I.J. Image, Graphics and Signal Processing, **2016, 7, 49-58**
Published Online July 2016 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijigsp.2016.07.06

Moving Object Detection Scheme for Automated Video Surveillance Systems

Sanjay Singh, Sumeet Saurav, Chandra Shekhar
CSIR - Central Electronics Engineering Research Institute (CSIR-CEERI) Pilani - 333031, Rajasthan, India
E-mail: sanjay.csirceeri@gmail.com

Anil Vohra
Electronic Science Department, Kurukshetra University, Kurukshetra - 136119, Haryana, India.

Abstract—In every automated video surveillance system, moving object detection is an important pre-processing step leading to the extraction of useful information regarding moving objects present in a video scene. Most of the moving object detection algorithms require large memory space for storage of background related information which makes their implementation a difficult task on embedded platforms which are typically constrained by limited resources. Therefore, in order to overcome this limitation, in this paper we present a memory optimized moving object detection scheme for automated video surveillance systems with an objective to facilitate its implementation on standalone embedded platforms. The presented scheme is a modified version of the original clustering-based moving object detection algorithm and has been coded using C/C++ in the Microsoft Visual Studio IDE. The moving object detection results of the proposed memory efficient scheme were qualitatively and quantitatively analyzed and compared with the original clustering-based moving object detection algorithm. The experimental results revealed that there is 58.33% reduction in memory requirements in case of the presented memory efficient moving object detection scheme for storing background related information without any loss in accuracy and robustness as compared to the original clustering based scheme.

Index Terms—Moving Object Detection, Automated Video Surveillance System, Smart Camera System.

I. Introduction

In today's world of automation, the design of an automated video surveillance system has become one of the most important field of research in the computer vision community and is getting increasingly attention day by day due to increasing level of terrorist activities, safety and security, and other general social problems. 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 automated video surveillance 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.

For a given sequence of images taken from the same scene at several different time intervals, the objective of the moving object detection algorithm is to identify the set of pixels that are significantly different to the last image of the sequence [1]. Thus, the purpose of the moving object detection algorithm is to classify the pixels of every frame of the image sequence into two classes: the background (corresponding to pixels belonging to the static scene) and the foreground (corresponding to pixels belonging to a moving objects [2]. A key requirement of any moving object detection algorithm is that it must have ability to discriminate the moving objects from the background as accurately as possible, without being too sensitive to the sizes and velocities of the objects, or to the changing conditions of the static scene. In addition to this, the algorithm should also be computationally efficient.

There are a number of algorithms reported 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 background 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 perturbation in the scenes are not desirable and should be incorporated into background. The statistical background modelling 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 suitable for implementation on limited resources standalone embedded platforms such as low-cost low-end embedded FPGA board for achieving real-time performance.

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. 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.

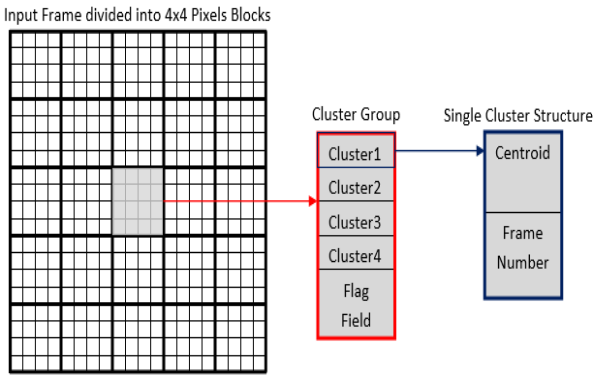


Fig.1. Clustering-based Moving Object Detection Scheme.

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.

```
//For each 4x4 pixels block of every incoming frame the following steps are performed
//Compute block centroid
        Block_Centroid = (∑_{i=1}^{4} ∑_{j=1}^{4} pixel(i,j))/16
//Initialization: For Frame Numbers 1, 2, 3, and 4 only
        ClusterCentroid_Cluster_No = Block_Centroid
        ClusterFrameNumber_Cluster_No = Present Frame Number
        Where Cluster_No is 1 for first frame, 2 for second frame, 3 for third frame, and 4 for
        fourth frame.
//Motion Detection: For all Frame Numbers ≥ 5
        //Find Matching Cluster in cluster group of four clusters
                //Compute Difference
                Difference_Cluster_No = ClusterCentroid_Cluster_No - Block_Centroid
                Where Cluster_No is 1, 2, 3, and 4
                // Find Minimum Difference
                MinimumDifference = Minimum(Difference_1, Difference_2,Difference_3, Difference_4)
                //Compare with threshold
                If (MinimumDifference ≤ Threshold)
                        Cluster corresponding to minimum difference is matching cluster
                Else
                        No matching cluster is found
        //Cluster Update: For matching cluster i.e If MinimumDifference ≤ Threshold
                ClusterCentroid_Matching_Cluster = (ClusterCentroid_Matching_Cluster + Block_Centroid)/2
                ClusterFrameNumber_Matching_Cluster = Present Frame Number
        //Cluster Replace: If no matching cluster found i.e If MinimumDifference >Threshold
                ClusterCentroid_Oldest_Cluster = Block_Centroid
                ClusterFrameNumber_Oldest_Cluster = Present Frame Number
        //Classification
                 If (MinimumDifference ≤Threshold)
                        Motion Flag (Current Block) = '0' (No Motion Detected)
                Else
                        Motion Flag (Current Block) = '1' (Motion Detected)
//Process Completed for Current Block
```

Fig.2. Pseudo-code for Clustering-based Moving Object Detection Scheme.

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. In the fourth frame, the last/fourth cluster of each block is initialized with its centroid set to the block centroid of corresponding block of the fourth frame and its frame number is set to 4. In this way, during initialization all the 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).

III. 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. Therefore, reducing cluster size or increasing block size will result in the degradation of accuracy and robustness of the clustering based moving object detection scheme. 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 background related information in the clustering-based moving object detection algorithm is stored and updated using two parameters viz. Centroid and Frame Number. Each Centroid memory location contains four Centroid values corresponding to four clusters/ background models which contain intensity information of pseudo-stationary background. Each Frame Number memory location stores four Frame Number values corresponding to four clusters / background models which are used to keep the record of Centroid value updation or replacement history i.e. for the particular 4x4 pixels block when (at what time or for what Frame Number) the cluster Centroid value is updated or replaced. Now, 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 cluster, 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.

IV. Proposed Memory Efficient Moving Object Detection Algorithm

In this section, we present a new memory efficient algorithm for moving object detection based on the observations discussed in previous section and associated modifications in the original clustering-based moving object detection scheme. The algorithm is proposed and designed with an aim to reduce the memory requirements without compromising on the robustness and accuracy of moving object detection. The block size and number of cluster/background models are kept same as in original algorithm (i.e. 4x4pixel block and 4 clusters). However, in this proposed scheme, each 4x4 pixel block is modeled by a group of four clusters where each cluster consists of a block Centroid and a 2-bit Index Value as shown in Fig. 3. The pseudo-code is shown in Fig. 4. The complete proposed memory efficient moving object detection algorithm is detailed below.

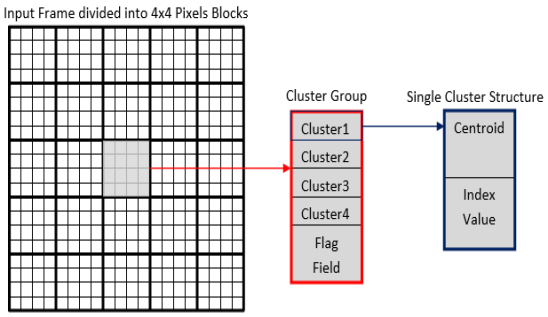


Fig.3. Proposed Memory Efficient Moving Object Detection Scheme.

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

During the initial four frames, **Cluster Group 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 index value is set to 3 as this is going to be the oldest value at the end of the Cluster Group Initialization process. In the second frame, the second cluster of each block is initialized with its centroid set to the block centroid of corresponding block of the second frame and its index value is set to 2 as this is going to be the second oldest value at the end of the Cluster Group Initialization process. 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 index value is set to 1 as this is going to be the second newest value at the end of the Cluster Group Initialization process. In the fourth frame, the last/fourth cluster of each block is initialized with its centroid set to the block centroid of corresponding block of the fourth frame and its index value is set to 0 as this is going to be the newest value at the end of the Cluster Group Initialization process. At the end of Cluster Group Initialization process, we will have four cluster for each 4x4 pixels block with Centroid and Index Value parameters.

After initialization (in initial four frames), for all subsequent frames the moving object detection is performed. During moving object detection either of the two processes (i.e. **Cluster Updating** process or **Cluster Replacement** process) is performed. Therefore, the objective is either to update the cluster node or to replace the cluster node.

If a matching cluster is found (i.e. the Minimum Centroid Difference is less than the threshold) within the cluster group then the **matching cluster is updated**. For this purpose, the Centroid value of the matching cluster is updated with the average value of the matching cluster Centroid value and incoming current Block Centroid value. The Index Value of matching cluster can be any of the four values i.e. 0, 1, 2, 3.

Consider the **first case of Cluster Updating** i.e. matching cluster Index Value is 0. In this case, the Index Values for all cluster will remain same as the newest cluster is updated and order of four clusters update/replacement history will remain same.

Consider the **second case of Cluster Updating** i.e. matching cluster Index Values is 1. In this case, the Index Value of matching cluster will be replace by 0 and also the cluster having Index Value 0 will be replaced by Index Value 1 to keep the proper update/replacement history order.

Consider the **third case of Cluster Updating** i.e. matching cluster Index Value is 2. In this case, the Index Value of matching cluster will be replaced by 0 and also the cluster having Index Value 0 and 1 will be replaced by Index Value 1 and 2 respectively to keep the proper update/replacement history order.

Consider the **last/fourth case of Cluster Updating** i.e. matching cluster Index Value 3. In this case, the Index Value of matching cluster will be replace by 0 and also the clusters having Index Value 0, 1, 2 will be replaced by Index Value 1, 2, ad 3 respectively to keep the proper

update/replacement history order.

If no matching cluster is found (i.e. the Minimum Centroid Difference is greater than the threshold) within the cluster group then **oldest cluster is replaced**. For this purpose, the oldest cluster which has not been updated for the longest period of time i.e. cluster having Index Value 3 is replaced. The Centroid of this cluster is set to incoming current Block Centroid value and its Index Value is set to 0 as it is replaced most recently. Also the cluster having Index Value 0, 1, and 2 will be replaced by Index Value 1, 2, and 3 respectively to keep the proper update/replacement history order.

Thus, at the end of any of the processes i.e. Cluster Group Initialization, Cluster Updating, or Cluster Replacing, the four clusters Index Values keep the record of their update/replacement history properly. This process is performed for all the blocks in every incoming frame and it keeps track of newest cluster (most recently updated or replaced), second newest cluster, second oldest cluster, or oldest cluster (which has not been updated for the longest period of the time) and require only 2-bit Index Value in place of 16-bit Frame Number value. The proposed novel scheme is thus successful in substantially reducing the memory requirement of original clustering based moving object detection scheme, without any loss of accuracy in moving object detection. The resulting reduction in memory size for different video resolution is presented in next section.

```
//For each 4x4 pixels block of every incoming frame the following steps are performed
//Compute block centroid
        Block_Centroid = (∑_{i=1}^{4} ∑_{j=1}^{4} pixel(i,j))/16
//Initialization: For Frame Numbers 1, 2, 3, and 4 only
        ClusterCentroid_Cluster_No = Block_Centroid
        ClusterIndexValue_Cluster_No = 4 - Cluster_No
        Where Cluster_No is 1 for first frame, 2 for second frame, 3 for third frame, and 4 for
fourth frame.
//Motion Detection: For all Frame Numbers ≥ 5
    //Find Matching Cluster in cluster group of four clusters
            //Compute Difference
            Difference_Cluster_No = ClusterCentroid_Cluster_No - Block_Centroid
            Where Cluster_No is 1, 2, 3, and 4
            // Find Minimum Difference
            MinimumDifference = Minimum(Difference1, Difference2,Difference3, Difference4)
            //Compare with threshold
            If (MinimumDifference ≤ Threshold)
                    Cluster corresponding to minimum difference is matching cluster
            Else
                    No matching cluster is found
    //Cluster Update: For matching cluster i.e. If MinimumDifference ≤ Threshold
            If ClusterIndexValue_Matching_Cluster = 0
               ClusterCentroid_Matching_Cluster = (ClusterCentroid_Matching_Cluster + Block_Centroid)/2
               ClusterIndexValue_Matching_Cluster = 0
            If ClusterIndexValue_Matching_Cluster = 1
               ClusterCentroid_Matching_Cluster = (ClusterCentroid_Matching_Cluster + Block_Centroid)/2
               ClusterIndexValue_Matching_Cluster = 0
               ClusterIndexValue_Cluster_with_Index_Value 0 = 1
```

```
                If ClusterIndexValue_Matching_Cluster = 2
                    ClusterCentroid_Matching_Cluster = (ClusterCentroid_Matching_Cluster + Block_Centroid)/2
                    ClusterIndexValue_Matching_Cluster = 0
                    ClusterIndexValue_Cluster_with_Index_Value 0 = 1
                    ClusterIndexValue_Cluster_with_Index_Value 1 = 2
                If ClusterIndexValue_Matching_Cluster = 3
                    ClusterCentroid_Matching_Cluster = (ClusterCentroid_Matching_Cluster + Block_Centroid)/2
                    ClusterIndexValue_Matching_Cluster = 0
                    ClusterIndexValue_Cluster_with_Index_Value 0 = 1
                    ClusterIndexValue_Cluster_with_Index_Value 1 = 2
                    ClusterIndexValue_Cluster_with_Index_Value 2 = 3
    //Cluster Replace: If no matching cluster found i.e. If MinimumDifference >Threshold
                    ClusterCentroid Cluster_with_Index_Value 3 = Block_Centroid
                    ClusterIndexValue Cluster_with_Index_Value 3 = 0
                    ClusterIndexValue_Cluster_with_Index_Value 0 = 1
                    ClusterIndexValue_Cluster_with_Index_Value 1 = 2
                    ClusterIndexValue_Cluster_with_Index_Value 2 = 3
    //Classification
        If (MinimumDifference ≤Threshold)
                Motion Flag (Current Block) = '0' (No Motion Detected)
        Else
                Motion Flag (Current Block) = '1' (Motion Detected)
//Process Completed for Current Block
```

Fig.4. Pseudo-code for Proposed Memory Efficient Moving Object Detection Scheme.

V. Verification Results of Proposed Moving Object Detection Scheme

Comparison of memory requirements by proposed memory efficient moving object detection algorithm and original clustering based algorithm [22] is summarized in Table 1 for different video resolutions. The percentage of reduction in memory requirements for all video resolutions is 58.33%.

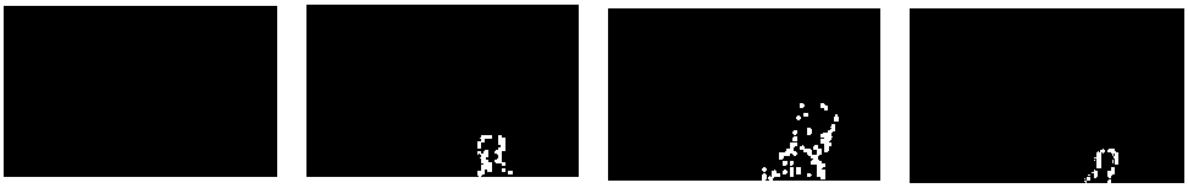
In order to reduce the memory requirement of moving object detection algorithm, without negatively impacting the quality of processed videos, it is important to accurately evaluate the proposed algorithm against original clustering-based moving object detection algorithm on video streams of different real-world scenarios. For this purpose, the original clustering based algorithm and proposed memory efficient algorithm have been programmed in C/C++ programming language. For running the code, a Dell Precision T3400 workstation (with Windows XP operating system, quad-core Intel® Core™2 Duo Processor with 2.93 GHz Operating Frequency, and 4GB RAM) was used. The Open Computer Vision (OpenCV) libraries were used in the code for reading video streams (either stored or coming from camera) and displaying moving object detection results. The effect of removing 16-bit Frame Number and using 2-bit Index Value was evaluated on test bench videos taken from surveillance cameras. The selected video streams have a resolution of 720 × 576 pixels (PAL size) and have five minute duration.

Fig. 5 and Fig. 6 visually compare the results for both the moving object detection algorithms (proposed and original) for different indoor and outdoor conditions with pseudo-stationary backgrounds. In Fig. 5, the moving fan in background is present, while in Fig. 6, the moving leaves of trees are present. The top row in both figures shows the original frames extracted from video streams. The second row shows the moving object detection results obtained from the original clustering based algorithm and the third row shows the moving object detection results obtained from the proposed memory efficient clustering based algorithm for respective frames. To compare the results, the pixel-by-pixel difference of second and third row images has been taken and it gives black images (fourth row). This indicates that the proposed memory efficient algorithm produces same moving object detection results as original clustering based moving object detection algorithm without any loss of accuracy and robustness but with significant reduction in memory requirement.

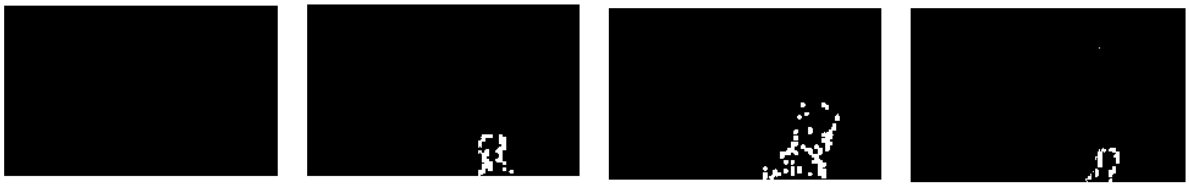
Table 1. Reduction in Memory Requirements for Different Video Resolutions.

Video Resolution	Memory Required by Original Algorithm [22]	Memory Required by Proposed Algorithm	Reduction in Required Memory Space by Proposed Algorithm
HD (1920 x 1080)	12150 Kb	5062.50 Kb	7087.50 Kb
PAL (720 x 576)	2430 Kb	1012.50 Kb	1417.50 Kb
NTSC (720 x 480)	2025 Kb	843.75 Kb	1181.25 Kb
VGA (640 x 480)	1800 Kb	750.00 Kb	1050.00 Kb
CIF (352 x 288)	594 Kb	247.50 Kb	346.50 Kb

(a) Input Video Frames



(b) Original Moving Object Detection Algorithm Output



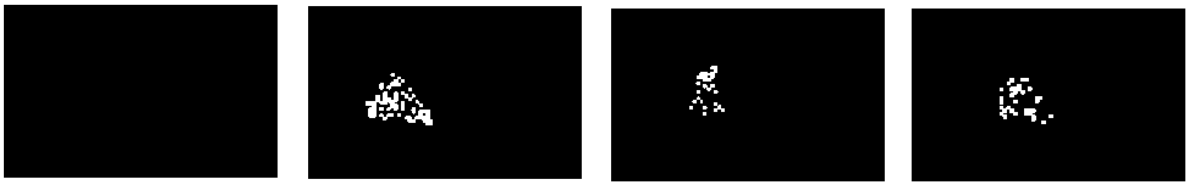
(c) Proposed Moving Object Detection Algorithm Output

(d) Difference of Original and Proposed Algorithm Outputs

Fig.5. Comparison of Results of the Proposed Memory Efficient Moving Object Detection Algorithm with the Original Clustering based Moving Object Detection Scheme for Indoor Scenario.

(a) Input Video Frames

(b) Original Moving Object Detection Algorithm Output



(c) Proposed Moving Object Detection Algorithm Output

(d) Difference of Original and Proposed Algorithm Outputs

Fig.6. Comparison of Results of the Proposed Memory Efficient Moving Object Detection Algorithm with the Original Clustering based Moving Object Detection Scheme for Outdoor Scenario.

For quantitative analysis, for every frame of each video stream, the mean square error (MSE) is calculated. MSE is a common measure of quality of video and is equivalent to other commonly used measures of quality. For example, the peak signal-to-noise ratio (PSNR) is equivalent to MSE [23]. Some researchers measure the number of false positives (FP) and false negatives (FN) whose sum is equivalent to the MSE [24]. MSE is defined as

$$MSE = \frac{1}{M * N} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} (I_{PROPOSED}(m,n) - I_{ORIGINAL}(m,n))^2 \quad (1)$$

In the above equation, $I_{ORIGINAL}$ (m, n) is the moving object detected binary output image produced by running the software (C/C++) implementation of original clustering based algorithm, while $I_{PROPOSED}$ (m, n) is the moving object detected binary output image produced by running the software (C/C++) implementation of proposed memory efficient clustering based algorithm. M is the number of columns in a video frame i.e. 720 and N is the number of rows in a video frame i.e. 576.

As the moving object detection outputs are binary images, therefore, the square of the difference between $I_{ORIGINAL}$ (m, n) and $I_{PROPOSED}$ (m, n) has only two possible values: "1" if the pixel has different values in $I_{ORIGINAL}$ and $I_{PROPOSED}$ and "0" if the pixel has same values. As a result of this MSE is equivalent to the ratio of the number of pixels which are different in $I_{PROPOSED}$ with respect to $I_{ORIGINAL}$ to the total number of pixels in a video frame.

The computed MSE for every frame of all the test bench videos is zero and it confirms that the proposed memory efficient moving object detection scheme produces the same moving object detection results as the original clustering based moving object detection scheme without negatively affecting the quality of processed videos, but with 58.33% reduction in memory requirement.

VI. Conclusions

In this research article, a memory optimized moving object detection scheme, useful for designing automated video surveillance systems, has been presented. The emphasis has been given on optimizing memory requirements for storing background related information. The presented scheme is best suited for implementation on limited resources standalone embedded platforms such as low-cost low-end FPGAs for achieving real-time performance. It has been coded using C/C++ in the Microsoft Visual Studio IDE. 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. There is 58.33% reduction in memory requirements in case of proposed memory efficient algorithm for storing background related information without any loss in accuracy and robustness of moving object detection as compared to original clustering based scheme.

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] L. Lacassagne, A. Manzanera, J. Denoulet, and A. Merigot, High performance motion detection: some trends toward new embedded architectures for vision systems, Journal of Real-Time Image Processing, Vol. 4, No. 2, pp. 127-146, 2009.

[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.
[23] M. Genovese and E. Napoli, ASIC and FPGA Implementation of the Gaussian Mixture Model Algorithm for Real-time Segmentation of High Definition Video, IEEE Transactions on Very Large Scale Integration, Vol. 22, No. 3, pp. 537-547, 2014.
[24] R. Rodriguez-Gomez, E.J. Fernandez-Sanchez, J. Diaz, and E. Ros, FPGA Implementation for Real-Time Background Subtraction Based on Horprasert Model, Sensors, Vol. 12, No. 1, pp. 585–611, 2012.

Authors' Profiles

Sanjay Singh is working as Scientist in CSIR-Central Electronics Engineering Research Institute, Pilani, India - A Government of India Research Laboratory. He is Member of IEEE - USA, IACSIT - Singapore, IAENG - Hong Kong, and SSI - India. He is involved in various projects sponsored by Indian Government on Computer Vision and VLSI Design. His research interests are VLSI Architectures for Computer Vision, FPGA Prototyping, and VLSI Design. He received his Ph.D. in VLSI Design for Computer Vision, M.Tech. in Microelectronics & VLSI Design, M.Sc. in Electronics, and B.Sc. in Electronics & Computer Science from Kurukshetra University, Kurukshetra, Haryana, India. He earned Gold Medal (First Position in University) during his M.Tech. and M.Sc. He topped college during B.Sc. He received more than 20 Merit Certificates and Scholarships during his academic career.

Sumeet Saurav is working as SRF in CSIR-Central Electronics Engineering Research Institute, Pilani, India - A Government of India Research Laboratory. His research interests are Computer Vision and VLSI Design. He received his M.Tech. in Advanced Semiconductor Devices from Academy of Scientific & Innovative Research (AcSIR), India. He is University rank holder in B.E.

Anil Vohra is professor with Electronic Science Department, Kurukshetra University, Kurukshetra, India. At present, he is holding the positions of Dean Academics and Dean R&D at Kurukshetra University, Kurukshetra, India. He received his M.Sc. and Ph.D. from Punjab University, India.

Chandra Shekhar is the former director of CSIR-Central Electronics Engineering Research Institute, Pilani, India - A Government of India Research Laboratory. His research interests are VLSI architectures and VLSI Design Methodologies. He received his Ph.D. from BITS, Pilani, India.

