Implementation of an Automated Single Camera Object Tracking System Using Frame Differencing and Dynamic Template Matching

Karan Gupta[1], Anjali V. Kulkarni[2]

1 karan@cse.vnit.ac.in, National Institute of Technology, Nagpur, India
2 anjalik@iitk.ac.in, Indian Institute of Technology, Kanpur, India

Abstract - **In the present work the concepts of dynamic template matching and frame differencing have been used to implement a robust automated single object tracking system. In this implementation a monochrome industrial camera has been used to grab the video frames and track an object. Using frame differencing on frame-by-frame basis a moving object, if any, is detected with high accuracy and efficiency. Once the object has been detected it is tracked by employing an efficient Template Matching algorithm. The templates used for the matching purposes are generated dynamically. This ensures that any change in the pose of the object does not hinder the tracking procedure. To automate the tracking process the camera is mounted on a pan-tilt arrangement, which is synchronized with a tracking algorithm. As and when the object being tracked moves out of the viewing range of the camera, the pan-tilt setup is automatically adjusted to move the camera so as to keep the object in view. The system is capable of handling entry and exit of an object. Such a tracking system is cost effective and can be used as an automated video conferencing system and also has application as a surveillance tool.**

I. INTRODUCTION

Object tracking is central to any task related to vision systems. Tracking objects can be complex [1] due to loss of information caused by projection of the 3D world on a 2D image, noise in images, complex object motion, non rigid or articulated nature of objects [5], partial and full object occlusions, complex object shapes, scene illumination changes, and real-time processing requirements.

Tracking is made simple by imposing constraints on the motion and/or appearance of objects. In our application we have tried to minimize the number of constrains on the motion and appearance of the object. The only constraint on the motion of the object is that it should not make sudden change in the direction of motion while moving out of the viewing range of the camera. Unlike other algorithms [4] the present algorithm is capable of handling the entry and exit of an object. Also, unlike [5],[6] , no colour information is required for tracking an object. There is no major constraint on the appearance of the object though an object which is a little brighter than the background gives better tracking results.

There are three key steps in implementation of our object tracking system:

- Detection of interesting moving objects,
- Tracking of such objects from frame to frame,
- Analysis of object tracks to automate the pan-tilt mechanism

A. Frame Differencing

The system first analyses the images, being grabbed by the camera, for detection of any moving object. The *Frame Differencing* algorithm [2], [7] is used for this purpose, which gives as output the position of the moving object in the image. This information is then used to extract a square image template (of fixed size) from that region of the image. The templates are generated as and when the appearance of the object changes significantly.

B. Dynamic Template Matching

The newly generated template is then passed on to tracking module, which starts tracking the object taking the template as the reference input. The module uses template-matching [3],[6] to search for the input template in the scene grabbed by the camera. If the object is lost while tracking (signifying that the object has changed its appearance) a new template is generated and used. Since the image templates, being used for matching, are generated dynamically the process is called Dynamic Template Matching [12].

C. Pan-Tilt Mechanism

The movement of the object is analyzed for automation of the Pan-Tilt mechanism [13]. Depending upon the movement of the object the pan-tilt mechanism is operated to keep the object in the camera's view.

This paper is organized as follows. Section II describes in detail the algorithm employed in our automated tracking system. The hardware setup of the pan-tilt setup is discussed briefly in Section III. In Section IV, we present the experimental results of a single person tracking example. Section V is devoted to conclusions.

II. THE ALGORITHM

In this section we will explain and analyze the algorithm employed in our automated tracking system. The discussion and analysis of the algorithm is divided into three subsections.

A. Object Detection using Frame Differencing

Identifying moving objects from a video sequence is a fundamental and critical task in an object tracking vision application. The approach we choose was to perform *frame differencing* [2] on consecutive frames in the image acquisition loop, which identifies moving objects from the portion of a video frame that differs significantly from the previous frame.

The frame method basically employs the image subtraction operator. The image subtraction operator [14] takes two images as input and produces as output a third image whose pixel values are simply those of the first image minus the corresponding pixel values from the second image. The subtraction of two images is performed straightforwardly in a single pass. The output pixel values are given by:

$$Q(i , j) = P1(i , j) – P2(i , j) \quad (1)$$

There are many challenges in developing a good *frame differencing* algorithm for object detection. First, it must be robust against changes in illumination. Second, it should avoid detecting non-stationary background objects such as moving leaves, rain, snow, and shadows cast by moving objects. In our efforts to develop a high performance algorithm for object tracking we have tried to overcome the difficulties in our object detection module. The images obtained from the camera are preprocessed to eliminate unwanted disturbances. This includes application of a *thresh-holding operation* [14],[15] followed by the use of *iterative-morphological erosion operation* [14][15]. By performing these operations we were able to desensitize the object detection from inaccuracies in the background. The goal of the object detection module is to provide the object tracking module with the positional information of the moving object.

Algorithm for Object Detection Module:

1. Grab the i^{th} frame f_i (8 bit gray-scale)

2. Retrieve the $(i-3)^{th}$ frame f_{i-3} from the image buffer.

The image buffer is an array of image variables which are used for temporary storage of frames. The array is programmed to behave as a queue with only three elements at any given point of execution.

3. Perform *Frame Differencing Operation* on the i^{th} frame and the $(i-3)^{th}$ frame where the resultant image is represented as f_r (8 bit gray-scale)

$$f_r = f_i - f_{i-3} \quad (2)$$

Here it is noticeable that instead of subtracting the i^{th} frame from the $(i-1)^{th}$ frame we are subtracting the i^{th} frame from the $(i-3)^{th}$ frame. This has been done taking into consideration that even slow moving objects should detected. It has been observed that image subtraction on consecutive frames detects only fast moving objects (objects whose position changes noticeably from one frame to other). Such a method fails to detect slow moving objects. Therefore to remove this limitation we subtract i^{th} frame from the $(i-3)^{th}$ frame to ensure a detection which is independent of speed.

4. Perform the *Binary Thresh-holding Operation* on f_r separating the pixels corresponding to the moving object from the background .This operation also nullifies any inaccuracies introduced due to the camera flickering. The result of this operation is a binary image, f_b , wherein only those pixels are set as '1' which correspond to the moved object.

In the thresh-holding technique a parameter called the brightness threshold (T) is chosen and applied to the image f[m,n] as follows:

IF f[m,n]≥T f_b[m,n]=object=1
ELSE f_b[m,n]=background=0

This version of the algorithm assumes that we are interested in light objects on a dark background. For dark objects on a light background we would use:

IF f[m,n]≤ T f_b[m,n]=object=1
ELSE f_b[m,n]=background=0

While there is no universal procedure for threshold selection that is guaranteed to work on all images, there are a variety of alternatives. In our case we are using fixed threshold (a threshold that is chosen independently of the image data). As our main objective is to separate the object pixels from that of the background this approach gives fairly good results.

5. Perform an *Iterative Mathematical Morphological Erosion Operation* on f_b to remove really small particles from the binary image. The result of this step is again a binary image, f_{bb}. This step ensures that small insignificant movements in the background are ignored ensuring better object detection.

6. Calculate the center of gravity (COG) of the binary image f_{bb} .The result of this operation is a set of two integers C(cog_x ,cog_y) which determines the position of the moving object in the given scene.

The COG is calculated by:

$$cog\_x = cog\_y + x \quad (3a)$$
$$cog\_y = cog\_y + y \quad (3b)$$
$$Total = Total + 1 \quad (3c)$$

for each pixel where x, y is the current pixel location. The resulting COG is then divided by the Total value:

$$cog\_x = cog\_x/Total \quad (3d)$$
$$cog\_y = cog\_y/Total \quad (3e)$$

to result in the final x, y location of the COG..

7. Transfer the positional information C(cog_x ,cog_y) to the object tracking module.

8. Store the frame f_i in the image buffer and discard the frame f_{i-3}

9. Increment *i* by 1

10. Goto (Step 1)

Fig. 1 shows the different stages on application of the above algorithm to a set of consecutive frames.

.

(a) Sample Frame i

(b) Sample Frame i-3

(c) Result of image subtraction on frame i and frame i-3

(d) Result of thresh-holding operation on image 1 (c)

(e) Result of morphological erosion operation on image 1(c)

(f) Center of gravity determination

Fig. 1 Different stages in application of Frame Differencing algorithm to detect a moving object.

B. Object Tracking using Dynamic Template Matching

The aim of our object tracker module is to generate the trajectory of an object over time by locating its position in every frame of the video. In our implementation the tasks of detecting the object and establishing correspondence between the object instances across frames are performed simultaneously (i.e multithreaded).

The possible object region in every frame is obtained by means of our object detection module, and then our tracker module corresponds the object across frames. In our tracking approach the object is represented using the Primitive Geometric Shape Appearance Model [1],[3] (i.e the object is represented as a rectangle). The limitation of the *frame differencing algorithm* in not being able to detect stationary objects is the reason why the template matching process must be coupled with it to create a good tracking system.

The object tracking module takes the positional information C(cog_x ,cog_y) of the moving object as an input from the object detection module. This information is then used to extract a square image template (whenever required) from the last acquired frame. The module keeps on searching it in the frames captured from that point. Whenever found it displays a red overlaid rectangle over the detected object. If the template matching doesn't yield any result (signifying that the object has changed its appearance) a new template is generated and used. As the templates for searching are being generated dynamically the object tracking module is said to be employing *the dynamic template matching* algorithm.

Further on the (x, y) co-ordinates of the tracked object are calculated and passed on to the pan-tilt module for analysis and automation.

Algorithm for the Tracking Module:

1 Get the positional information C(cog_x ,cog_y) of the object from the *Object Detection Module.*

2 Generate a image template T_i by extracting a square image from the last frame grabbed by the camera. The template is extracted in the form of a square whose image coordinates are given by

-Top, Left corner: (cog_x - 100, cog_y - 100)
-Top, Right corner: (cog_x + 100, cog_y - 100)
-Bottom, Left corner: (cog_x - 100, cog_y + 100)
-Bottom, Right corner: (cog_x + 100, cog_y + 100)

3 Search the generated template T_i in the last frame grabbed by the camera. This is done by using an efficient *template matching algorithm as in* [10].

4 IF the template matching is successful
THEN
IF the tracker has NOT detected motion of the object
AND the detector has
THEN goto STEP 1 (get a new template)
ELSE goto STEP 5 (get the x, y position)
ELSE goto STEP 1 (get a new template)

5 Obtain the position P(x, y) of the match and pass it on to the pan-tilt automation module for analysis.

6 Goto STEP 3

The output of Object Tracking is further utilized by the Pan-Tilt Automation module to operate the pan or tilt motors according to the behavior of the object.

Fig. 2 shows the dynamically generated template in correspondence with the center of gravity co-ordinates determined in the Fig 1(f)

C. Pan-Tilt Automation

The Pan-Tilt Automation module serves the purpose of operating the pan-tilt motors depending upon the motion behavior of the object being tracked. The goal of the module is to operate the motors such that the object remains in view all the time. This module takes the position C(cog_x, cog_y) of the tracked object as the input for analysis.

Depending upon the direction of movement of the object the pan-tilt mechanism is operated. Any object is allowed to enter the viewing range of the camera. This means that the

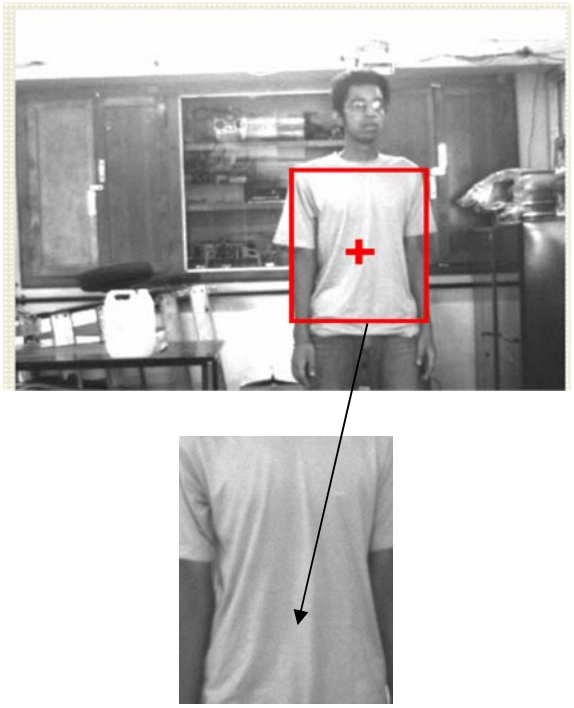

Fig. 2. A template generated dynamically based on the Center of Gravity information of the moving object

mechanism is not activated for an object which enters the scene, it's only activated when the object under consideration tries to leave the scene. In such a scenario the camera is panned or tilted in the direction of the movement of the object. This ensures that the object stays in view of our tracking system even if it tries to leave the viewing range of the camera.

Algorithm for the Pan-tilt Automation Module:

1 Get the position P(x, y) of the tracked object from the *Object Tracking Module.*

2 Get the horizontal direction of movement (D_H) of the tracked object (as shown in Fig. 3.)

3 Get the vertical direction of movement (D_V) of the tracked object (as shown in Fig. 3.)

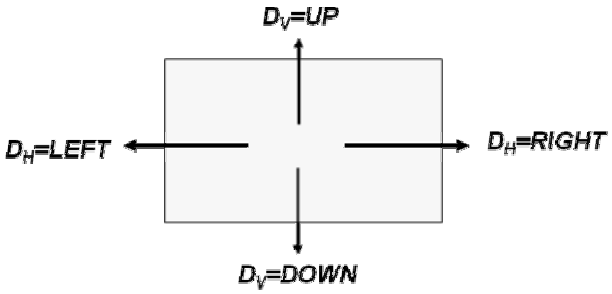

Fig.3. Values of D_H and D_V depending upon the direction of motion

4 IF (x >700) { region C} AND (DH = RIGHT) THEN *Pan Right*
ELSE IF (x < 100) { region A} AND (DH = LEFT) THEN *Pan Left*
ELSE IF (y < 100) { region B} AND (DV = UP) THEN *Tilt Up*
ELSE IF (y > 500) { region D} AND (DV = DOWN) THEN *Tilt Down*
ELSE goto STEP 1

The commands Pan Left, Pan Right, Tilt Down and Tilt Up have the following meanings
-Pan Left : The pan stepper motor is turned by a fixed angle Anticlockwise
-Pan Right : The pan stepper motor is turned by a fixed angle clockwise
-Tilt Up : The tilt stepper motor is turned by a fixed angle Anticlockwise
-Tilt Down : The tilt stepper motor is turned by a fixed angle clockwise
The regions corresponding to the conditions above are shown in Fig. 4.

One important thing to be noted is that during the time when the motors are turning the object tracking is stopped. The system assumes that after the fixed angle rotation the object comes back in view. This limitation is due to the fact that frame differencing algorithm has been employed for object detection. Algorithms which detect objects while the camera is in motion [8], [9] can be used but they are computationally expensive.

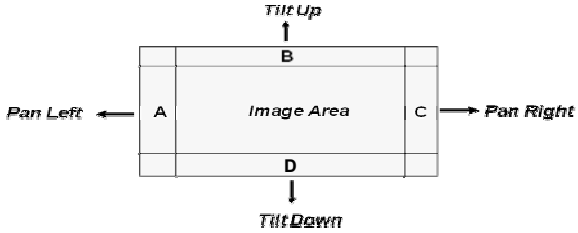

Fig. 4. Diagram showing different regions of the camera view and the pan-tilt directions

C. Merging the three modules

Having discussed all the three main modules of our object tracking system we will now show how the three different process have been brought together to form a single algorithm.

The three processes are *Multithreaded* for concurrent execution. Hence instead of executing sequentially they execute parallel to each other. The data exchanged between the three processes is shown in Fig. *5*. This approach has a major advantage, the simultaneous execution of the three processes ensure that the object in question is detected and tracked immediately i.e there is no lag between detection and tracking of the object.

III. HARDWARE SETUP

This section deals with the pan-tilt arrangement of the tracking system. The system utilized two stepper motors to account for the pan and tilt motion of the camera. As presented in Fig. *6* each motor is operated using an electronic stepper motor driver circuit. The motor driver circuits are in turn controlled by the tracking program through the parallel port

interface. The *tilt* motion of the camera is governed by the stepper motor installed vertically whereas the *pan* motion of the camera is governed by the motor installed horizontally. The driver circuit used to operate the motors has three inputs which are connected to the parallel port of the computer. These three inputs are:

1. ***Motor On/Off:*** The motor is turned on as long as a + 5V signal is given to this line
2. ***Direction Control*****:** A zero volt signal defines the clockwise rotation whereas +5 volts defines the anti-clockwise direction.
3. ***Speed Control*****:** A square wave is provided to this input line. The frequency of the square wave determines the speed of rotation of the motor.

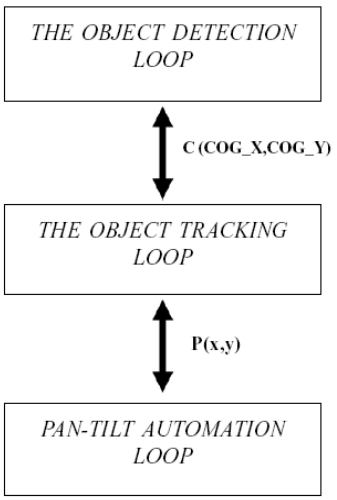

Fig. 5. Concurrent execution of the 'Object Detection', 'Object Tracking' and the 'Pan-Tilt Automation' modules

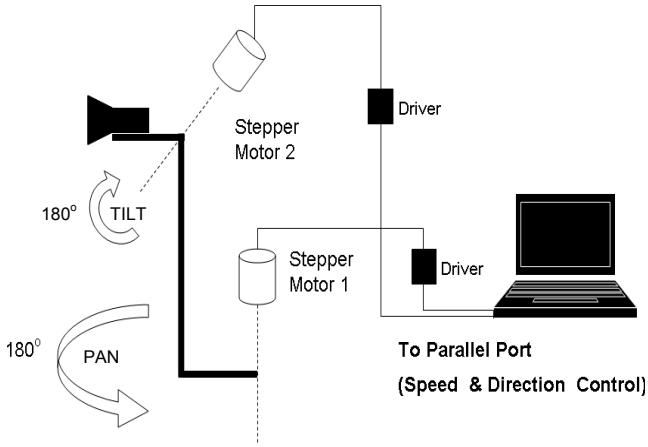

Fig. 6 A schematic diagram of the Pan-Tilt setup

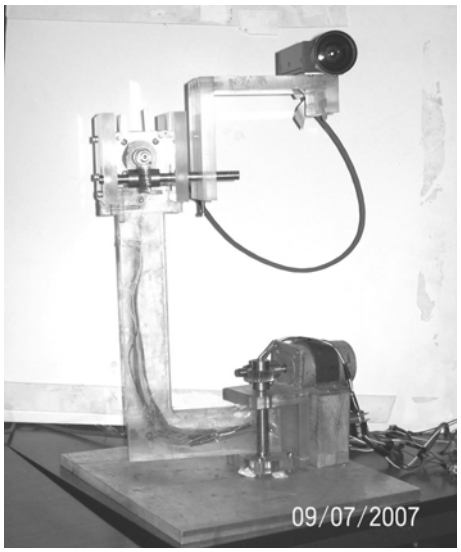

Fig. 7 Actual pan-tilt setup

IV. EXPERIMENTS AND RESULTS

Results are presented of the experiment performed using the tracking setup shown in Fig *7*. The experiment was performed using a dual core Pentium machine fitted with a high performance image acquisition card.

The camera used was a monochrome industrial camera configured to take 640x480 resolution images at 30frames/second.

Using the given setup the tracking system was found to work well under different lighting conditions. Also the pan-tilt mechanism responded as desired to the movement of the object being tracked.

The experiment involved the tracking of a single person walking inside a room. The results of the experiment are presented in Fig *8*. Please note that the faint red square represents the result of the template matching module and has been superimposed by the tracking module on the original frame.

In the given sequence of images the camera remains stationary from Fig 8(a) to Fig 8(b) as the person remains within the viewable limits. In Fig 8(f) the person is occluded as it reaches the end of the video frame. It can be seen that even in such a condition the person is being tracked. Also in Fig 8(f) the object has crossed the end limit. To keep the person in camera's view the pan-tilt mechanism pans the camera to left. This is evident from Fig 8(g), where the background has changed.

Another noticeable point in Fig 8(g) is that the red square is no longer visible. This is due to the fact that in our algorithm an object is not tracked when the camera is in motion. As can be seen in Fig 8(h) the system starts tracking the person again. This happens when the camera becomes stationary again.

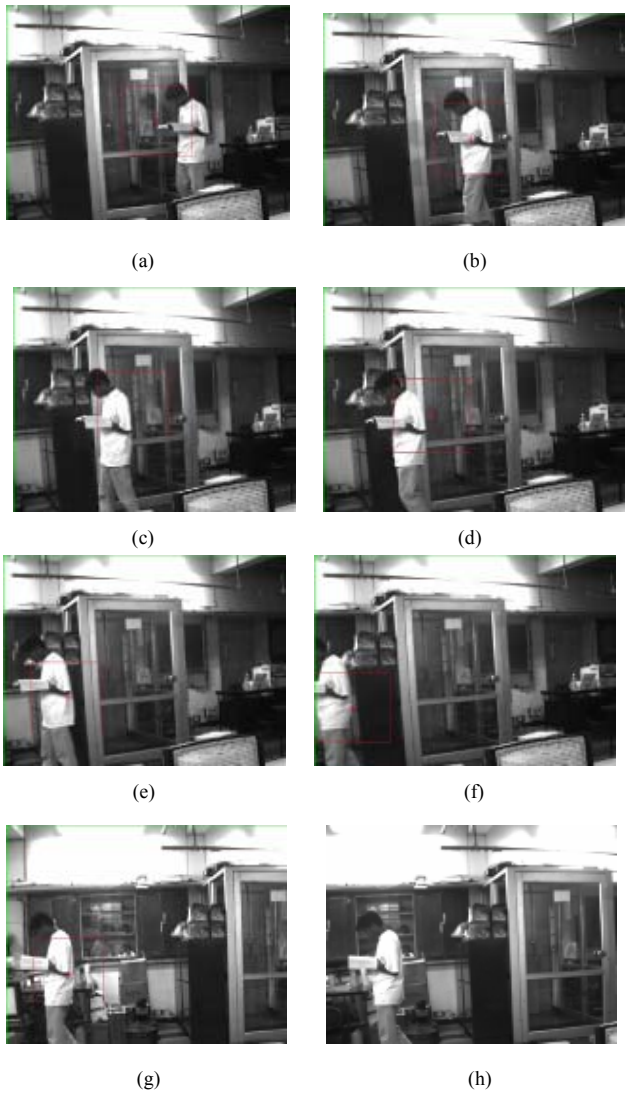

Fig. 8 Experimental results from real time tracking of a person walking inside a room

V. CONCLUSION

A robust and efficient automated single object tracking system is presented. The system has been implemented using a algorithm based on frame differencing and dynamic template matching. The algorithm has experimentally been shown to be quite accurate and effective in detecting a single moving object even under bad lighting conditions or occlusions. The system has been automated using a pan-tilt setup which is synchronized with the algorithm. Such an automated object tracking system can be used in applications where accurate tracking is required but good lighting conditions can not be provided. The system is also very much applicable to areas like surveillance and video conferencing. Future work focuses on tracking multiple objects at the same time as well as on improving tracker accuracy during camera motion.

ACKNOWLEDGMENT

Thanks are due to Dr. Bhaskar Dasgupta of ME Department for giving the idea of developing an object tracking system. Mr. Rajendra's help is duly acknowledged in the hardware set-up development.

REFERNCES

[1]. YILMAZ, A., JAVED, O., AND SHAH, M. 2006. Object tracking: A survey. *ACM Comput. Surv. 38*, 4, Article 13 (Dec. 2006)

[2]. JAIN, R. AND NAGEL, H. 1979. On the analysis of accumulative difference pictures from image sequences of real world scenes. *IEEE Trans. Patt. Analy. Mach. Intell. 1*, 2, 206–214.

[3]. COMANICIU, D., RAMESH, V., AND MEER, P. 2003. Kernel-based object tracking. *IEEE Trans. Patt. Analy. Mach. Intell. 25*, 564–575.

[4] BAR-SHALOM, Y. AND FOREMAN, T. 1988. *Tracking and Data Association*.Academic Press Inc

[5] RICHARD Y. D. XU , JOHN G. ALLEN , JESSE S. JIN, Robust real-time tracking of non-rigid objects, Proceedings of the Pan-Sydney area workshop on Visual information processing, p.95-98, June 01, 2004

[6] FIEGUTH, P. AND TERZOPOULOS, D. 1997. Color-based tracking of heads and other mobile objects at video frame rates. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 21–27.

[7] HARITAOGLU, I., HARWOOD, D., AND DAVIS, L. 2000. W4: real-time surveillance of people and their activities. *IEEE Trans. Patt. Analy. Mach. Intell. 22*, 8, 809–830.

[8] MONNET, A., MITTAL, A., PARAGIOS, N., AND RAMESH, V. 2003. Background modeling and subtraction of dynamic scenes. In *IEEE International Conference on Computer Vision (ICCV)*. 1305–1312.

[9] ZHONG, J. AND SCLAROFF, S. 2003. Segmenting foreground objects from a dynamic textured background via a robust kalman filter. In *IEEE International Conference on Computer Vision (ICCV)*. 44–50.

[10] SCHWEITZER, H., BELL, J. W., AND WU, F. 2002. Very fast template matching. In *European Conference on Computer Vision (ECCV)*. 358–372.

[12] S. YOSHIMURA AND T. KANADE, "Fast Template Matching Based on the Normalized Correlation by Using Multiresolution Eigenimages, "*Proc IROS '94*, Munich, Germany, 1994.

[12] L. WANG, W. HU, AND T. TAN. Face tracking using motion guided dynamic template matching. In *ACCV*, 2002.

[13] D. MURRAY AND A. BASU, "Motion tracking with an active camera," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 16, pp. 449–459, May 1994.

[14] RAFAEL C. GONZALEZ, RICHARD E. WOODS. (2002): Digital Image processing, Second Edition. Prentice Hall International

[15] EDWARD R. DOUGHERTY (1993) Mathematical Morphology in Image Processing. CRC Press