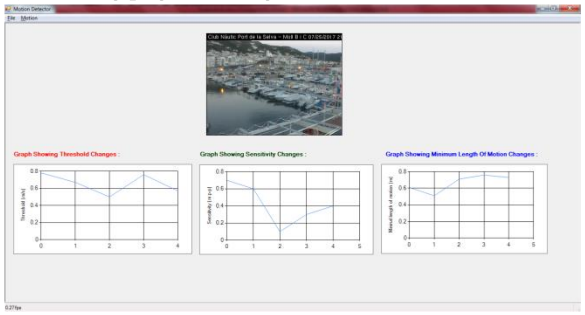
Speed Comparison

	Speed by Original Algorithm	Speed by Proposed Algorithm	Increase in Speed
▸	10 mm/s	7 mm/s	3 mm/s

Reliability Comparison

	Relibility by Original Algorithm	Relibility by Proposed Algorithm	Increase in Relibility Percent
▸	84%	91%	7%

C) Landing page after sign in


Motion Detector
File Motion
Graph Showing Threshold Changes :
Graph Showing Sensitivity Changes :
Graph Showing Minimum Length Of Motion Changes :

In the first, we will get an image with white pixels on the place where the current frame is different from the previous frame on the specified threshold value. If the amount of pixels will be greater than a predefined alarm level we can signal a motion event. Due to erosion filter, we will get only the regions of the actual motion.
In this research article, a memory optimized moving object detection scheme, useful for designing moving object detection systems, has been presented. The emphasis has been given on optimizing memory requirements for storing background related information. It has been coded using AForge.NET framework. The moving object detection results of proposed memory efficient scheme were qualitatively as well as quantitatively analyzed and compared with the original clustering-based moving object detection algorithm. The experimental results revealed that there is 11.66% reduction in memory requirement, hence speed has increased by 2% and therefore performance and reliability has increased by 4%, as compared to original without affecting accuracy and robustness.

REFERENCES

[1] R.J. Radke, S. Andra, O.A. Kofahi, and B. Roysam, Image Change Detection Algorithms: A Systematic Survey, IEEE Transactions on Image Processing, Vol. 14, No. 3, pp. 294-307, 2005.

[2] S. Singh, SumeetSaurav, and Chandra Shekhar, Moving Object Detection Scheme for Automated Video Surveillance Systems, I.J. Image, Graphics and Signal Processing, Vol. 7, pp. 49-58, 2016.

[3] L. Bruzzone and D.F. Prieto, Automatic Analysis of the Difference Image for Unsupervised Change Detection, IEEE Transaction on Geosciences and Remote Sensing, Vol. 38, No. 3, pp. 1171–1182, 2000.

[4] J.E. Colwell and F.P. Weber, Forest Change Detection, In Proceedings: 15th International Symposium on Remote Sensing of the Environment, pp. 839-852, 1981.

[5] W.A. Malila, Change Vector Analysis: An Approach for Detecting Forest Changes with Landsat, In Proceedings: Symposium on Machine Processing of Remotely Sensed Data, pp. 326-336, 1980.

[6] A. Singh, Review Article: Digital Change Detection Techniques using Remotely-sensed Data, International Journal of Remote Sensing, Vol. 10, No. 6, pp. 989-1003, 1989.

[7] L.D. Stefano, S. Mattoccia, and M. Mola, A Change-detection Algorithm based on Structure and Color, In Proceedings: IEEE Conference on Advanced Video and Signal-Based Surveillance, pp. 252-259, 2003.

[8] Y.Z. Hsu, H.H. Nagel, and G. Rekers, New Likelihood Test Methods for Change Detection in Image Sequences, Computer Vision, Graphics, Image Processing, Vol. 26, No. 1, pp. 73-106, 1984.

[9] K. Skifstad and R. Jain, Illumination Independent Change Detection for Real World Image Sequences, Computer Vision, Graphics, Image Processing, Vol. 46, No. 3, pp. 387-399, 1989.

[10] A.S. Elfishawy, S.B. Kesler, and A.S. Abutaleb, Adaptive Algorithms for Change Detection in Image Sequence, Signal Processing, Vol. 23, No. 2, pp. 179-191, 1991.

[11] Z.S. Jain and Y.A. Chau, Optimum Multisensor Data Fusion for Image Change Detection, IEEE Transaction on System, Man and Cybernetics, Vol. 25, No. 9, pp. 1340-1347, 1995.

[12] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers, Wallflower: Principles and Practice of Background Maintenance, in Proceedings: Seventh International Conference on Computer Vision, pp. 255-261, 1999.

[13] C. Clifton, Change Detection in Overhead Imagery using Neural Networks, Applied Intelligence, Vol. 18, pp. 215-234, 2003.

[14] E. Durucan and T. Ebrahimi, Change Detection and Background Extraction by Linear Algebra, In Proceedings: IEEE, Vol. 89, No. 10, pp. 1368-1381, 2001.

[15] L. Li and M.K.H. Leung, Integrating Intensity and Texture Differences for Robust Change Detection, IEEE Transaction on Image Processing, Vol. 11, No. 2, pp. 105-112, 2002.

[16] S.C. Liu, C.W. Fu, and S. Chang, Statistical Change Detection with Moments under Time-Varying Illumination,‖ IEEE Transaction on Image Processing, Vol. 7, No. 9, pp. 1258-1268, 1998.

[17] A. Cavallaro and T. Ebrahimi, Video Object Extraction based on Adaptive Background and Statistical Change Detection, In Proceedings: SPIE Visual Communications and Image Processing, pp. 465-475, 2001.

[18] S. Huwer and H. Niemann, Adaptive Change Detection for Real-Time Surveillance Applications, In Proceedings: Third IEEE International Workshop on Visual Surveillance, pp. 37-46, 2000.

[19] T. Kanade, R.T. Collins, A.J. Lipton, P. Burt, and L. Wixson, Advances in Cooperative Multi-Sensor Video Surveillance, In Proceedings: DARPA Image Understanding Workshop, pp. 3-24, 1998.

[20] C. Stauffer and W.E.L. Grimson, Learning Patterns of Activity using Real-Time Tracking, IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol. 22, No. 8, pp. 747-757, 2000.

[21] D.E. Butler, V.M. Bove, and S. Sridharan, Real-Time Adaptive Foreground/Background Segmentation, EURASIP Journal on Applied Signal Processing, Vol. 2005, pp. 2292-2304, 2005.

[22] E.R. Chutani and S. Chaudhury, Video Trans-Coding in Smart Camera for Ubiquitous Multimedia Environment, In Proceedings: International Symposium on Ubiquitous Multimedia Computing, pp.185–189, 2008.