Automatic Tracking of Moving humans in Real-Time Scene for Security Applications

Hala abdel-Galil Elsayed and Mostafa-Sami M. Mostafa and Salha Ibrahim Alhashani

Department of Computer Science, Faculty of Computers and Information, Helwan, university, Cairo, Egypt

salhaabrhem@yahoo.com

Abstract

Reliable tracking of multiple moving humans in real-world video is an interesting challenge, made difficult by various sources of noise and uncertainty. We propose an efficient approach to find relation between moving humans over frames. By using gray color values and position information of the moving humans, we assign tracks to those humans. We allow for tracks to be lost and then recovered when they appear again. This dynamic method, along with the ability to recover lost tracks, adds robustness to the tracking system. We present results that show that our method performs well in difficult tracking cases and compare the results by using an Euclidean distance based method. And also we find a general way to determine the effective blob size according to the moving tracks size to be independent of camera position and this let the camera location to be dynamic not fixed static location to be used effectively in several security applications inside buildings.

Keywords: *Motion segmentation, Humans tracking, Background subtraction*

1. Introduction

The ability to track a particular object or objects in successive frames to represent the human's body is an important step in object tracking and classification applications. In many tracking applications, whether in the visible or non-visible spectrum, multiple target objects are to be analyzed at each time for steps. Finding reliable correspondences between objects in one time step to objects in the next is a non-trivial task given the non-deterministic nature of the humans, their motion, and the image capture process itself. It can be a difficult task in some cases, especially in the presence of noisy measurements from the camera, occlusion from objects in the field of view, and changes in orientation and direction of movement of the objects and also with different camera position.

In this paper, we tackle the issue of real-world multiple object correspondence in indoor stationary camera scenes inside buildings. Our application is in the visible spectrum and uses gray color information from objects. We begin with the detection of moving pixels using subtraction from a moving average background model. The moving pixels are clustered into regions, which we refer to as blobs, so that pixels belonging to a single object are grouped together representing the whole human body.

Once moving regions have been identified, the next task is to determine tracks of these objects over successive frames. This is essentially a correspondence task, the goal of which is to find a match for each blob or object in the previous frame to one of the blobs or objects in the current frame. When the information from a frame does not support the presence of an object, we allow for the corresponding track to be declared 'lost'. If the object appears again in subsequent frames, the system reassigns the track to the object.

2. Previous Work

In this section we focus on various models and techniques for video based object tracking. We review here only the most relevant work to video tracking, which includes both real-time image processing, image Segmentation and object tracking.

A real-time system is one that must satisfy explicit bounded response time constraints to avoid failure. Equivalently, a real-time system is one whose logical correctness is based both on the correctness of the outputs and their timeliness. Notice that response times of, for example, microseconds are not needed to characterize a real-time system - it simply must have response times that are constrained and thus predictable. In fact, the misconception that real-time systems must be "fast" is because in most instances, the deadlines are on the order of microseconds. But the timeliness constraints or deadlines are generally reflection of the underlying physical process being controlled [1].

Segmentation involves partitioning an image into groups of pixels which are homogeneous with respect to some criteria. Segmentation techniques are divided into two basic categories: edge-based and region-based. Edge-based segmentation is primarily used to look for image discontinuities; this technique is generally applied where changes of gray-level intensity occur in the image. Including the moving objects are usually the major exporter of information in surveillance video, most approach converge on the detection of such objects. popular methods for motion segmentation comprise Gaussian background modeling [2], [3], average background modeling [4], optical flow methods [5-6], foreground pixel modeling [7], gradient methods such as the moving edge detector [8], and tensor-based motion segmentation [9]. Easy background subtraction is an openly used method for the detection of moving objects. The background image is not always known. Sometimes needs to be automatically generated by the system [10]. In our method, we are using an average background model.

Object tracking is an important task within the field of computer vision. The proliferation of high-powered computers, the availability of high quality and inexpensive video cameras, and the increasing need for automated video analysis has generated a great deal of interest in object tracking algorithms. There are three key steps in video analysis: detection of interesting moving objects, tracking of such objects from frame to frame, and analysis of object tracks to recognize their behavior [11].

In its simplest form, tracking can be defined as the problem of estimating the trajectory of an object in the image plane as it moves around a scene. In other words, a tracker assigns consistent labels to the tracked objects in different frames of a video. Additionally, depending on the tracking domain, a tracker can also provide object-centric information, such as orientation, area, or shape of an object, Objects can be represented by their shapes and appearances [12], multiple objects tracking component of an automated CCTV surveillance system [13].

3. Motion Segmentation

First, we read a video file from digital camera which is divided into a number of frames with the default frame rate which is 30frames/second. Then convert all the frames images from color to gray and stored as one matrix for every frame. Then forming a weighted sum matrix of the R, G, and B components of each pixel as follows:

$$\text{Gray} = (Cr * R) + (Cg * G) + (Cb * B) \qquad (1)$$

Where Cr = 0.2989, Cg = 0.587 and Cb = 0.114

Working on moving objects detection in the video by comparing the reduced frames matrix. Most pixels in the frame belong to the background and static regions. Proposed algorithms are needed to detect individual targets in the scene. Since motion is the key indicator of target presence in surveillance videos, the block diagram in Fig.1.gives a high-level overview of the software architecture of the human motion detector. We used two approaches to detect the moving objects locations

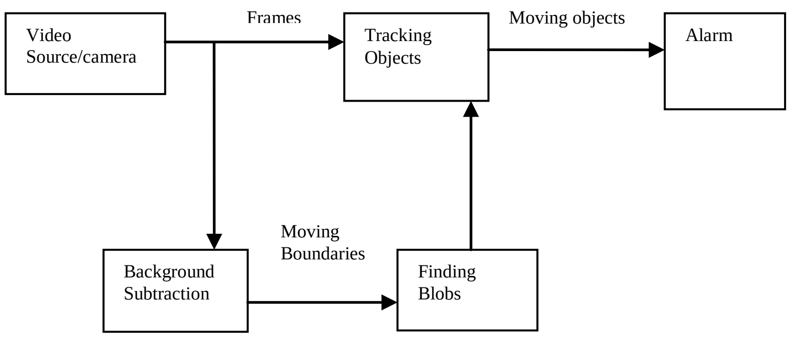


Fig (1): The software block diagram of the human motion detection system.

3.1 Average Background Model

A standard approach for finding moving objects in still-camera video data is to create a model for the background and compare that model with the frame in question to decide which pixels are likely to be the foreground (moving objects) and which the background. There are various methods for developing background models. Our approach is to use Create a simple average model where the average intensity value for each pixel over window of N (reduced number) frames is regarded as the background model.

Subtract every frame with the average background model and store these differences matrices to know the moving object in each frame. The entire background modeling and segmentation process is carried out on grayscale versions of the images. Fast-moving objects do not contribute much to the average intensity value [4]; thus, the model becomes a reasonable estimate of the background.

If Ik(y, x) is the intensity level at coordinates Y = y and X = x of the K th frame in the sequence and bg(y, x) is the average background model value at (y, x), then

$$bg\,(y, x) = \frac{1}{N}\sum_{j=1}^{N} I^{j}(y, \qquad (2)$$

The segmented output is:

$$Seg1^{k}(y,x) = \begin{cases} 1\ , & if\,|bg^{k}(y,x)| > T \\ 0\ , & otherwise \end{cases} \qquad (3)$$

Where T is a predetermined threshold, usually are equal 30.

Fig.2. shows the grayscale version of a frame that is being processed. Fig.3. is an example of how the moving average background model appears of the video sequence. After subtracting this background model from the current frame and thresholding the difference image and the results is the white pixels in the image are the regions that have been segmented as moving pixels in the current frame.

Smart FLV Converter Trial 01/01 11:09:01
Smart FLV Converter Trial 01/01 11:09:01

Fig (2).Grayscale version of current frame being processed for background segmentation

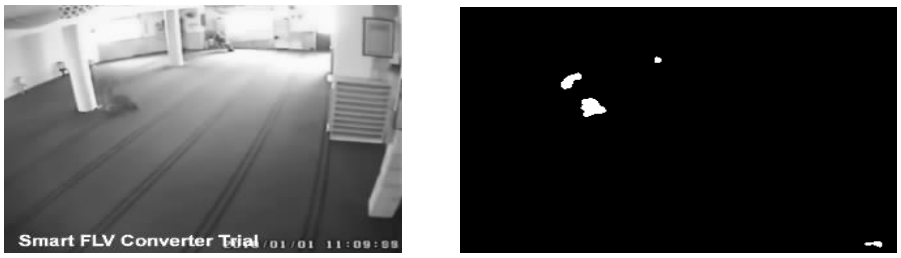


Fig (3). Moving average background model and the segmented image after subtraction from average background model and thresholding the difference

This base method does have a major drawback in that the segmented object regions have "tails" due to the effect of the window that is used for the average background model and due to the following disadvantages:

- Slow (take time of average all frames before start in segmentation)
- Inaccurate (affected by extreme track in single or few no. of frames)
- More tracks (due to subtraction all frames from a single background model)
- Same background (can't work probably with different backgrounds in videos)

3.2 Secondary Background Model

Here we subtract every frame from its previous frames and also store all these differences matrices to know the moving object in each frame. It was observed that the intensity level of a background pixel changes very little from frame to frame. The value of the background in the previous frame is actually a better estimate of the background in the current frame than is the average value over N frames.

We use a secondary model of the background (called Secondary Background) which is the actual intensity level of background pixels in the previous frame (or in the frame where the pixel was last classified as background). By combining the two models, a near perfect segmentation of the moving objects is obtained. The average model is used to identify moving regions and the Secondary Background is used to identify the valid moving object within these regions.

Secondary Background for frame k is denoted by$\mathrm{sbg}^{\mathrm{k}}$. For initialization, we start to calculate $\mathrm{sbg}^{\mathrm{k}}$ for second frame (k=2) by subtracting the intensity level of frame 2 from the intensity level or frame 1.

$$\mathrm{sbg}^1 = \mathrm{bg}^1(y,x) \qquad (4)$$

Where sbg^1 is the initial value for Secondary Background.
The segmented output based on the Secondary Background is denoted by:

$$_{seg2}^{\ k}(y,x) = \begin{cases} 1, & if(|I^k(y,x) - I^{k-1}(y,x)| > T \ and \ seg1^k(y,x) = 1) \\ 0, & otherwise \end{cases} \qquad (5)$$

T is a threshold, usually 30

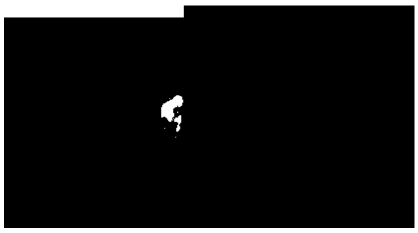
The Secondary Background model for an example frame subtracted from current frame and thresholding, as described in equation 5, results in the segmentation of the moving pixels as Fig.4.

Our background modeling method is much simpler than other adaptive background methods such as Stauffer and Grimson's Gaussian background pixel model [3]. We found these complex backgrounds; they have very complicated procedures and are computationally expensive. Our method is simple implementation was able to produce good results with little computational complexity. And from advantages to secondary model following:

- Faster (because it subtract frames directly to get segmented objects)
- More accurate (because it compare only 2subsequent frame, with each other)
- Fewer tracks (because it detect direct change in every frames)
- Different background (don't have problem with different backgrounds except in are segmented frame in the time of switching background only)

(a) Previous Frame before the subtraction of background

(b) current Frame before the subtraction of background



(c) Moving pixels segmented after subtraction from Secondary Background model
Fig (4). Secondary Background model results

4. Humans Tracking

Here we trace the exterior boundaries of objects in subtracted frames in the binary images representing the different as a result from this subtraction. Then determine which boundaries are important to consider and which ones will be ignored. First, we get all exterior boundaries of all objects in the binary difference frames by getting outer boundaries only and don't get the inner holes boundary. then, merge in these boundaries the ones which are very near to each other and this reduce the number of boundaries after that, we loop on all the result boundaries and store the ones which is large(have length>=20 pixels).

4.1 Blob Representation

The variables and their description in the data structure used for representing an individual blob are given in table.1. BlobID is the number that is given to a blob. The track number of the blob, as assigned by the tracking algorithm, is stored in this variable. The meanGray variables store the average intensity values of gray levels of all pixels belonging to the blob. The centroid position of all pixel locations is stored in yCenter and Xcenter. The size variable keeps record of the size of the blob in pixels (number of pixels belonging to the blob)

Table (1): Variables in the individual blob data structure

Variable	Description
BlobID	ID number of blob-usually the number of the track to which the blob belongs
MeanGray	average intensity values of gray levels of all pixels in blob
yCenter	Position of centroid of blob along Y axis(column)
Xcenter	Position of centroid of blob along X axis(row)
size	size of the blob in pixels

4.2 Track Representation

The data structure used to represent an individual track is trackID stores the unique track number for the track. startingFrame and endingFrame variables store the first and last frame in which the track was seen, respectively. lostCount variable keeps count of how many consecutive frames for which a track has been 'lost'. This is kept so that a track that is lost for more than 5 frames can be deleted and closed. y and x keep record of the current position of the track in Y and X directions.

Table (2): Variables in the individual track data structure

Variable	Description
trackID	ID number of track
startingFrame	The frame number when the track first appeared
EndingFrame	The frame number when the track first appeared
lostCount	The number of frames for which a track has been lost,this is 0 if the track was found in the previous frame
Y	Location of track along Y direction
X	Location of track along X direction

4.3 Match Matrix

After the moving objects have been segmented as blobs, the next task is to find correspondence between tracks of previous frame and blobs of current frame. Most commonly, a match matrix is calculated for the tracks and the blobs, and the best matches are calculated from the match matrix. Each row in the match matrix corresponds to a track from the previous frame and each column corresponds to a blob from the current frame. Each element in the matrix is, thus, the degree of match between a track and a blob. The values entered in the matrix are usually the distance (Euclidean or Manhattan) between a track and a blob. For instance, in [1] the match matrix consists of the weighted sum of Euclidean distance in position in Y and X coordinates. Smaller distance is better match. In our system, each element in the match matrix is the sum of the Euclidean distance in position values (Y and X coordinate values) and the Euclidean distance in gray values the values for Y and X values are normalized against the maximum Y and X dimensions of the images, respectively. Similarly, gray level values are normalized against their maximum possible value, which are 255.

Say$T^{k-1} = \{t1^{k-1}, t2^{k-1}, t3^{k-1}, \ldots, t^{k-1}_{u^{k-1}}\}$ is the set of tracks from the previous frame (k - 1), wheretj^{k-1} is the jth track and u^{k-1} is the total number of tracks in frame (k-1), and $o^k=\{o1^k, o2^k, o3^k, \ldots, o^k_{u^k}\}$is the set of moving blobs in frame k, where $u^k$is the total number of moving blob in frame k.

The jth row and ith column in the match matrix, denoted by $MM_{j,i}$ is given by:

$$MM_{j,i}=\sqrt{\left(\frac{\Delta y}{ydim}\right)^2 + \left(\frac{\Delta x}{xdim}\right)^2} + \sqrt{(\Delta mG/255)^2} \qquad (6)$$

Where

Δy is the difference in Y position values,

Δx is the difference in X position values,

ΔmG is the difference in mean of gray levels values between track j of the previous frame and blob i of the current frame.

Y dim and X dim are the image dimension values in Y and X coordinates, respectively (240 and 320, in our images).

The Y and X position values of a blob correspond to the location of the centroid of all pixels that have been segmented as belonging to the concerned blob. The mean gray levels values for each blob are calculated by averaging the gray level values for all pixels that have been segmented as belonging to the concerned blob.

5. Implementation parameters

5.1 Minimum allowed boundary length

We want to determine which minimum boundary length is allowed for detected moved objects, to get this, we need to try different values and check the best for our implementation to get best efficiency.

Minimum length = No. of pixels = video size* percentage
Vide size 320*240 pixels.

$$\mathrm{eff}_{minbou} = \frac{COE}{COE+W} \qquad (7)$$

Where eff_{minbou} is the efficiency for minimum boundary, and COE is correct tracks expected, and W is wrong tracks is false and missed.
We will try the following minimum lengths.

Table (3): Efficiency tracks for minimum boundary length is allowed for video

percentage	0.013	0.02	0.025	0.04	0.05
Min length	10	15	20	30	40
Correct tracks	3	3	3	3	3
Wrong tracks	3	1	0	2	2
Efficiency tracks	50%	75%	100%	60%	60%

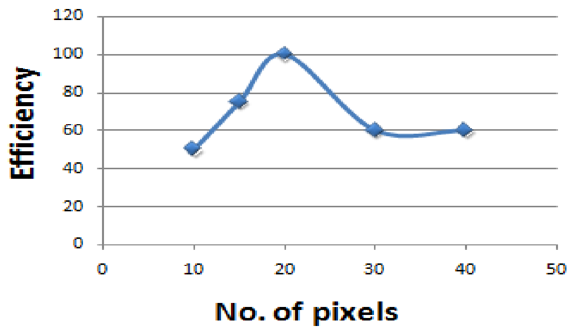


Fig (5): Minimum boundary length allowed for measure efficiency track

From Fig 5, we consider 20 pixels are the best minimum boundary lengths to get the best efficiency from our implementation.

5.2 Maximum allowed Merged distance

We need to determine the maximum distance in which we can merge objects have this distance between them , to get this, we need to check different distances in pixels and check which apply the best merge between related objects.
Distance (No. of pixels) = 320*240*percentage
The distance is the No. of pixels calculated depending on video resolution multiplying it by a specific percentage. In our cases we use video with resolution 320* 240

$$eff_{Merge} = \frac{COM}{COM+W} \tag{8}$$

Where $eff_{Merg}$the efficiency for is merge boundary, and CO is correct merge tracks expected, and W is wrong merge tracks is false and missed.
We will try the following distances.

Table (4): The values from specific frame boundaries by compare before and after merge.

Percentage	0.02	0.025	0.035	0.04	0.045
Distance	15	20	25	30	35
Correct merge	3	4	5	5	4
Wrong merge	3	2	1	1	2
Efficiency Of merge boundary	50%	65%	85%	85%	65%

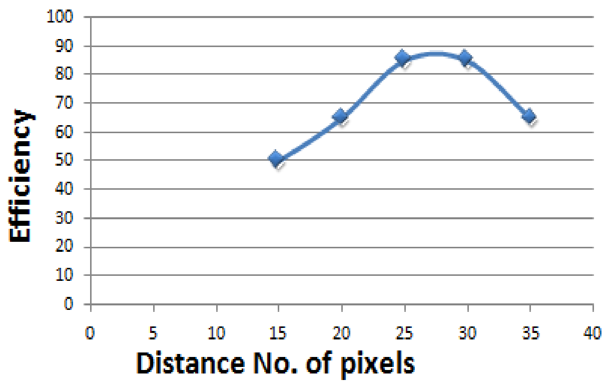


Fig (6): The relation between Max distance and merge boundaries allowed.

We found best distance range between 25 and 30 pixels we choose 25.

5.3 Binary Threshold

While converting gray frames into binary to get the boundaries from it we use a constant threshold in this conversion which is 30.

This means that if the pixels gray value otherwise will be white. We examine different values of threshold and this value get best result.

6. Results and Analysis

6.1 Results

Some results of the base system are shown below. In the figures where match matrices are presented, the assigned blob-track pairs are shaded differently in the matrix for easy reading.

Fig.7. shows the match matrix for a case when a new track has to be initialized to account for a new unmatched blob that occurs in the current frame. Current frame 030 has two new blobs compared to the previous frame 025. Since there is a blob not matched to any active previous tracks.

Fig.8 shows the match matrix when a previously active track gets lost. In this case, a track, which was active in frame 0125 is now lost. Since longer running track numbers are more reliable and less likely to be noise, we always choose the older numbered track in case of a tie between track numbers to be assigned to a blob.

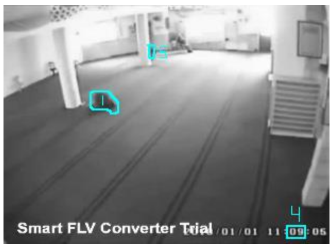
(a) previous Frame 025

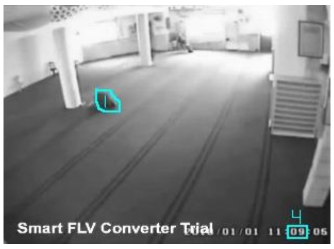
(b) current Frame 030

	Blob1	Blob2	Blob3
Track1	0.2113	0.2360	0.9530
Track2	1.1760	1.0130	0.1770

(c) Match matrix for current frame 030

Fig (7): Match matrix when new track is initialized

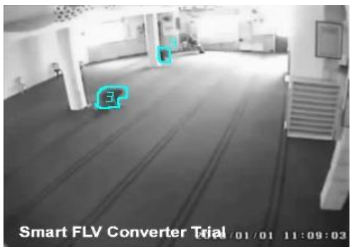

(a) previous Frame 125

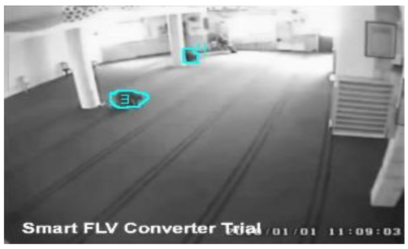

(b) current Frame 130

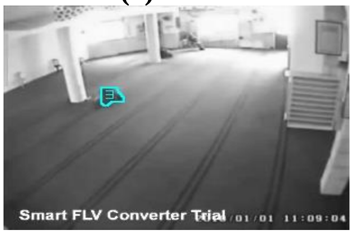
	Blob1	Blob2
Track1	0.0127	1.1818
Track4	1.2841	0.0900
Track5	0.4229	0.9387

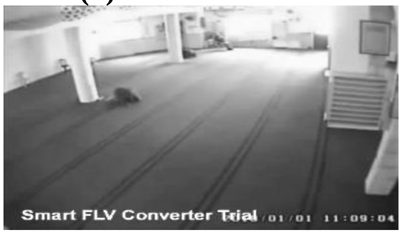
(c) Match matrix for current frame 0130

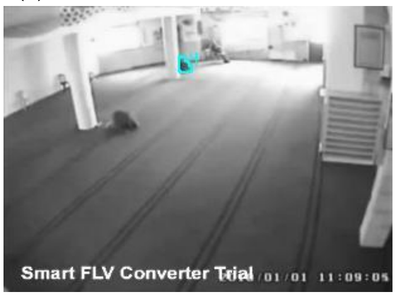
Fig (8) Match matrix when tracks are lost


(a)Frame095


(b)Frame100


(c)Frame 105-track 4 lost


(d)Frame 110-both tracks 3 & 4 lost


(e)Frame. 115-track 4 recovered

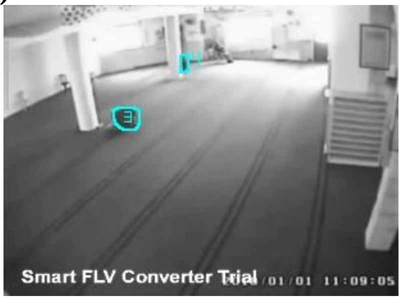

(f)Frame. 120-both tracks recovered

Fig (9) Tracks from subsequence in video

6.2 Analysis

Our performance analysis will compute the efficiency of each video sequence using the two different implementation methods (average background model and secondary background model) to draw the relation between this efficiency and method for each video sequence file used as equation.

$$eff_{Sys} = \frac{cor}{cor+wr} \tag{9}$$

Where cor : the number of correct tracks

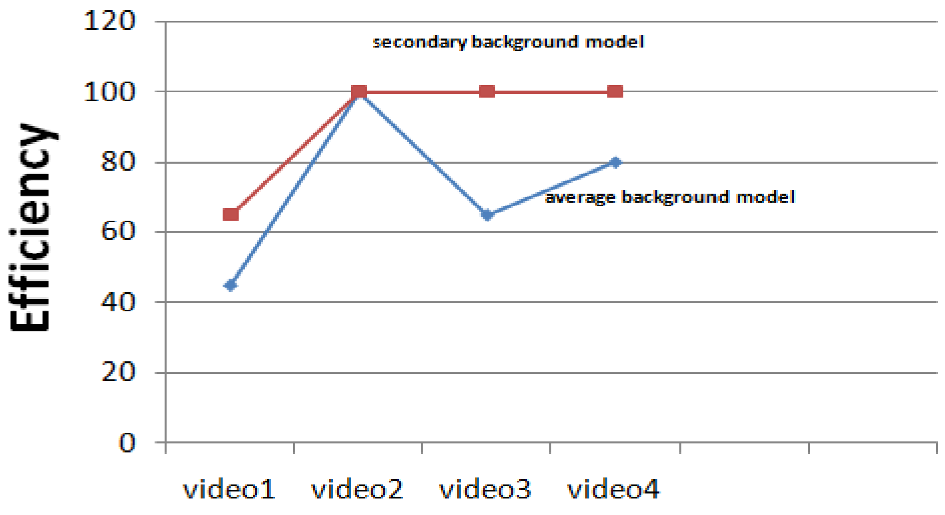
wr : The number of wrong tracks whether false or missed

eff_{Sys} is The efficiency of system

Table (5) The results of system efficiency for different video by different background model.

method		Video1	Video2	Video3	Video4
average background model	Correct	4	6	5	4
	Wrong	5	0	3	1
	eff_{Sys}	45%	100%	65%	80%
Secondary background model.	Correct	5	6	5	5
	Wrong	3	0	0	0
	eff_{Sys}	65%	100%	100%	100%

Then we represent these relations in the following chart



Fig(10): Performance and accuracy of the human motion detection system.

6.3 Computational Cost

The algorithm presented in this paper was designed to be efficient enough for real-time applications. For video file running at25 frames per second with a size of 320 pixel by240 pixel, real-time performance would require each frame to be processed in 40 milliseconds or less. Our results show that this algorithm is tacking little time.

The results for video file runtime speed of each stage are shown in Table.6. The software was run on a single-processor machine equipped with Core 2duo 2.0 GHz Processor.

These results demonstrate that for a frame using secondary background method Furthermore,

Real-time performance could be achieved and also if divide video into equal (periods of 2 seconds each).

Table (6) Runtime for each stage of the human motion tracking algorithm from video file "Per frame" operations are executed once for each frame of video

Stage	Time
Read frames and convert to gray	24 ms per frame
Background subtraction and getting boundaries of moving object	11 ms per frame
Tracking human objects	4 ms per frame

The Total execution time for every frame is 39ms, thus the system is capable of working for surveillance video in real time because it is less than 40ms.

The results for video from direct camera, its name chicony USB2.0 model CNF7051, its type imaging devices. The algorithm takes about 2 seconds video frames from camera (we take 30 frame/sec) in each once of CPU time to process a frames sequence of 60 frames of size 320 by 240 pixels. Real-time performance would require for each frame to be processed. It equals 33.3 milliseconds or less. The results for direct camera runtime speed of each stage are shown in Table .7.

Table (7): Runtime for each stage of the human motion tracking algorithm from direct camera "Per frame" operations are executed once for each frame of video.

stage	time
Read frames and convert to gray	7.8 ms per frame
Background subtraction and getting boundaries of moving object	70 ms per frame
Tracking human objects	4 ms per frame

The Total execution time for every frame is 81.8ms.

Comparison between different object tracking techniques

The following table represented our technique and other technique of real-world surveillance scenarios [14][15] with different factor.

Table (8):Comparison between the proposed technique and different technique in different factor

factor	our technique	Technique[5]	Technique[4]
No. of frame per second	- data video file 25 frame per second - data video from direct camera 30frame per second With different video size	30 frame per second	-Size video to 640 * 480 pixels take 5 frame per second. - Size video to 320*240 pixels take15 frame per second.
Total execution time of system	-recorded video file take 39 ms -direct video file take 81.8 ms the system is capable of working in real time.	108 ms.	250 ms.
Specifications of used pc	Single-processor machine equipped with Core 2duo 2.0 GHz Processor.	single-processor machine equipped with a 2.0 GHz Pentium 4 Processor.	The program was tested on an Intel Core i7, 2.67 GHz processor

7. Conclusion and Future work

Automatic tracking is a very interesting research area that can lead to intelligent applications. Our work focused on setting up a system that may be used for video surveillance of security scenarios.

Our work focuses on developing a framework to detect moving objects and generate reliable tracks from real-world surveillance video. After setting up a basic system that can serve as platform for further automatic tracking research. It based on object motion in different parts of the scene. The proposed new algorithm, consisting of five stages i.e. image Sequence (video), motion segmentation, noise removal, object tracking and alarm system.

Proposed new algorithm can process in real time with high performance, for both real-time video and recorded video with different sizes. Total execution time is small compare with different techniques.

Future work

We can improvement in the current system's segmentation and tracking algorithms, and extension of the system towards building higher-level intelligence applications based on motion tracks. In terms of improving the current system, there are many opportunities for future work in the object segmentation, the object tracking algorithm sections. It has a few drawbacks and it may be improved for future applications. For tracking algorithms where multiple classes of objects are to be tracked. It can be useful in calculating object size in different parts of the scene. It can also be used as a robust method to detect 'active' regions of the scene.

References

[1] S. Intille, J. Davis, and A. Bobick," Real-time closed-world tracking", IEEE CVPR, pages 697-703, 1997.

[2] Tao Zhao and Ram Nevatia, "Tracking multiple humans in crowded environment", CVPR, 02:406-413, 2004.

[3] Chris Stauffer and W. Eric L.Grimson, "Learning patterns of activity using real-time tracking", IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(8):747-757, 2000.

[4] Nils T Siebel and Steve Maybank, " Real-time tracking of pedestrians and vehicles", In Proceedings of the 2nd IEEE International Workshop on Performance Evaluation of Tracking and Surveillance , Kauai, USA, December ,2001.

[5] J.L. Barron, D.J. Fleet, S.S. Beauchemin, and T.A. Burkitt, "Performance of optical flow techniques", CVPR, 92:236-242, Mar 6, 2009.

[6] J. Wang and E. Adelson, "Spatio-temporal segmentation of video data", in SPIE Proc. Image and Video Processing II, 1994.

[7] Yaser Sheikh and Mubarak Shah, "Bayesian modeling of dynamic scenes for object detection", IEEE Transactions on Pattern Analysis and Machine Intelligence, 27(11):1778-1792, 2005.

[8] L. Biancardini, E. Dokadalova, S. Beucher, and L. Letellier, "From moving edges to moving regions", In Proceedings Second Iberian Conference , volume 1, pages 119-127, 2005.

[9] Gerald Kuhne, Stephan Richter, and Markus Beier, "Motion-based segmentation and Contour-based classification of video objects", In multimedia'01: Proceedings of the Ninth ACM International Conference on Multimedia, pages 41-50, New York, NY, USA, 2001.

[10] Sangho Park and J. K. Aggarwal, "Recognition of two-person interactions using a Hierarchical Bayesian Network", In IWVS '03: First ACM SIGMM International Workshop on Video Surveillance, pages 65-76, New York, NY, USA, 2003.

[11] Alan C. Bovik , "The Essential Guide to Video Processing", 2 Edition, 2009.

[12] Johnson I Agbinya and David Rees, "Multi-Object Tracking in Video", Real-Time Imaging 5, 295-304 ,1999.

[13] Alexandros Iosifidis, Spyridon G. Mouroutsos, "Real-time video surveillance by a hybrid static/active camera mechatronic system", 2010 IEEE/ASME International Conference on Advanced Intelligent Mechatronics Montréal, Canada, July 6-9, 2010.

[14] Gell´ert M´attyus, "multi target tracking on aerial videos", ISPRS Istanbul Workshop on Modeling of optical airborne and spaceborne Sensors, WG I/4, IAPRS Vol. XXXVIII, part 1/W4, Oct. 11-13,2010.

[15] David Moore, "real-world system for human motion detection and tracking", California Institute of Technology, June 5, 2003.

[16] Guohui Li, Jun Zhang, Hongwen Lin, D. Tu, and Maojun Zhang, "A moving object detection approach using Integrated Background template for smart video sensor", In Proceedings of the 21st IEEE Instrumentation and Measure-ment Technology Conference, volume 1, pages 462-466, May, 2004.

[17] Soniya Kumari, "real time image processing", cochin university of science & technology kochi- 682022, march, 2010.