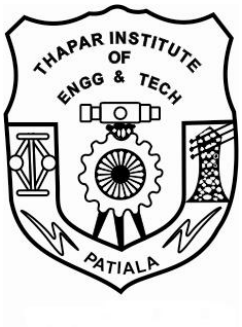
Video Image Enhancement and Object Tracking

A Thesis

Submitted in partial fulfillment of the
requirement for the award of degree of

Master of Engineering
in
Electronics and Communication Engineering



By:
Rajan Sehgal
(8044119)

Under the supervision of:
Dr. R.S. Kaler
Professor & Head, ECED

JUNE 2006

ELECTRONICS AND COMMUNICATOIN ENGINEERING
DEPARTMENT
THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY
(DEEMED UNIVERSITY)
PATIALA – 147004

ABSTRACT

To develop the real world computer vision system, detection of moving objects in video images is very important. The automatic detection of moving objects in video images is very important. The automatic detection of moving objects in monitoring system needs efficient algorithms. The common method is simple background subtraction i.e to subtract current image from background. But it can't detect the difference when brightness difference between moving objects and background is small. The other approach is to use some algorithms such as color based subtraction technique but the costs are very high and have problem in stability. Here a method is proposed to detect moving objects using difference of two consecutive frames. The objective is to provide a software that can be used on a pc for performing tracking along with video enhancement using bilinear interpolation. The program is able to track moving objects and it is structured as different blocks working together. Initially the spatial resolution and the contrast of the extracted frames of the video sequence are enhanced. The position of the object is now marked manually so as to obtain the "Region of Interest". The algorithm is implemented in MATLAB and the results demonstrate that both the accuracy and processing speed are very promising. Furthermore, the algorithm is robust to the changes of lighting condition and camera noise. The algorithm can be used in video based applications such as automatic video surveillance.

Certificate

I hereby certify that the work, which is being presented in the thesis, entitled **"Video Image Enhancement and Object Tracking"** in partial fulfillment of the requirements for the award of degree of Master of Engineering in Electronics and Communication Engineering at Electronics and Communication Engineering Department of Thapar Institute of Engineering and Technology (Deemed University), Patiala, is an authentic record of my own work carried out under the supervision of Dr. R.S. Kaler

I have not submitted the matter presented in the thesis for the award of any other degree of this or any other university.

(Rajan Sehgal)

This is to certify hat the above statement made by the candidate is correct and true to best of my knowledge.

(Dr. R.S.Kaler)
Supervisor
Professor & Head
Electronics and Communication Engineering Department,
Thapar Institute of Engineering & Technology, PATIALA-147004

Countersigned by

(Dr. R.S. Kaler)
Professor & Head
Electronics and Communication Engineering
Thapar Institute of Engineering and Technology
PATIALA-147004

(Dr. T.P. Singh)
Dean of Academic Affairs
Thapar Institute of Engineering Department and Technology
PATIALA-147004

Acknowledgement

It is with the deepest sense of gratitude that I am reciprocating the magnanimity, which my guide Dr. R.S. Kaler , Professor and Head, Electronics and Communication Engineering Department has bestowed on me by providing individual guidance and support throughout the Thesis work.

I am also thankful to Dr. S.C. Chatterjee , P.G. Coordinator, Electronics and Communication Engineering Department for the motivation and inspiration that triggered me for my thesis work.

I would also like to thank all the staff members and my co-students who were always there at the need of the hour and provided with all the help and facilities, which I required for the completion of my thesis.

I am also thankful to the authors whose works I have consulted and quoted in this work. Last but not the least I would like to thank God for not letting me down at the time of crisis and showing me the silver lining in the dark clouds.

Rajan Sehgal

(8044119)

Table of Contents

Contents **Page No.**

Chapter 1
Introduction

1.1 Background

Detection of moving objects in video images is one of the most important and fundamental technologies to develop the real world computer vision systems, such as video monitoring system, intelligent-highway system, intrusion surveillance, etc. Traditionally, the most important task of monitoring safety is based on human visual observation, which is a hard work for watchmen. Therefore, the automatic detection of

moving objects is required in the monitoring system that can help a human operator, even if it cannot completely replace the human's presence. To facilitate a monitoring system,

efficient algorithms for detecting moving objects in video images need to be used. The usual method for detecting moving objects is simple background subtraction that is to subtract current image from background image. However, there exist gradual illumination changes, sudden changes in illumination and other scene parameters alter the appearance of the background. Simple background subtraction is susceptible to these changes. And when the brightness difference between moving objects and the background is small, it cannot detect the difference. In order to resolve these problems, some algorithms such as color based subtraction technique and the technique based on optical flows have been proposed**.** But the computational costs of these methods are very high and have problem in stability. In our method Moving objects are detected from the difference of two consecutive frames. This approach uses the motion to distinguish moving objects from the background. So it is more efficient than the previous approaches. In our system, images are captured with a stationary camera. The experiment results demonstrate that both the accuracy and processing speed are very promising. Furthermore, the algorithm is robust to the changes of lighting condition and camera noise.

1.2 AVI Video File Formats

1.2.1 Resolution, Pixels, Colors and Compression

There's a standard audio / video file format under Windows called AVI (Audio Video Interleave)."AVI" file is just a wrapper, a package that contains some audio / visual stuff, but with no guarantees about what's inside. Microsoft created the AVI format for packaging A/V data, but it's just a specification for sticking A/V data in a file, along withsome control information about what's inside.[5] It's sort of like having rules for putting boxes together ("insert flap A in slot B"), and rules for labeling the box, but anybody can put any sort of stuff in a box, in any language. You can make sense of the contents of a box only if you have the right translation kit.

Each developer of a new A/V format is responsible for writing the translation kits that permit your Windows system to understand that flavor of AVI. These kits are called "codecs", for compressor-decompressor, because the video and audio formats usually perform some form of compression to reduce the amount of data in the file.

1.2.2 Size Matters

Video makes large files. A single image file can get big by itself, but video consists of frame after frame of images, which keep piling on one after another. To make video useful and shareable on PC's, the amount of data must be reduced to reasonable sizes. There are three basic approaches to reducing the amount of data in a video clip: reducing the resolution and length of the clip, trimming the individual pixels and color data, and compressing the frames. Each clever new approach to squeezing down the size of video data creates yet another incompatible format, and requires a new codec to play the resulting files.

Reduction	Video Format	Size (MB)	Reduce
Resolution	640x480, 30 fps, 24-bit	26.37	100%
	320x240, 30 fps, 24-bit	6.59	25%
	160x120, 30 fps, 24-bit	1.65	6%
Frame Rate	320x240, 15 fps, 24-bit	3.30	13%
	320x240, 10 fps, 24-bit	2.20	8%
Pixel Size	320x240, 10 fps, 16-bit	1.46	6%
Color Format	320x240, 10 fps, 9-bit	0.82	3%
	320x240, 10 fps, 12-bit	1.10	4%
Compression	320x240, 10 fps, JPEG 90%	0.29	1%
	320x240, 10 fps, JPEG 75%	0.18	1%

Table 1.1 AVI Video Formats[5]

Various approaches to reducing the amount of data include:

- Resolution: Dropping the frame size from 640 x 480 to 320 x 240 to 160 x 120
- Frame rate: Lowering the frame rate from 30 to 10 frames per second
- Pixel size: Trimming the pixel depth from 32 to 16 bits per pixel
- Color format: Subsampling the color to 9- or 12-bit
- Compression: JPEG compression at different ratios

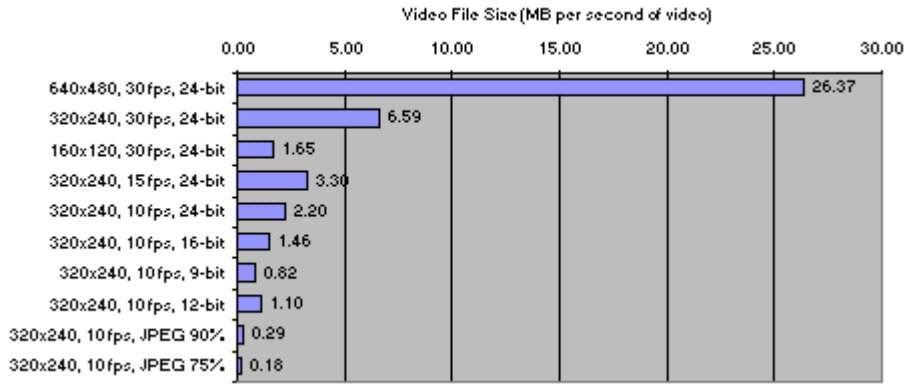


Fig. 1.1: Video File sizes: Chart of AVI video file sizes for a one-second video clip

But reducing the amount of data adversely affects the quality of the video.[1] Keeping this in mind we have tried to obtain a better quality video by just increasing the spatial resolution of the individual frames.

Chapter 2.

Resolution Enhancement using Bilinear Interpolation

2.1 What is Interpolation?

Interpolation is a bit like gambling...guessing what something is going to be. Or to put it more mathematically, "calculating values between known pixels."
An image consists of lots of pixels. For simplicity, let's stick to grayscale (black-and-white) images for now, where each pixel is simply one value - usually ranging from 0 to 255, where a higher value means more brightness, 0 being black and 255 being white. When the image is 400 x 300 pixels, and we want to enlarge it to 400%, it becomes 1600 x 1200 pixels. This means that from a mere 120,000 pixels in the original, we go to almost 2 million pixels in the enlargement. So our computer software has to "guess" what all those new pixels are going to be, based only on those 120,000 original values. The values in the original picture appear in the enlargement at every 4th pixel. The newly introduced pixels, the ones in between, we have to fill in ourselves. This can be done in many.ways[2].
For example we can simply repeat each original pixel 4 times, in both horizontal and vertical directions, this is called "nearest neighbour" . This is of course a very simple and fast method, but also yields poor results (the typical jagged mosaic effect is glaringly obvious).

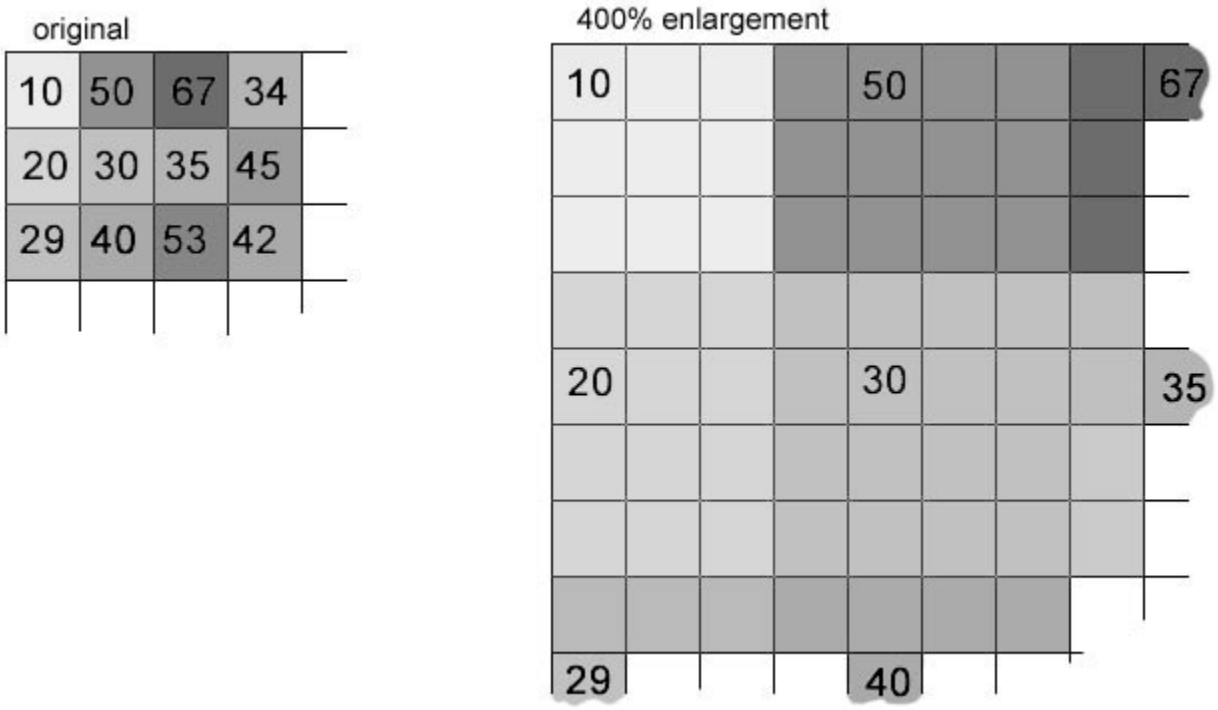


Fig. 2.1: 400% enlargement using interpolation

We can also gradually change the values from one to another. In the top row of the example image, the three unknown pixels between 10 and 50 would become (from left to right) 20, 30 and 40[3]. Doing the same in vertical direction, we can fill up all the empty places. This is called "bilinear" interpolation.

However, this technique gives better results but still it lacks preciseness, especially because the averaging of information results in adulates and loss of local contrast. Thus in order to be more precise we implement the bilinear interpolation along with minimization of the bending energy. Although this will lead to an increase in the complexity of the program but at the same time we will get better results with high quality.

A bilinear interpolation function can be written as

$$F(x,y) = a_o + a_x x + a_y y + a_{xy} xy$$

Now for each 2X2 block in the original image we will get a 4X4 block in the new image. We have to predict the values of the missing pixels, which can be done by matrix manipulation as shown.

$$\begin{pmatrix} F(x_1,y_1) \\ F(x_2,y_2) \\ F(x_3,y_3) \\ F(x_4,y_4) \end{pmatrix} = \begin{pmatrix} 1 & x_1 & y_1 & x_1y_1 \\ 1 & x_2 & y_2 & x_2y_2 \\ 1 & x_3 & y_3 & x_3y_3 \\ 1 & x_4 & y_4 & x_4y_4 \end{pmatrix} X \begin{pmatrix} a_o \\ a_x \\ a_y \\ a_{xy} \end{pmatrix}$$

From above equation we can calculate the coefficients $\{ a_o, a_x, a_y, a_{xy} \}$ which are used to predict the missing values as shown below.

1	**3**
4	**2**

1.0000	1.6667	2.3333	**3.0000**
2.0000	2.2222	2.4444	2.6667
3.0000	2.7778	2.5556	2.3333
4.0000	3.3333	2.6667	**2.0000**

Fig. 2.2: Implementing Bilinear interpolation

The results of the two interpolation methods are shown below.

Fig 2.3: Nearest neighbour :Result not satisfactory

Fig 2.4: Bilinear: Result obtained is much better

Chapter 3

Adaptive Image Enhancement

Homomorphic filtering or histogram equalizations have been used for the enhancement of images with shaded regions and images degraded by cloud cover[7]. However, it is not easy to recover the details of very high and/or low luminance regions in such images when the dynamic range of the recorded medium is smaller than that of the original images. Local contrast of very high and/or low luminance regions cannot be well represented by the dynamic range constraints. Moreover, small local contrasts in the very high and/or low luminance regions cannot be well detected by the human eye. Image enhancement which enhances contrasts that were hardly seen in the original image is possible by enhancing the local contrast as well as modifying the local luminance mean for the very high and/or low luminance regions to the level where the human eye can easily detect them.

This image enhancement algorithm was proposed by T. Peli and J.S. Lim They obtained the local luminance mean and the local contrast of original images by using 2-D low-pass filtering. It is possible to enhance an image with a hidden contrast in the shaded regions by the contrast enhancement and dynamic range reduction. In the algorithm proposed by T. Peli and J.S. Lim, the enhancement parameters need to be acquired for each image by experiments. [9]Therefore, it was inconvenient to apply this algorithm to real-time video signals captured by video camera, although image enhancement has applications for video signals as well as still images.

3.1 Peli and Lim's Algorithm

A block diagram of the image enhancement algorithm proposed by Peli and Lim is shown below. In this, an input image f is 2-D low-pass filtered to have a local luminance mean f_L. This local luminance mean f_L is subtracted from the input image f to have a local contrast f_h. f'_h is a modified local contrast where the local contrast f_h **is** multiplied by a local contrast modification factor $k(f_L)$ .This local contrast modification factor $k(f_L)$is a function of the local luminance mean where $k(fi)$ > 1 means the enhancement of the local contrast signal.

This local contrast modification factor $k(f_L)$is designed according to the input picture and how the input picture should be enhanced. f_L, is a modified local luminance mean where nonlinearity is applied to the local luminance mean. This nonlinearity is designed to control the dynamic range of the enhanced image to be in the range of the recorded medium. The sum of the modified local contrast f_h and the modified local luminance mean f_L, is the final output image **g.**

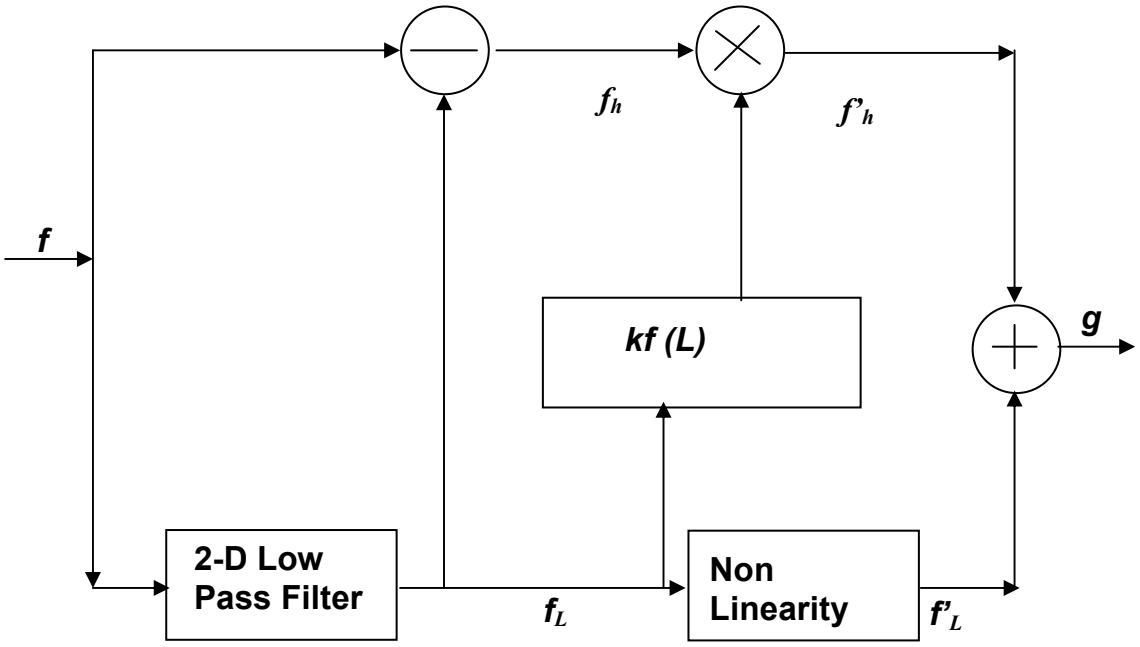


Fig 3.1: Block Diagram of Peli and Lim's Algoritm

The function of the local contrast modification factor ***k(fL)*** and the nonlinearity function, respectively, which are used for the image enhancement are shown below. These functions are designed to enhance the local contrast component.

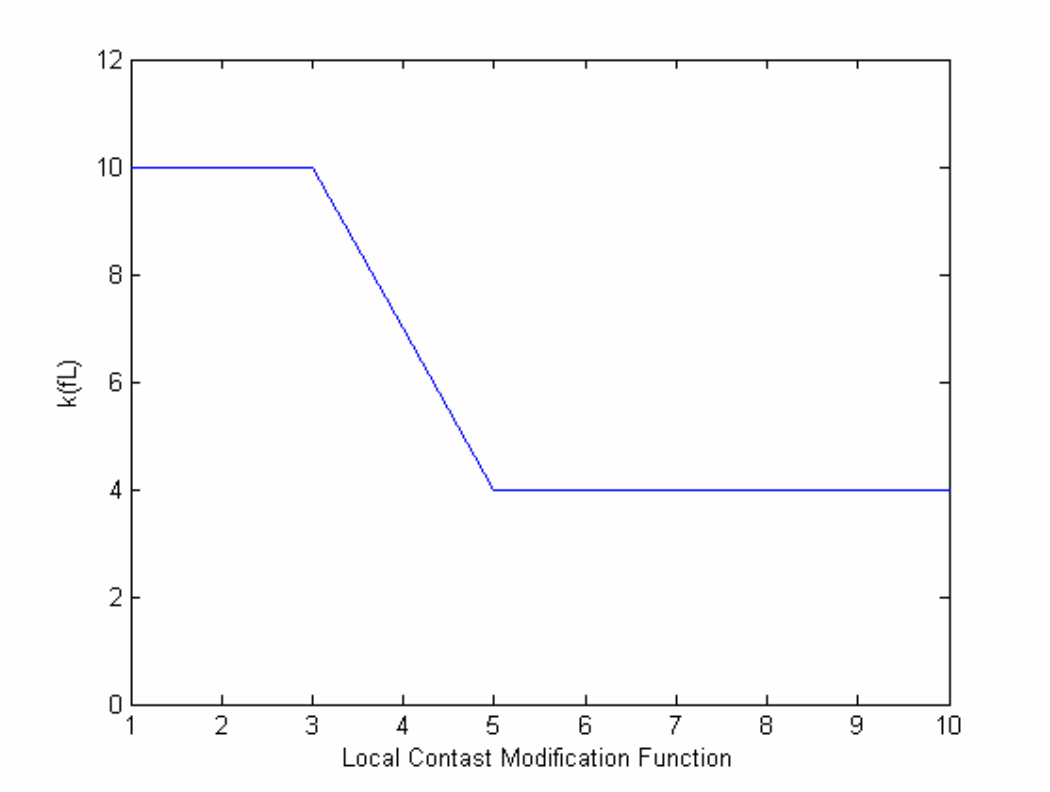


Fig 3.2: Local Contrast Modification Function

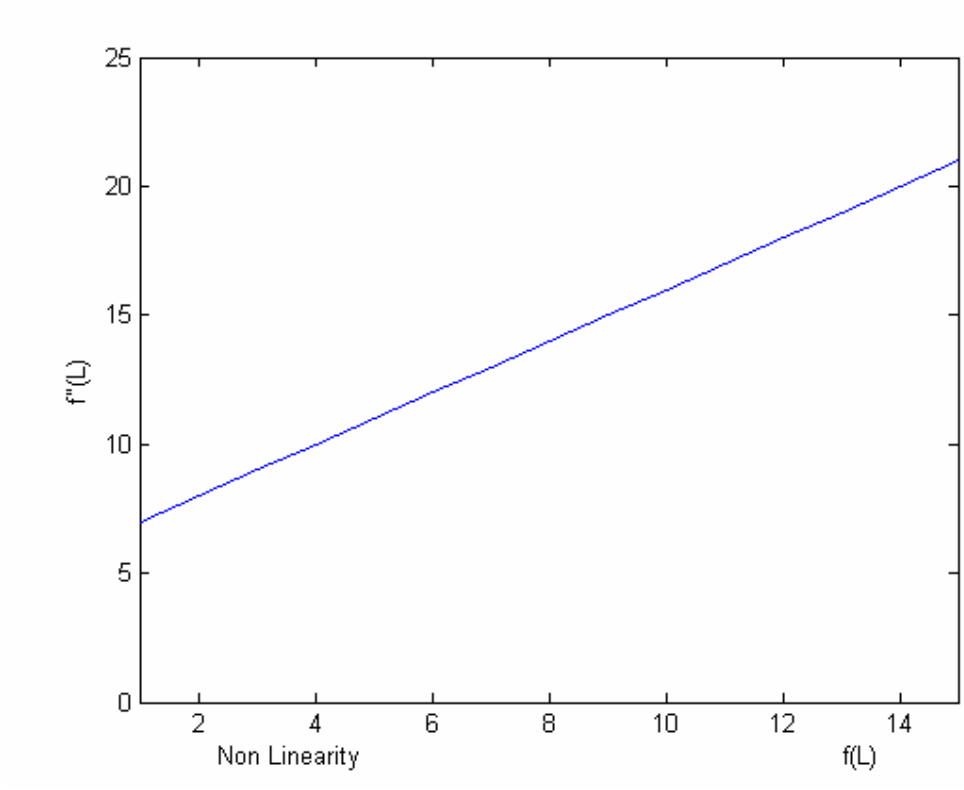


Fig 3.3: Non-Linearity used in Peli's Algorithm

3.2 Adaptive Image Enhancement using Histogram Equalization

3.2.1 Histogram Equalization:

Histogram equalization is a common technique for enhancing the appearance of images. Suppose we have an image which is predominantly dark. Then its histogram would be skewed towards the lower end of the grey scale and all the image detail is compressed into the dark end of the histogram.[1] If we could `stretch out' the grey levels at the dark end to produce a more uniformly distributed histogram then the image would become much clearer. Histogram equalization involves finding a grey scale transformation function that creates an output image with a *uniform histogram* (or nearly so).

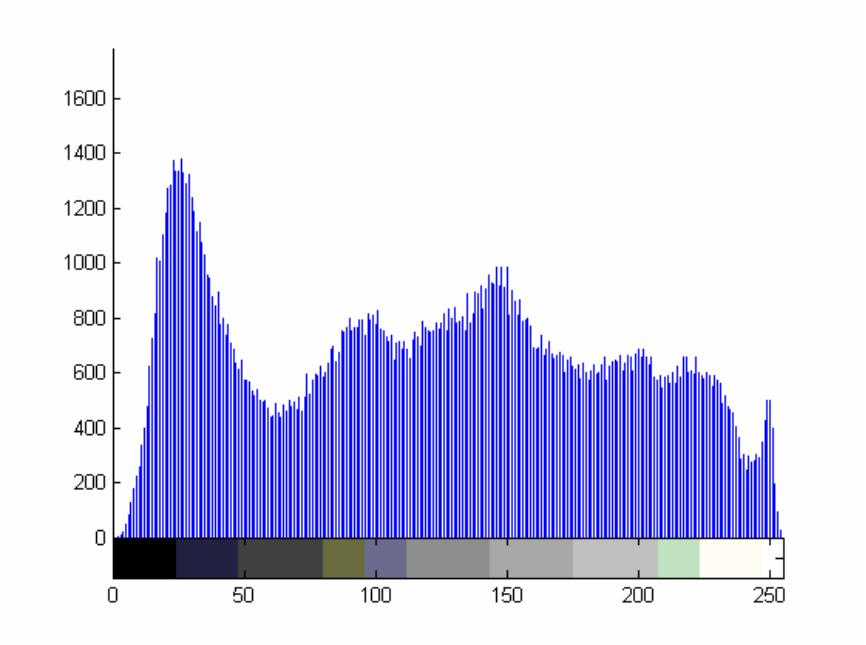


Fig 3.4: Without histogram equalization intensity values are crowded to narrow range

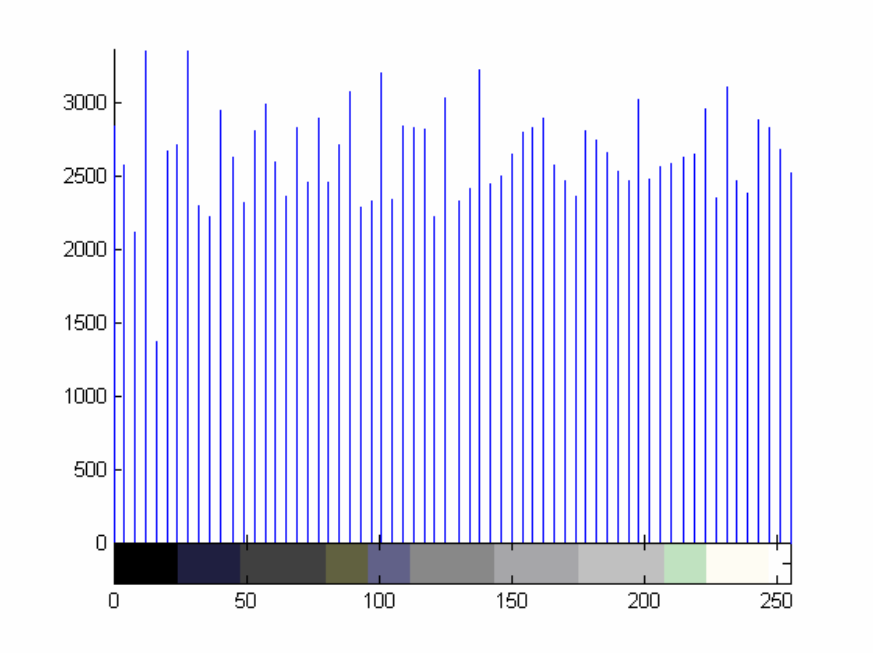


Fig 3.5: With histogram equalization intensity values get spread up improving local contrast

An effective local contrast modification factor $k(f_L)$in Peli and Lim's algorithm has to be designed for each image, which makes it difficult to apply this algorithm to a number of images in a limited time, especially for realtime video processing. Therefore, an algorithm is proposed which designs the effective local contrast modification factor according to the input images. In Peli and Lim's algorithm, local contrast components are enhanced according to their local luminance mean. In the case of enhancing hidden signals in the shaded area, if the local luminance mean of the shaded area is known, the hidden signals can be enhanced by designing the local contrast modification factor to enhance the level of the local luminance mean of the shaded area. [15]Therefore, the effective local contrast modification factor can be designed by first finding the local luminance mean of the area where the hidden local contrast signals exist, then designing a local contrast modification factor that can effectively enhance the local contrast signals of this area. The following four steps can design the local contrast modification factor. The first step detects the hidden local contrast signals that should be enhanced. One simple way to do this is to select the hidden local contrast by using the magnitude of the local contrast. This is realized by defining the two threshold levels. [17]One is the threshold level THL, which divides the noise and the hidden contrast. The other

threshold level THH divides the hidden contrast and large (visible) contrast. The hidden contrast can be selected by choosing the contrast whose magnitude is within these two threshold levels.

The second step calculates the histogram of the hidden contrast signals in the whole image for each local luminance mean level. This histogram shows the local luminance mean levels where the contrast signals which should be enhanced exist. The third step is the smoothing of the histogram. In general, histogram levels are not smooth for luminance levels, which is not appropriate for their direct use as a local contrast modification function. Therefore, smoothing of the histogram is necessary. [19]Finally, the fourth step is the weighting correction of the histogram. Local contrast signals close to the maximum or minimum level are usually suppressed by the limitation of the dynamic

range of the recorded medium. This is realized by enhancing the local contrast modification factor for those levels where the local luminance mean is very high or very low. Following is block diagram of the adaptive image enhancement algorithm.

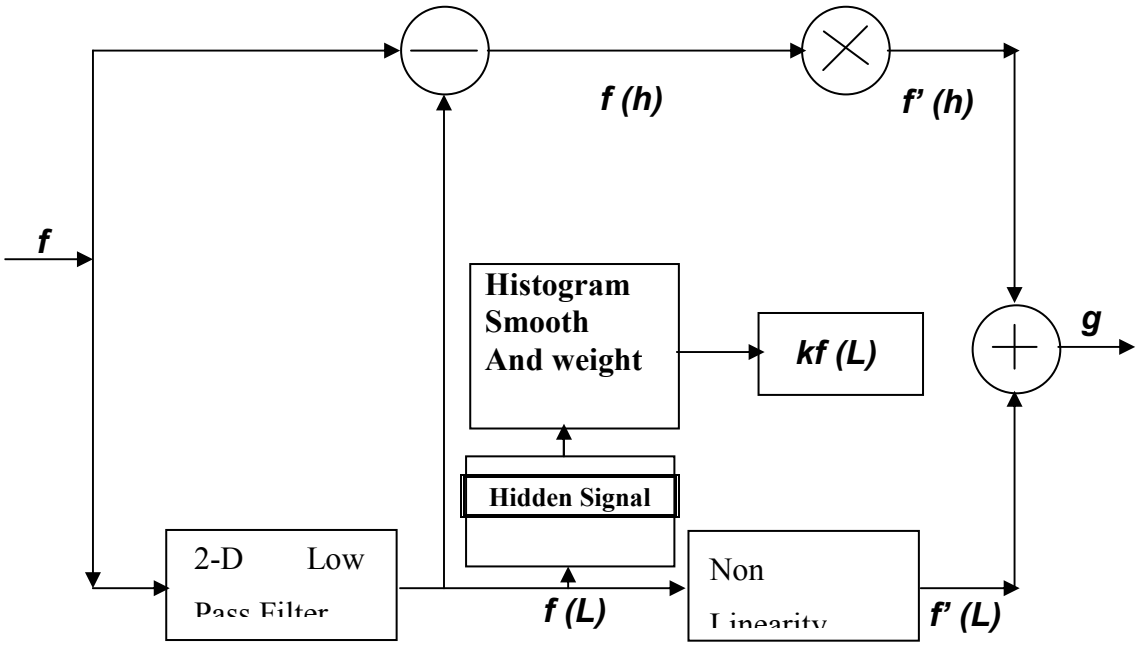


Fig 3.6: Block Diagram of Image Enhancement algorithm using histogram

3.2.2 Adaptive Histogram Equalization :

Adaptive Histogram Equalization is a good algorithm to obtain a good looking image directly from a raw grayscale image, without window and level adjustment. The AHE algorithm partitions the images into contextual regions and applies the histogram equalization to each one. This evens out the distribution of used grey values and thus makes hidden features of the image more visible.[16] The full grey spectrum is used to express the image. An example of Adaptive Histogram Equalization is shown in the figure below.

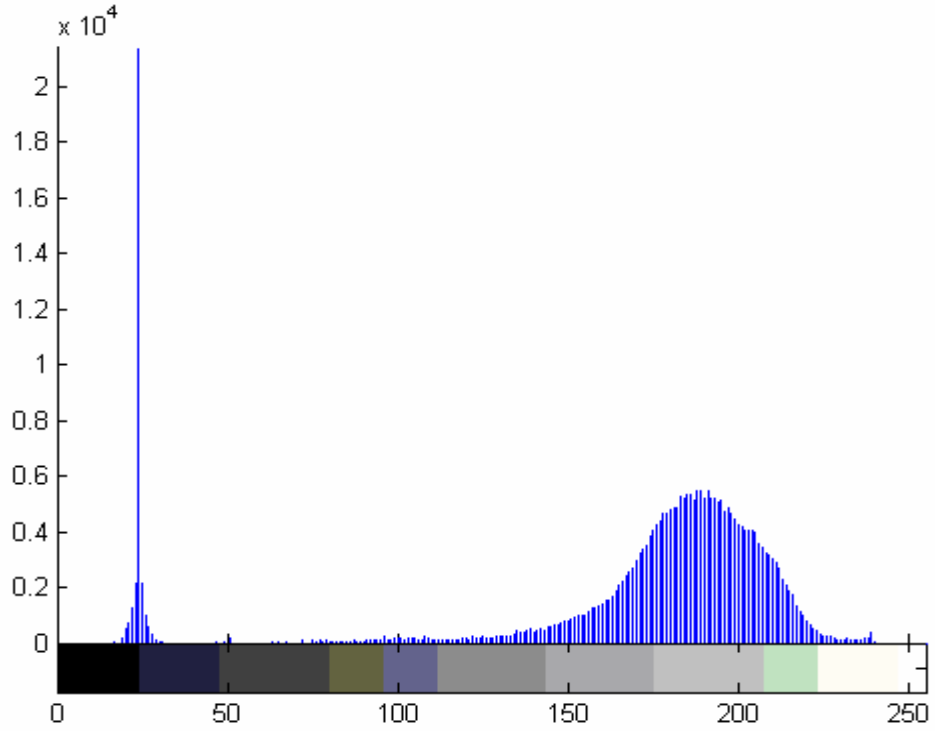


Fig 3.7: Image Histogram of initial image after resolution enhancement

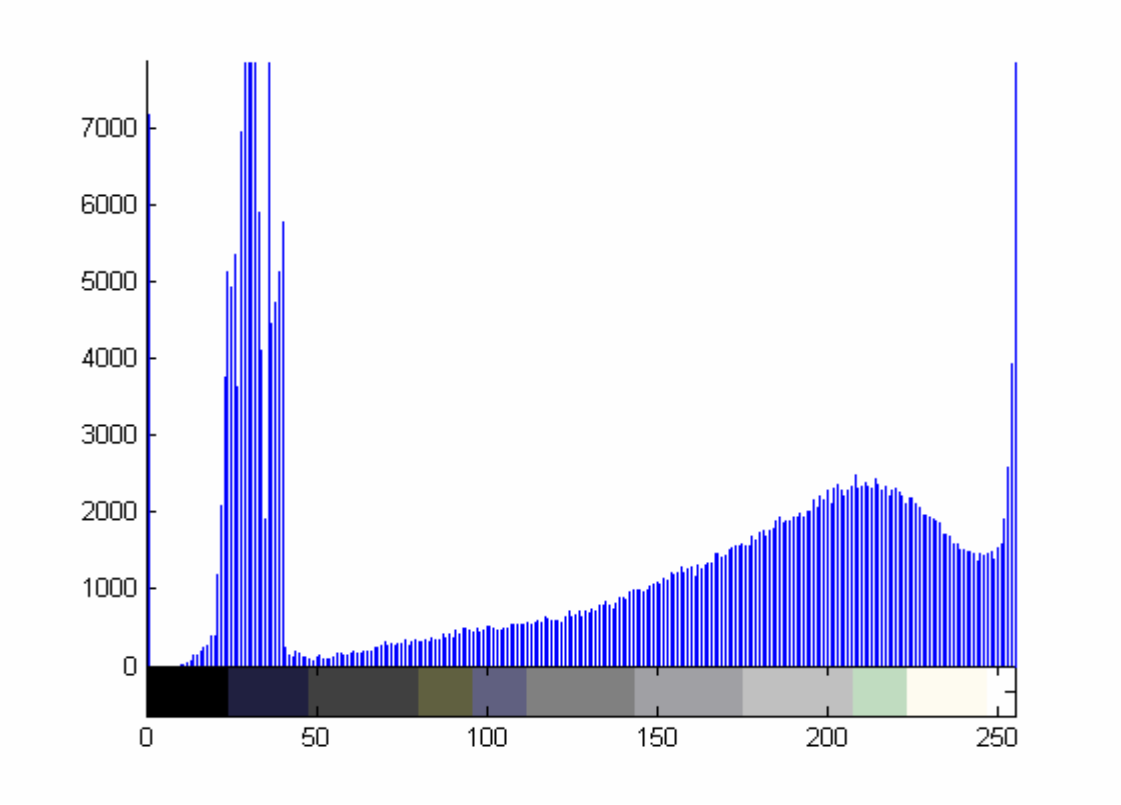


Fig 3.8: Image histogram after adaptive image enhancement

3.3 PROPOSED ALGORITHM FOR ADAPTIVE IMAGE ENHANCEMENT

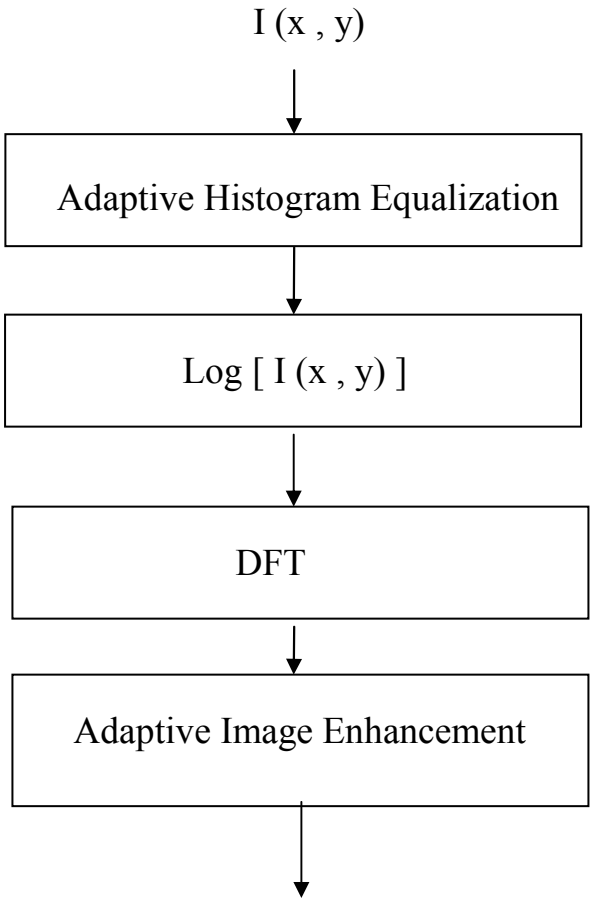


Fig 3.9: ***Block Diagram of Proposed Algorithm***

An image can be denoted by a two dimensional function of the form f(x,y). The value or amplitude of f at spatial coordinates (x,y) is a positive scalar quantity whose physical meaning is determined by the source of image.[20].When a image is generated from a physical process, its values are proportional to energy radiated by a physical source (e.g. electromagnetic waves, infrared waves). As a consequence, f(x,y) must be non zero and finite. The function f(x,y) may be characterized by two components:

1.The amount of source illumination incident on the scene being viewed

2. The amount of illumination reflected by objects in scene.

Appropriately these are called illumination and reflectance components and are denoted by i(x,y) and r(x,y) respectively. The two functions combined as a product to form f(x,y).

$$f(x,y)=i(x,y)*r(x,y)$$

The nature of i(x,y) is determined by illumination source, and r(x,y) is determined by the characteristics of the imaged objects.

The function f(x,y) can not be used directly to operate separately on the frequency components of illumination and reflectance because the Fourier transform of the product of two function is not separable. However if we define

$$z(x,y)=\ln [f(x,y)]$$
$$=\ln [i(x,y)] + \ln [r(x,y)]$$

Then

$$F\{z(x,y)\}=F\{ \ln [f(x,y)] \}$$
$$=F\{ \ln [i(x,y)] \} + F\{ \ln [r(x,y)] \}$$

Now we can operate on illuminance and reflectance components separately.

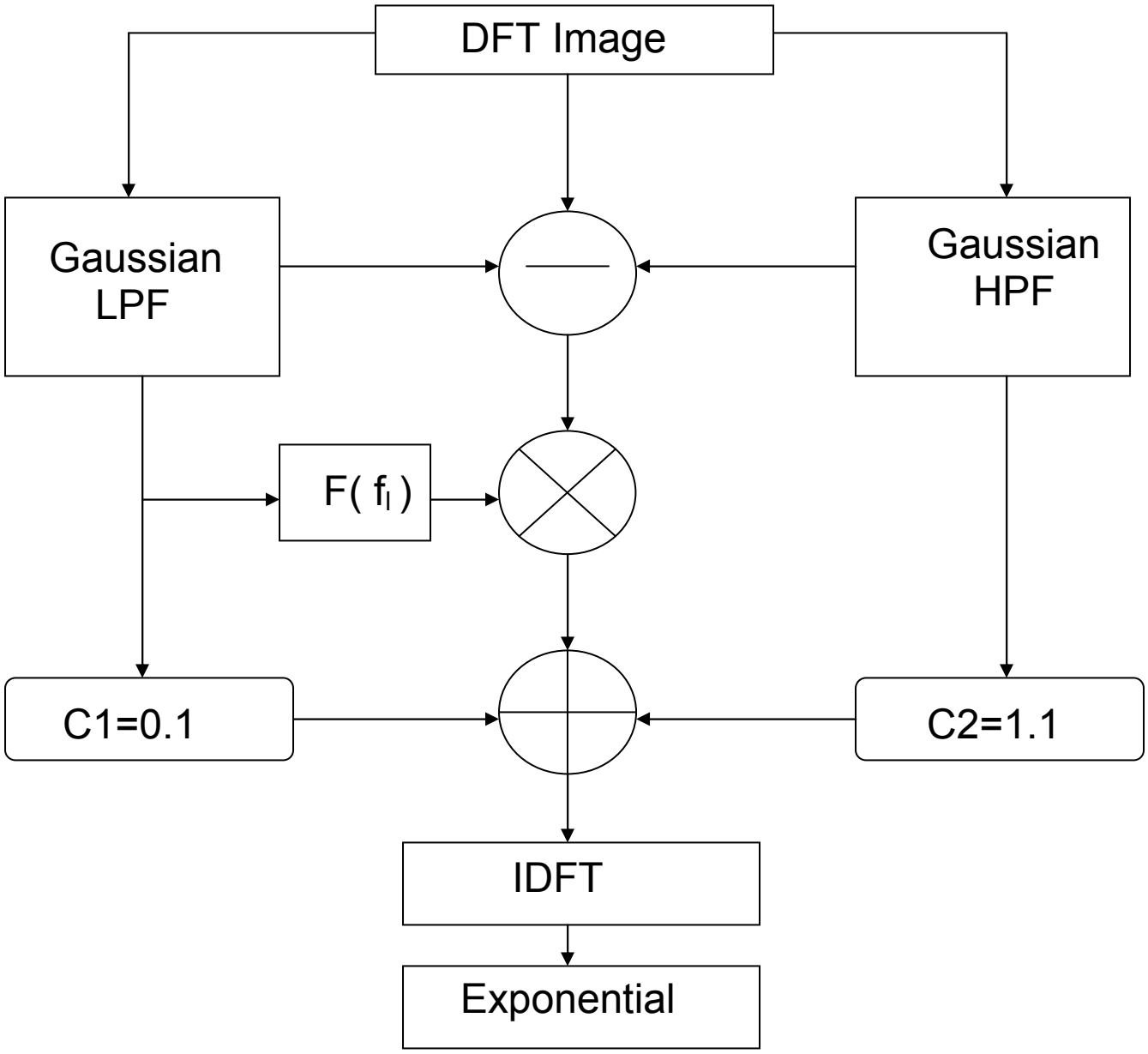
The illumination component of an image generally is characterized by slow spatial variation, while the reflectance component tends to vary abruptly, particularly at the junction of dissimilar components. These characteristics lead to associating low frequencies of the Fourier transform of the logarithm of an image with illumination and the high frequencies with reflectance[12,10]. A good deal of control can be gained over the illumination and reflectance components by defining a filter function that affects low and high frequency components of the Fourier transform in different ways. The filter function should be such that it tends to decrease the contribution made by the low frequencies (illumination) and amplify the contribution made by high frequencies (reflectance). The net result is simultaneous dynamic range compression and contrast enhancement.

In our approach we have suppressed the low frequency components by 90%.and increased the clearly visible high frequency components by 110%. The hidden frequency components are locally enhanced depending on illumination of that particular region[4]. The hidden frequency components are convolved with the function $F(f_l)$ where F is defined as

$$F(f_l)= 1+k f_l$$

The value of k is different for each 17 x 17 block.

Now the modified low frequency components, high frequency components and hidden frequency components are added to give new enhanced image in the frequency domain whose inverse transform is taken to get the image in spatial domain.[24] Finally, as z(x,y) was formed by taking the logarithm of the original image f(x,y), the inverse (exponential) operation yields the desired enhanced image. The process is shown in the block diagram below.



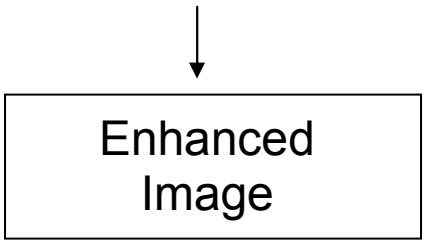


Fig 3.10 : Implementation of Adaptive Image Enhancement

Chapter 4

Object Detection

Detection of moving objects in video images is one of the most important and fundamental technologies to develop the real world computer vision systems, such as video monitoring system, intelligent-highway system, intrusion surveillance, etc. Traditionally, the most important task of monitoring safety is based on human visual observation, which is a hard work for watchmen[22,24]. Therefore, the automatic detection of moving objects is required in the monitoring system that can help a human operator, even if it cannot completely replace the human's presence[8]. To facilitate a monitoring system, efficient algorithms for detecting moving objects in video images need to be used.

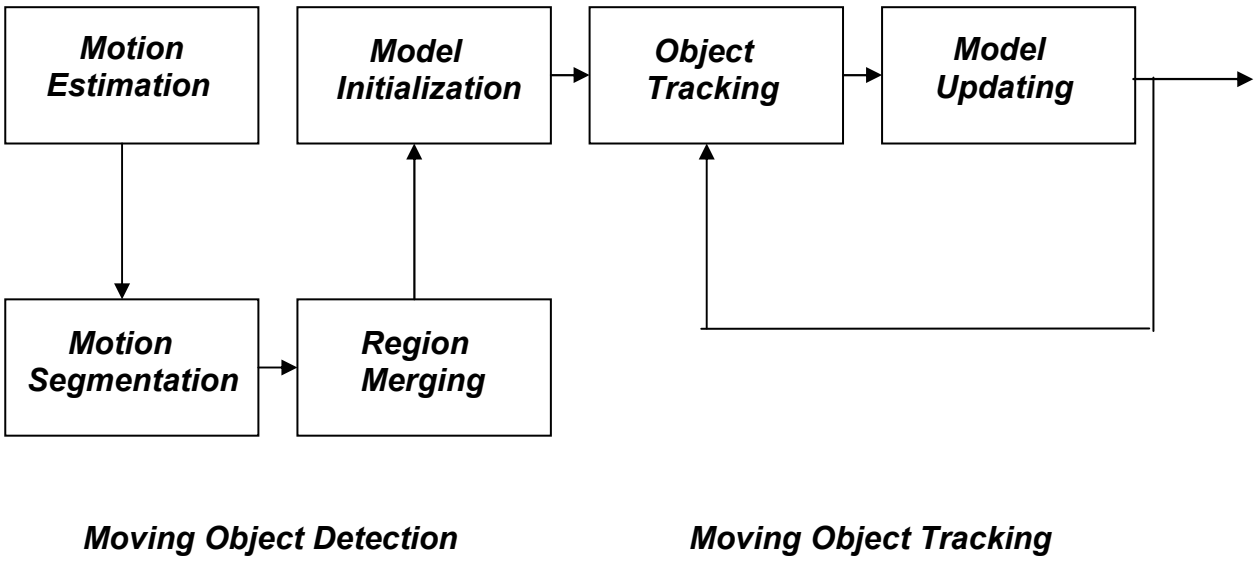


Fig 4.1 : General Model for object detection and tracking

The usual method for detecting moving objects is simple background subtraction that is to subtract current image from background image. However, there exist gradual illumination changes, sudden changes in illumination and other scene parameters alter the appearance of the background. Simple background subtraction is susceptible to these changes. And when the brightness difference

between moving objects and the background is small, it cannot detect the difference. In order to resolve these problems, some algorithms such as color based subtraction technique and the technique based on optical flows have been proposed. But the computational costs of these methods are very high and have problem in stability.[14,21]. The other commonly used method for moving objects detection is frame difference. Moving objects are detected from the difference of two consecutive frames. This approach uses the motion to distinguish moving objects from the background. So it is more efficient than the previous approaches. Frame difference approach is robust to environmental changes, however, unable to detect motionless objects [20].Currently, all the systems using frame difference approach to detect moving objects, in which, the subtraction is done pixel-wise in luminance on the whole image, even though the object moves in relatively local range. In fact, instead of processing a whole image, only the neighborhood area around the moving object needs to be processed. So computation time can be reduced.

In this, a real-time algorithm for detecting moving objects in the image sequence is proposed, which integrates the region-based frame difference with adjusted background

Subtraction[13]. In our system, images are captured with a stationary camera. The region-based frame difference is used to extract the moving objects and the adjusted background subtraction is used to get the motionless foreground objects in the scene.[11,18].The experiment results demonstrate that both the accuracy and processing speed are very promising. Furthermore, the algorithm is robust to the changes of lighting condition and camera noise.

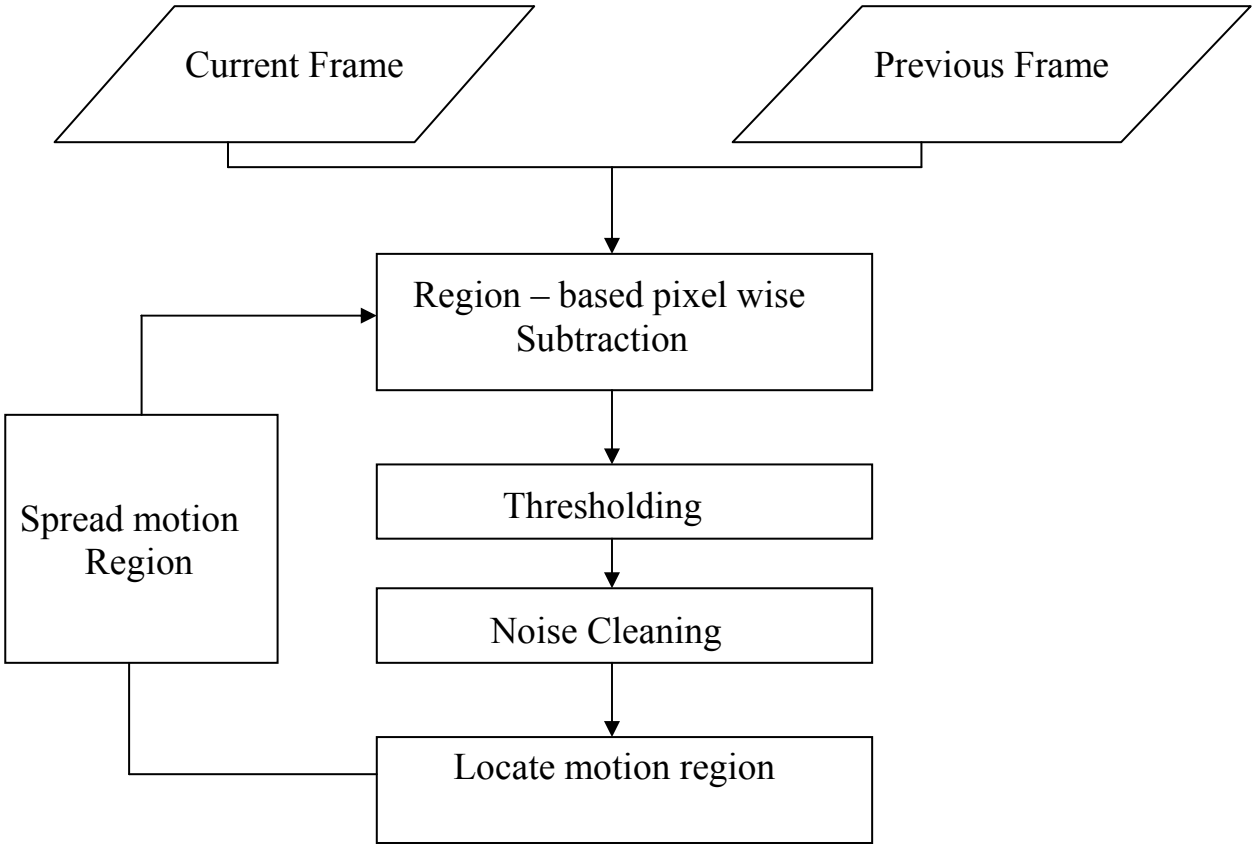


Fig 4.2: Block Diagram showing Object Tracking Algorithm

The process sequence of the algorithm is shown in Fig. The algorithm is based on a region-based frame difference motion detection technique and adjusted background subtraction. The purpose is to indicate the position of moving object in a video frame. By utilizing the region-based frame difference, it detects motion in the scene, and furthermore, it determines the position of moving regions.[7].A good detection and location algorithm should be able to detect and track the suspicious objects even when they stop. So by utilizing the adjusted background subtraction, the algorithm can also detect motionless foreground objects in the scene. [3]By using the algorithm, the moving and still objects in video images can all he detected and located.

4.1 Region-based Frame Difference

The first process of the proposed algorithm is the detection of moving objects.[23] Frame difference technique is used, which is simple and yet powerful enough, to discriminate between moving objects and non-moving ones. The frame difference method is simply finding the absolute difference between two consecutive frames**.** In stream video sequences, the frame rate is more than one shot per second. Thus, if there is any object that is in motion, it will have a slight position change and the maximum change will occur at the edges of the image since the discontinuity points are there. Supposing that the intensity of a pixel at location *(x, y)* and time ***f*** is represented by f **(x,** *y, t).* Then the difference of two consecutive frames can be represented as.

$$\mathbf{D(x,y,t) = f(x,y,t) - f(x,y,t+1)}$$

The noise occurring in D(x,y,t) is removed by convolving it with gaussian low pass filter. Since noise is made up of high frequency components, so most of noise is removed. After that thresholding is done i.e. selecting maximum value pixel out of all pixels and making it 1, rest are made 0. In this way coordinates of point that is one are found out and hence object is located[6]. After that region of interest is marked around that point and in next iteration it is expected that object will be located within that region (assuming motion of object is smooth and not abrupt). Therefore the coordinates of object within region of interest are taken as center for marking region of interest for next iteration. This procedure continues and with each iteration the object is always tracked. This is shown in the sequence of frames below.

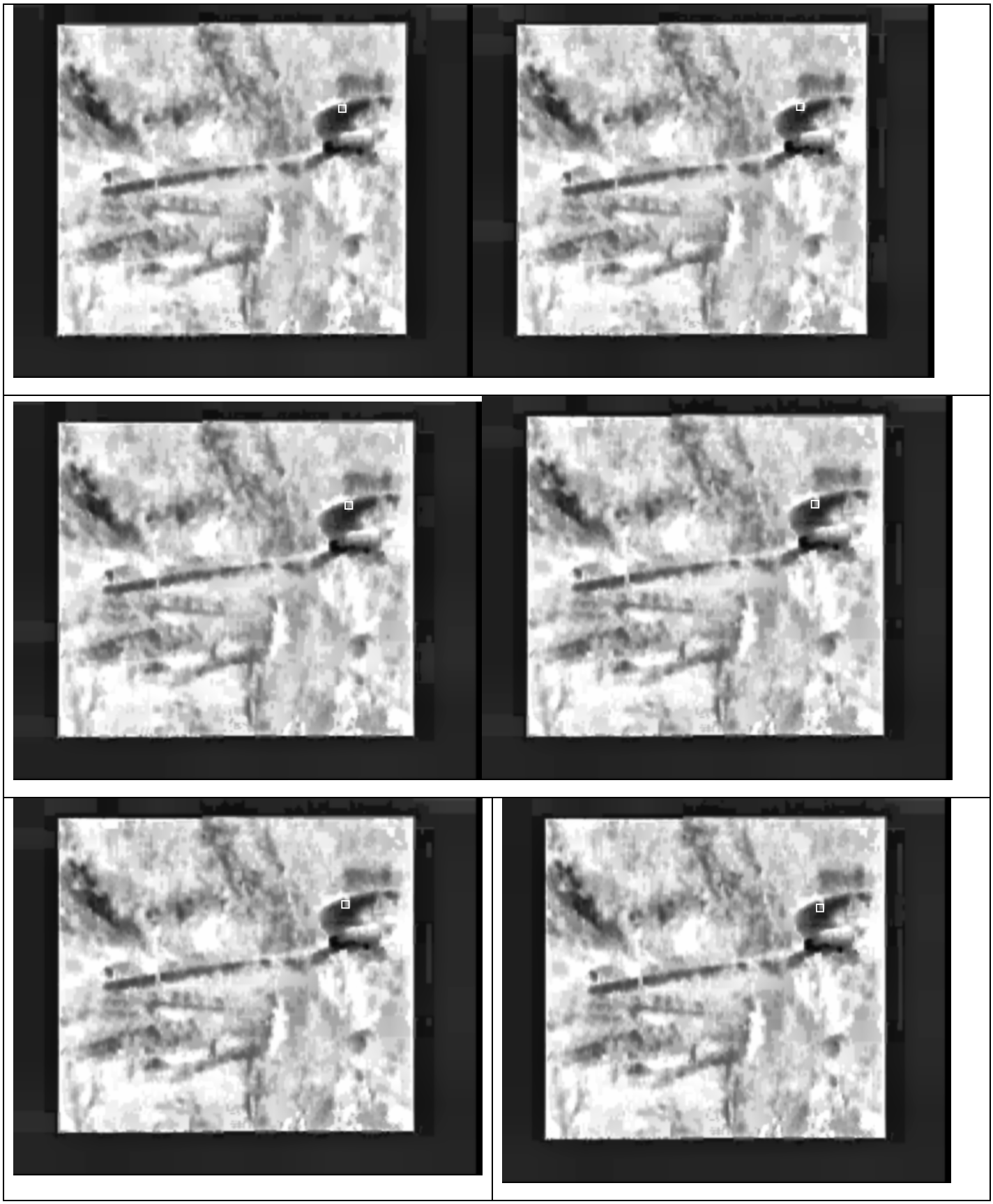


Figure 4.3: Frames during the object tracking

Chapter 5

Experimental Results

5.1 Assumptions

1. The entry coordinates of object are known.
2. The camera that is capturing videos is fixed.
3. Single object tracking is there.
4. The movement of object is smooth and not abrupt

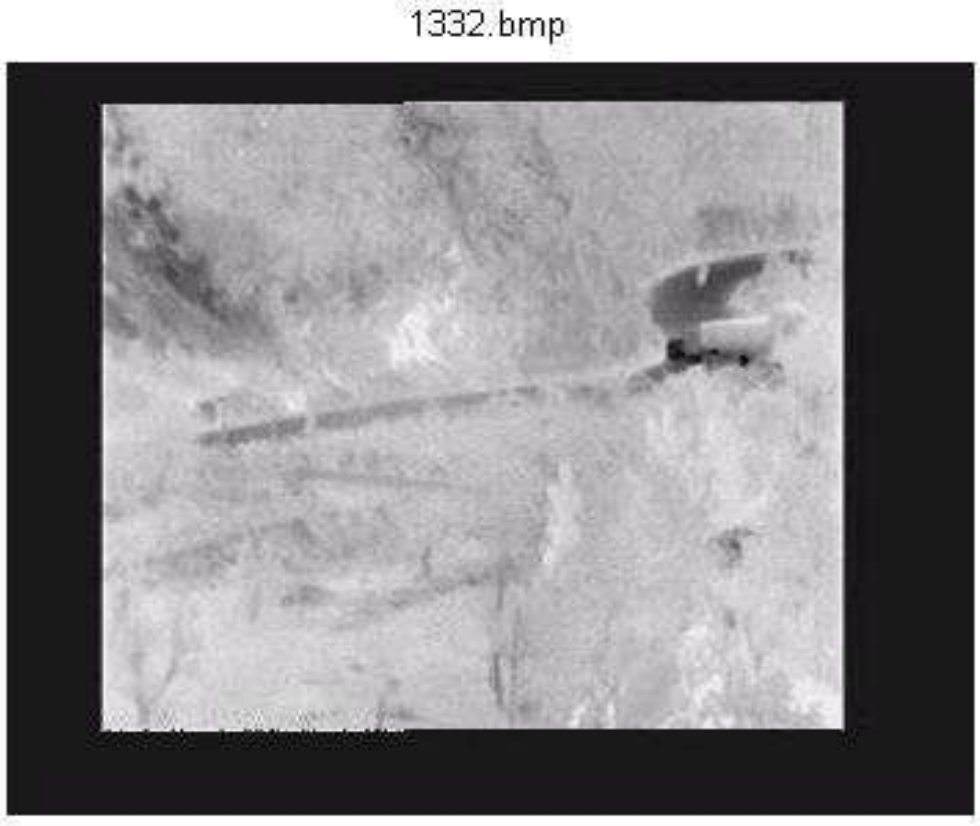


Figure 5.1:Original Image obtained from Infrared video

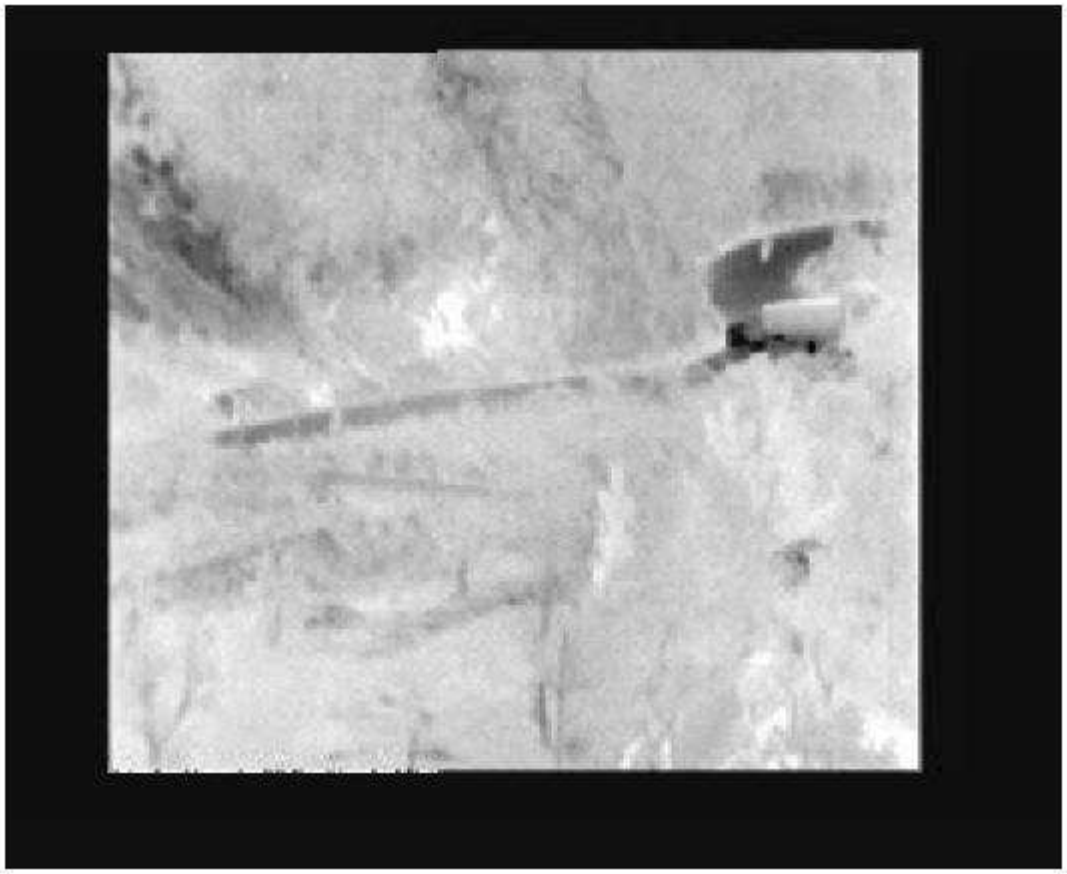


Figure 5.2 : **Image obtained after resolution enhancement**

Figure 5.3: Image obtained after adaptive image enhancement

Figure 5.4: ***Previous Frame***

Figure 5.5 ***:Current Frame***

Figure 5.6: Difference Image

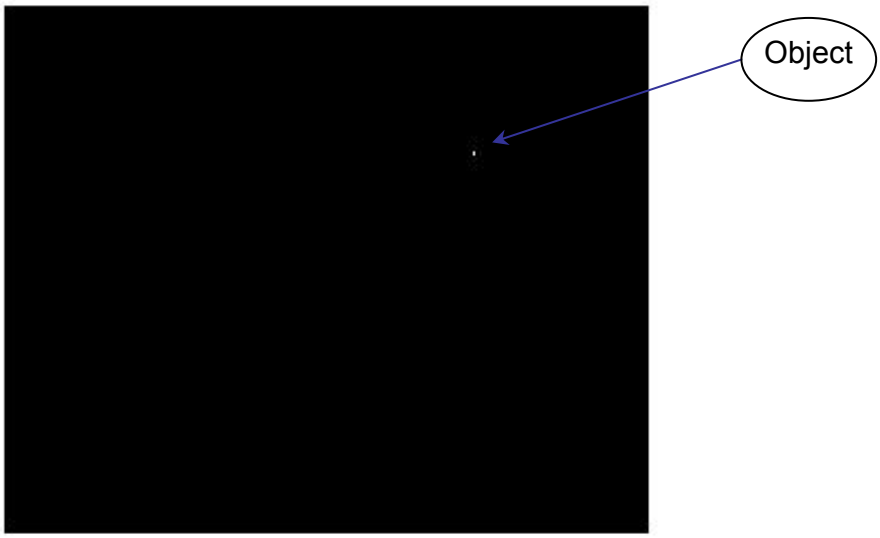


Figure 5.7 : Binary Image showing the location of the object after noise removal and thresholding

Figure 5.8: Detected Object shown by a white square

Chapter 7

Conclusion

There is a huge interest on the market to make technical equipment "smart" and "self learning". An important component in such systems is the ability for a computer to track and identify moving objects. The problem of tracking the movement of a desired object that is captured by a real time video stream is of interest because of the many applications that can be derived from it. The task addressed in this work is to track the movement of an object in the given video sequences. We suppose that this object can be easily recognized and individualized in terms of relations between its characteristics and the characteristics of the background. The objective is to provide a software that can be used on a PC for performing object tracking along with video enhancement using bilinear interpolation. Different applications can be easily derived from this tool. One of the requirements that we had specified was that this project was to run on a PC with a MATLAB installed and Windows operating system. The program is able to track moving objects and it is structured as different blocks working together. Initially the spatial resolution and the contrast of the extracted frames of the video sequence are enhanced. The position of the object is now marked manually so as to obtain the "Region of Interest"**.** The object is marked using a black square superimposed on the object which makes the position of the object clear to the observer. Now the trajectory of the object is traced assuming that the motion of the object is smooth around the region of interest. The algorithm improves the detection and location of moving objects in the video images. It is very useful for the video based applications, such as automatic video surveillance.

Chapter 8.

Future Work

In this project we have developed a system for single object detection and tracking. As video processing as a field is gaining momentum with increase in number of applications there is wide scope for further research in this area. The two main applications, which can be further worked upon, are as follows:

1. Multiple object tracking.

   In the system developed we are pursuing single object tracking by localized frame subtraction method. In case of multiple object tracking, depending on number of objects, localized frame subtraction will be carried at different points and after processing of these regions multiple objects can be tracked.

2. Tracking of object if camera is also moving along with object**.**

In cases where the camera is also moving along with the object (e.g. video shot from moving aero plane) the complications are further increased. In such cases a relative displacement between consecutive frames is to be accounted for. Thus instead of simple frame subtraction relative frame subtraction needs to be carried out. Further in this case information regarding the motion of the camera should be known fairly accurately for correct object tracking.

3. Tracking of object with moving background

This include cases such as tracking of ships or boats in sea/rivers. In this case both object and background are moving so a simple frame subtraction is not sufficient. In order to account for the motion of background , a complete knowledge of the motion of the background is necessary.

Based on our work we have gained enough exposure of the various methodologies and techniques which can be used to develop the above applications.

References

[1] Haritaoglu, D. Hanvood and L-S Davis, "Real-time Surveillance of people and their activities," IEEE Trans. Panem Analysis and Machine Intelligence. vol. 22, no. 8, pp. 809-830,2000.

[2] B. Kim., et al, " Wavelet-based vehicle Tracking for Automatic Traffic Surveillance," In Proc. IEEE Tencon. vol. I , pp. 313-316.. 2001

[3] C. Stauffer, W-E. Lgrimson, "Adaptive background mixture model; for real-time tracking," Proc .lEEE CVPR'YY, vol. 2, pp. 246-253, 1999.

[4] Y. Mae, Y. Shirai, J. Miura and Y. Kuno, "Object tracking in dunered background based an optical flow and edges," Proc IAPR ICPRY6, vol. 1, pp. 196-200, *1996.*

[5] J-B. Kim, et SI, " a Real-time moving object detection for video monitoring system," In Proc. ITC-CSCC. vol. I , pp. 454-457,2001

[6] T. Peli, J.S. Lim, "Adaptive Filtering for Image Enhancement," Proc. ICASSP81, Atlanta, pp. 1117-1120, Mar. 1981.

[7] I. Kuroda, T. Nishitani, "Adaptive Image Enhancement Using Two-Dimensional Digital Filter" (in Japanese) , Proc. 2nd DSP Symposium IEICE, pp. 211-216, Dec. 1987.

[8] I. Kuroda, T. Nishitani" Algorithm and Architecture for Realtime Adaptive Image Enhancement".

[9] F. Moscheni, S**.** Bhatacharjee, and M. Kunt, "Spatie temporal Segmentation based on regon merging," IEEE Trans. on PAMI, V01.20, pp.897-915, 1998**.**

[10] F. Dufaux, F. Morcheni, and A. Lippmaq "Spatie Tempml Segmentation Based on Motion and Static Segmentation," Proc. Of ICIP , pp.306-309, 1995.

[11] L. Torres, D. Garcia and A. Mates, "A Robust Motion Estimation and Segmentation Approach to Represent Moving Images with Layers," Int'l Conf. on ASSP, Vo1.4, pp.2661-2664, 1997.

[12] J. L. Michele. and M. H. Hayes, "A Comoressed Domain Video Object segmentation-System," Proc. of KIP,pp. 113-1 16,2002.

[13] H. Wang, and S**.** F. Chang, "A Highly Efficient System for Automatic Face Region Detection in MPEG Video," IEEE Trans. on Circuit and Systems for Video Technology, Vo1.7,pp.615-628, 1998.

[14] R. V. Babu, and K. R. Ramakrisw "Compressed Domain Motion Segmentation for Video Object Extraction," Proc. Of the IEEE ICASSP, pp.3788-3791, May, 2002.

[15] H. Zen, T Hasegawr) and ***S.*** Ozawa, "Moving object detection from MPEG wded picture," Pmc. of ICIP, pp25-29, I999

[16] R. Wan& H. J. Zhang, and Y. Q. Zhang, "A Confidence measure based movine obiect extraction system built for compressed domain,"-hc. ISCAS 2000 Geneva, Vol. ***5,*** pp.21-24,2000.

[17] J**.** F. Canny, '' A Computational Approach to Edge Detection," IEEE Trans on PAM1 Vo1.8, pp. 679698.1986.

[18] N. Bose and K. Boo. High-resolution image reconstruction with multisensors, *International Journal of Imaging Systems and Technology*, 9:294–304, 1998.

[19] R. Chan, S. D. Riemenschneider, L. Shen and Z. Shen, Tight frame: an efficient way for high-resolution image reconstruction, *Applied and Computational Harmonic Analysis*, 17:91– 115, 2004.

[20] D. Donoho and I. Johnstone, Ideal spatial adaptation by wavelet shrinkage, *Biometrika*, 81:425–455, 1994.

[21] K. Jack, *Video demystified : a handbook for the digital engineer*, HighText Publications, San Diego, Calif., 1996.
[22] B. J¨ahne, *Digital image processing*, 5th Edition, Springer - Verlag, Berlin Heidelberg, 2002.

[23] W. H. Press et al., *Numerical recipes in C++ : The art of scientific computing*, 2nd Edition, Cambridge Univ. Press, Cambridge, England, 1992.

[24] A. Ron and Z. Shen, Affine systems in *L*2(*Rd*): the analysis of the analysis operator, *Journal of Functional Analysis*, 148:408–447, 1997.

[25] R. Szeliski, Video mosaics for virtual environments, *IEEE Computer Graphics and Applications*, 22–30, March 1996.

[26] B. C. Tom and A. K. Katsaggelos, Resolution enhancement of monochrome and color video using motion compensation, *IEEE Transactions on Image Processing*, 10(2):278– 287, 2001.

[27] R. H. Chan, Z. Shen and T. Xia, Resolution enhancement

Appendix

System Requirements:

1. Visual Studio
2. MATLAB

3. Windows Operating System
4. Pentium IV processor

MATLAB and C code :

Main Program

```
%%% frame separation

!orig_frames\trial2.exe

%%% resolution enhancement

fileinfo = aviinfo('orig_frames/original.avi');

for index=1:fileinfo.NumFrames

   q1=['C:\Matlab7\work\' int2str(index) '.bmp'];

   i=imresize(rgb2gray(imread(q1)),2,'bilinear');

   q2=['c:\Matlab7\work\resol_increased\' int2str(index) '.bmp'];

   imwrite(i,q2,'bmp');

end

%%% adaptime image enhancement

for index=1:fileinfo.NumFrames

   q1=['c:\Matlab7\work\resol_increased\' int2str(index) '.bmp'];
```

```
image=imread(q1);
image=adapthisteq(image);
image=double(image);
logimage=log(image+1);
[m n]=size(image);
M=m+17-mod(m,17);
N=n+17-mod(n,17);
logimage2=zeros(M,N);
for i=1:m
   for j=1:n
      logimage2(i,j)=logimage(i,j);
   end
end
out=zeros(M,N);
for i=1:17:m
   for j=1:17:n
      im=logimage2(i:i+16,j:j+16);
      fftimage=fft2(im);
      hl=fspecial('gaussian',[17 17]);
```

```
        low=fftimage.*hl;
        low_new=0.1*low;
        hh=hl(9,9)-hl;
        high=fftimage.*hh;
        high_new=high*1.05;
        band=fftimage-low-high;
        k=1+abs((max(max(low)))/255);
        band_new=band*k;
        out_new=high_new+low_new+band_new;
        out_new2=exp(ifft2(out_new));
        out(i:i+16,j:j+16)=out_new2;
      end
   end
   q2=['c:\Matlab7\work\new2\' int2str(index) '.bmp'];
   out=uint8(out);
   imwrite(out,q2,'bmp');
end

%%%using Cpp for object detection
```

```
!image.exe

%%% making movie from the frames

M = moviein(20);

for index=1:fileinfo.NumFrames

   q=['c:\Matlab7\work\object_frames\' int2str(index) '.bmp'];

   i=imread(q);

   M(:,index) =im2frame(i);

end

%movie(M);

movie2avi(M,'object.avi');
```

Programme for resolution enhancement using Bilinear Interpolation:

```
%%%Main function for resolution enhancement

for index=1:n
```

```
% %% n is the total number of frames

q1=['E:\2\' int2str(index) '.bmp'];

%%%This function enhances resolution of images using bilinear interpolation

function [output]=resolution_enhancement(q1)

i=imread(q1);

i=rgb2gray(i);

i=double(i);

[m n]=size(i);

%%%make row,column multiple of 2

if(mod(m,2) == 1)

   m=m+1;

end

if(mod(n,2) == 1 )

   n=n+1;

end

output=zeros(2*m,2*n);

%%%output is 4 times original image



f1=1; f2=1;x=1;y=1;
```

```
for j=1:m/2
   for k=1:n/2
      temp=i(f1:f1+1,f2:f2+1);
     %%%call the function cal_coffs to calculate the coefficients
      coffs=cal_coffs(temp,x,y);
       f2=f2+2;
      for q1=0:3
         for q2=0:3
 output(x+q1,y+q2)=coffs(1)+coffs(2)*(x+q1)+coffs(3)*(y+q2)+coffs(4)*(x+q1)*(y+q2);
         end
      end
      y=y+4;
   end
   f1=f1+2; f2=1;  x=x+4; y=1;
end


q=['E:\1\' int2str(index) '.jpg'];
   out=uint8(output);
   imwrite(out,q,'jpg');
```

```
end


%This function carries out matrix inversion

function coffs = cal_coffs(temp,x,y)

f=[temp(1,1);temp(1,2);temp(2,1);temp(2,2)];

A=[1 x y x*y

   1 x y+4 x*(y+4)

   1 x+4 y (x+4)*y

   1 x+4 y+4 (x+4)*(y+4)];

coffs=inv(A)*f;
```

Programme for image enhancement using adaptive filtering:

```
for index=1:n
q1=['E:\1\' int2str(index) '.jpg'];
%%%Above command helps to read frames by name using for loop
```

```
   image=imread(q1);
%%%reads a given frame
   image=adapthisteq(image);
%% %perfoam adaptive histogram equalisation
   image=double(image);
%%%convert to double for using mathematical operators
   logimage=log(image+1);
%%%take logarithm of image, add 1 to avoid log 0
   [m n]=size(image);
%% %for computing 17*17 DFT of image make row and column a mutiple of 17 and pad
 %% %new elements with 0

   M=m+17-mod(m,17);
   N=n+17-mod(n,17);
   logimage2=zeros(M,N);
   for i=1:m
      for j=1:n
         logimage2(i,j)=logimage(i,j);
      end
   end
   out=zeros(M,N);
%%%intialise output matrix elements to 0
      for i=1:17:m
      for j=1:17:n
         im=logimage2(i:i+16,j:j+16);
%%%start selecting 17*17 matrix from total image
         fftimage=fft2(im);
%%%take it's fast Fourier transform
         hl=fspecial('gaussian',[17 17]);
%% %generate a low pass filter
         low=fftimage.*hl;
```

```
%%%convolve image with low pass file to select low frequency components
        low_new=0.1*low;
%%%supress low frequency components
        hh=hl(9,9)-hl;
%%%generate a high pass filter by subtacting low pass filter from it's highest value
        high=fftimage.*hh;
%%%convolve image with high pass filter to select high frequency components
        high_new=high*1.05;
%%%increase magnitude of high frequency components
        band=fftimage-low-high;
%%%now select band pass values i.e frequency components neither high nor low
        k=1+abs((max(max(low)))/255);
%%%magnify their magnitude
        band_new=band*k;
        out_new=high_new+low_new+band_new;
        out_new2=exp(ifft2(out_new));
%%%take it's exponential
        out(i:i+16,j:j+16)=out_new2;
%%%save result in new matrix
      end
   end
   q2=['E:\3\' int2str(index) '.bmp'];
   out=uint8(out);
   imwrite(out,q2,'bmp');
%%%save new obtained matrix in image
end
```

C code for image.exe

```
#include <math.h>
#include <string.h>
#include <stdlib.h>
```

```
#include "CImg.h"

using namespace cimg_library;

// Begin the program
int main(int argc,char **argv) {

  // Define program usage and read command line parameters
  //---------------------------------------------------

  // Display program usage, when invoked from the command line with option '-h'.

        int n=atoi(argv[0]);

  // Read image filename from the command line (or set it to "img/parrot_original.ppm" if option '-
i' is not provided).
  const char* file_i = cimg_option("-i","resol_increased/1.bmp","Input image");
  const char* file_i2 = cimg_option("-i","resol_increased/4.bmp","Input image");
  // Read pre-blurring variance from the command line (or set it to 1.0 if option '-blur' is not
provided).
  const double sigma = cimg_option("-blur",1.0,"Variance of gaussian pre-blurring");

  // Init variables
  //----------------

  const char* file_enhanced = cimg_option("-i","new2/1.bmp","Input image");
  CImg<unsigned char> image_enhanced =
CImg<>(file_enhanced).normalize(0,255).blur((float)sigma).resize(-100,-100,1,3);
```

```
 // Load an image, transform it to a color image (if necessary) and blur it with a standart deviation sigma.
 CImg<unsigned char> image = CImg<>(file_i).normalize(0,255).blur((float)sigma).resize(-100,-100,1,3);

 CImg<unsigned char> image2 = CImg<>(file_i2).normalize(0,255).blur((float)sigma).resize(-100,-100,1,3);

 // Create two display window, one for the image, the other for the color profile.
 CImgDisplay
   main_disp(image_enhanced,"Color image (Click on the refion of Interest)",0);

 CImg<unsigned char> diff(image);
 // Define colors used to plot the profile, and a hatch to draw the line
 unsigned long hatch=0xF0F0F0F0;
 const unsigned char
   red  [3] = {255,  0,  0},
   green[3] = {  0,255,  0},
   blue [3] = {  0,  0,255},
   white[3] = {255,255,255},
   grey [3] = {128,128,128};

 // Enter event loop. This loop ends when one of the two display window is closed.

 while (!main_disp.closed){

   // Handle display window resizing (if any)
   /*if (main_disp.resized) main_disp.resize().display(image);
   draw_disp.resize();*/

   if (main_disp.mouse_x>=0 && main_disp.mouse_y>=0) { // Mouse pointer is over the image
```

```
// wait for mouse click...
                while(main_disp.button != 1)
        {}
    const int
        xm = main_disp.mouse_x,                 // X-coordinate of the mouse pointer over the image
        ym = main_disp.mouse_y,                 // Y-coordinate of the mouse pointer over the image
//      xl = xm*draw_disp.dimx()/main_disp.dimx(),  // Corresponding X-coordinate of the
hatched line
        x = xm*image.dimx()/main_disp.dimx(),       // Corresponding X-coordinate of the pointed
pixel in the image
        y = ym*image.dimy()/main_disp.dimy();       // Corresponding Y-coordinate of the pointex
pixel in the image

    // Retrieve color component values at pixel (x,y)
    const unsigned int
        val_red   = image(x,y,0),
        val_green = image(x,y,1),
        val_blue  = image(x,y,2);
        const unsigned int val_gray = (image(x,y,0)+image(x,y,1)+image(x,y,2))/3;



// drawing a square where click occurs
        while(main_disp.button != 1)
        {}


                for (int i=0;i<=12;i++)
                {
                        image_enhanced(x-6+i,y-6,0) =255;
```

```
                image_enhanced(x-6+i,y-6,1) =255;
                image_enhanced(x-6+i,y-6,2) =255;

                image_enhanced(x-6+i,y+6,0) =255;
                image_enhanced(x-6+i,y+6,1) =255;
                image_enhanced(x-6+i,y+6,2) =255;

                image_enhanced(x-6,y-6+i,0) =255;
                image_enhanced(x-6,y-6+i,1) =255;
                image_enhanced(x-6,y-6+i,2) =255;

                image_enhanced(x+6,y-6+i,0) =255;
                image_enhanced(x+6,y-6+i,1) =255;
                image_enhanced(x+6,y-6+i,2) =255;

        }

        main_disp.resize().display(image_enhanced);

         //reload the image
        CImg<unsigned char> image3 =
CImg<>(file_i).normalize(0,255).blur((float)sigma).resize(-100,-100,1,3);
        int image3_gray;
        int image2_gray;
        for (  i=1;i<image.dimx();i++)          //i=x-9;i<x+10;i++
        {
                for ( int j=1;j<image.dimy();j++)     //int j=y-9;j<y+10;j++
                {
                        image3_gray = (image3(i,j,0)+image3(i,j,1)+image3(i,j,2))/3;
                        image2_gray = (image2(i,j,0)+image2(i,j,1)+image2(i,j,2))/3;
                        diff(i,j,0)=diff(i,j,1)=diff(i,j,2)=abs(image3_gray-image2_gray);
```

```
                }

            }



        /// find the max coordinates....
            int max=(diff(x,y,0)+diff(x,y,1)+diff(x,y,2))/3;
            int col_ini=x,row_ini=y;
            for (  i=x-4;i<x+4;i++)          //i=x-9;i<x+10;i++
            {
                for ( int j=y-4;j<y+4;j++)     //int j=y-9;j<y+10;j++
                {
                    if((diff(i,j,0)+diff(i,j,1)+diff(i,j,2))/3 > max)
                    {
                        max=(diff(i,j,0)+diff(i,j,1)+diff(i,j,2))/3 ;
                        col_ini=i;
                        row_ini=j;
                    }
                }

            }

for (int index = 4;index<=88;index++)
{
        char file_name1[25]="resol_increased/";
        char file_name2[25]="resol_increased/";
        char file_name3[25]="new2/";


        int current = index-3;
```

```
        char name[25];
        itoa(current,name,10);
        strcat(file_name1,name);
        strcat(file_name1,".bmp");
        const char* file_current = cimg_option("-i",file_name1,"Input image");
   CImg<unsigned char> image_current =
CImg<>(file_current).normalize(0,255).blur((float)sigma).resize(-100,-100,1,3);

// reading next image
        int next = index;
        char name2[25];
        itoa(next,name2,10);
        strcat(file_name2,name2);
        strcat(file_name2,".bmp");
        const char* file_next = cimg_option("-i",file_name2,"Input image");
   CImg<unsigned char> image_next =
CImg<>(file_next).normalize(0,255).blur((float)sigma).resize(-100,-100,1,3);



        /// difference
                for (  i=1;i<image_current.dimx();i++)          //i=x-9;i<x+10;i++
                {
                        for ( int j=1;j<image_current.dimy();j++)     //int j=y-9;j<y+10;j++
                        {
                                int       image_current_gray =
(image_current(i,j,0)+image_current(i,j,1)+image_current(i,j,2))/3;
                                int image_next_gray =
(image_next(i,j,0)+image_next(i,j,1)+image_next(i,j,2))/3;
                                diff(i,j,0)=diff(i,j,1)=diff(i,j,2)=abs(image_current_gray-
image_next_gray);
```

```
            }

        }

        /// find the max coordinates....

        int max=(diff(col_ini-2,row_ini-2,0)+diff(col_ini-2,row_ini-2,1)+diff(col_ini-
2,row_ini-2,2))/3;
        for (  i=col_ini-2;i<col_ini+2;i++)          //i=x-9;i<x+10;i++
        {

            for ( int j=row_ini-2;j<row_ini+2;j++)     //int j=y-9;j<y+10;j++
            {

                if((diff(i,j,0)+diff(i,j,1)+diff(i,j,2))/3 > max)
                {
                    max=(diff(i,j,0)+diff(i,j,1)+diff(i,j,2))/3 ;
                    col_ini=i;
                    row_ini=j;
                }
            }

        }


        //draw the rect at new position
```

```
char name3[25];
itoa(current,name3,10);
strcat(file_name3,name3);
strcat(file_name3,".bmp");
const char* file_enhanced = cimg_option("-i",file_name3,"Input image");
CImg<unsigned char> image_enhanced =
CImg<>(file_enhanced).normalize(0,255).blur((float)sigma).resize(-100,-100,1,3);


for (int i=0;i<=12;i++)
{
        image_enhanced(col_ini-6+i,row_ini-6,0) =255;
        image_enhanced(col_ini-6+i,row_ini-6,1) =255;
        image_enhanced(col_ini-6+i,row_ini-6,2) =255;

        image_enhanced(col_ini-6+i,row_ini+6,0) =255;
        image_enhanced(col_ini-6+i,row_ini+6,1) =255;
        image_enhanced(col_ini-6+i,row_ini+6,2) =255;

        image_enhanced(col_ini-6,row_ini-6+i,0) =255;
        image_enhanced(col_ini-6,row_ini-6+i,1) =255;
        image_enhanced(col_ini-6,row_ini-6+i,2) =255;

        image_enhanced(col_ini+6,row_ini-6+i,0) =255;
        image_enhanced(col_ini+6,row_ini-6+i,1) =255;
        image_enhanced(col_ini+6,row_ini-6+i,2) =255;

}
```

```
for ( i=0;i<=10;i++)

        {
                image_enhanced(col_ini-5+i,row_ini-5,0) =255;
                image_enhanced(col_ini-5+i,row_ini-5,1) =255;
                image_enhanced(col_ini-5+i,row_ini-5,2) =255;

                image_enhanced(col_ini-5+i,row_ini+5,0) =255;
                image_enhanced(col_ini-5+i,row_ini+5,1) =255;
                image_enhanced(col_ini-5+i,row_ini+5,2) =255;

                image_enhanced(col_ini-5,row_ini-5+i,0) =255;
                image_enhanced(col_ini-5,row_ini-5+i,1) =255;
                image_enhanced(col_ini-5,row_ini-5+i,2) =255;

                image_enhanced(col_ini+5,row_ini-5+i,0) =255;
                image_enhanced(col_ini+5,row_ini-5+i,1) =255;
                image_enhanced(col_ini+5,row_ini-5+i,2) =255;

        }


        char file_name4[25]="object_frames/";
        itoa(next,name,10);
        strcat(file_name4,name);
        strcat(file_name4,".bmp");
        const char* file_object = cimg_option("-o",file_name4,"Output image");
        image_enhanced.save(file_object);
```

```
} ///end of for

    }
    // Temporize event loop
    //cimg::wait(40);
  }

  return 0;
}
```

C code for frame extraction

```
// trial2.cpp : Defines the entry point for the console application.

#include "StdAfx.h"

BOOL CreateFromPackedDIBPointer(LPBYTE pDIB, int iFrame)
{
   ASSERT(pDIB!=NULL);

   //Creates a full-color (no palette) DIB from a pointer to a
   //full-color memory DIB

   //get the BitmapInfoHeader
   BITMAPINFOHEADER bih;
   RtlMoveMemory(&bih.biSize, pDIB, sizeof(BITMAPINFOHEADER));

   //now get the bitmap bits
   if (bih.biSizeImage < 1)
   {
      return FALSE;
   }

   BYTE* Bits=new BYTE[bih.biSizeImage];

   RtlMoveMemory(Bits, pDIB + sizeof(BITMAPINFOHEADER), bih.biSizeImage);

   //and BitmapInfo variable-length UDT
   BYTE memBitmapInfo[40];
   RtlMoveMemory(memBitmapInfo, &bih, sizeof(bih));
```

```
    BITMAPFILEHEADER bfh;
    bfh.bfType=19778;    //BM header
    bfh.bfSize=55 + bih.biSizeImage;
    bfh.bfReserved1=0;
    bfh.bfReserved2=0;
    bfh.bfOffBits=sizeof(BITMAPINFOHEADER) + sizeof(BITMAPFILEHEADER);
    CString FileName;
    FileName.Format("Frame-%05d.bmp", iFrame);

    FILE* fp=fopen(FileName, "wb");
    if (fp!=NULL)
    {
        fwrite(&bfh, sizeof(bfh), 1, fp);
        fwrite(&memBitmapInfo, sizeof(memBitmapInfo), 1, fp);
        fwrite(Bits, bih.biSizeImage, 1, fp);
        fclose(fp);
    }
    else
    {
        TRACE0(_T("Error writing the bitmap file"));
        return FALSE;
    }

    delete [] Bits;
    return TRUE;
}


BOOL ExtractAVIFrames(CString szFileName)
{
    AVIFileInit();
```

```
PAVIFILE avi;
int res=AVIFileOpen(&avi, szFileName, OF_READ, NULL);

if (res!=AVIERR_OK)
{
   //an error occures
   if (avi!=NULL)
      AVIFileRelease(avi);

   return FALSE;
}

AVIFILEINFO avi_info;
AVIFileInfo(avi, &avi_info, sizeof(AVIFILEINFO));

CString szFileInfo;
szFileInfo.Format("Dimention: %dx%d\n"
            "Length: %d frames\n"
            "Max bytes per second: %d\n"
            "Samples per second: %d\n"
            "Streams: %d\n"
            "File Type: %d", avi_info.dwWidth,
                avi_info.dwHeight,
                avi_info.dwLength,
                avi_info.dwMaxBytesPerSec,
                (DWORD) (avi_info.dwRate / avi_info.dwScale),
                avi_info.dwStreams,
                avi_info.szFileType);

//AfxMessageBox(szFileInfo, MB_ICONINFORMATION | MB_OK);
```

```
printf("Dimention: %dx%d\n"
            "Length: %d frames\n"
            "Max bytes per second: %d\n"
            "Samples per second: %d\n"
            "Streams: %d\n"
            "File Type: %d", avi_info.dwWidth,
                avi_info.dwHeight,
                avi_info.dwLength,
                avi_info.dwMaxBytesPerSec,
                (DWORD) (avi_info.dwRate / avi_info.dwScale),
                avi_info.dwStreams,
                avi_info.szFileType);
PAVISTREAM pStream;
res=AVIFileGetStream(avi, &pStream, streamtypeVIDEO /*video stream*/,
                                0 /*first stream*/);

if (res!=AVIERR_OK)
{
   if (pStream!=NULL)
      AVIStreamRelease(pStream);

   AVIFileExit();
   return FALSE;
}

//do some task with the stream
int iNumFrames;
int iFirstFrame;

iFirstFrame=AVIStreamStart(pStream);
if (iFirstFrame==-1)
```

```
{
   //Error getteing the frame inside the stream

   if (pStream!=NULL)
      AVIStreamRelease(pStream);

   AVIFileExit();
   return FALSE;
}

iNumFrames=AVIStreamLength(pStream);
if (iNumFrames==-1)
{
   //Error getteing the number of frames inside the stream

   if (pStream!=NULL)
      AVIStreamRelease(pStream);

   AVIFileExit();
   return FALSE;
}

//getting bitmap from frame
BITMAPINFOHEADER bih;
ZeroMemory(&bih, sizeof(BITMAPINFOHEADER));

bih.biBitCount=24;    //24 bit per pixel
bih.biClrImportant=0;
bih.biClrUsed = 0;
bih.biCompression = BI_RGB;
bih.biPlanes = 1;
```

```
    bih.biSize = 40;
    bih.biXPelsPerMeter = 0;
    bih.biYPelsPerMeter = 0;
    //calculate total size of RGBQUAD scanlines (DWORD aligned)
    bih.biSizeImage = (((bih.biWidth * 3) + 3) & 0xFFFC) * bih.biHeight ;

    PGETFRAME pFrame;
    pFrame=AVIStreamGetFrameOpen(pStream,
         NULL/*(BITMAPINFOHEADER*) AVIGETFRAMEF_BESTDISPLAYFMT*/
/*&bih*/);

    //Get the first frame
    int index=0;
    for (int i=iFirstFrame; i<iNumFrames; i++)
    {
       index= i-iFirstFrame;

       BYTE* pDIB = (BYTE*) AVIStreamGetFrame(pFrame, index);

       CreateFromPackedDIBPointer(pDIB, index);
    }

    AVIStreamGetFrameClose(pFrame);

    //close the stream after finishing the task
    if (pStream!=NULL)
       AVIStreamRelease(pStream);

    AVIFileExit();

    return TRUE;
```

```
}

int main(int argc, char* argv[])
{

        ExtractAVIFrames("clock.avi");
                printf("\nDone!\n");
        return 0;
}
```

Abstract— **In this paper we have proposed an algorithm to track moving objects in a low resolution video. In our algorithm initially the spatial resolution of each frame is increased using *bilinear interpolation*. The quality of the frames is further increased by adaptive image enhancement technique so that the *region of interest* is marked correctly. The object is now tracked by taking the difference of the consecutive frames in the region of interest.**

Kaler R.S, Sehgal Rajan

rajan.sehgal@gmail.com

Thapar Institute of Technology, Patiala.

Key Words: Bilinear Interpolation, Adaptive Image Enhancement, Region of Interest.

INTRODUCTION

With the emergence of vast multimedia applications and increasing stress on surveillance systems, video processing and object tracking have gained a status of prominence in the field Digital Image Processing. The foundation of real world computer vision systems is laid on the emerging technologies in the field of object detection and tracking. The need for automated monitoring and tracking arises form the inadequacy of human as a continuous monitor. The human factor is also prone to errors which compromise the infallibility of the system. In order to make the system robust and secure, efficient algorithms are being worked upon. The most common methodology adopted for object tracking constitutes simple background subtraction which implies subtraction of background image from the current image to obtain object information. This method though simple proves inefficient and inadequate in cases where resolution is less. It completely falters if some change occurs in the background or if the difference in illumination levels between the object and the background is indiscernible. To overcome these shortcomings a number of algorithms have been proposed such as color based subtraction techniques and optical flow method. However the applications involving color encodings are computationally more intensive and in applications such as surveillance systems the reduction in calculation time is a primary objective. We have proposed an algorithm which tries to overcome the shortfalls of the background subtraction method keeping in view the computational costs involved. In our method the object is tracked using consecutive frame subtraction method. This approach utilizes the fact that maximum intensity variation occurs at the point of motion of the object. Hence using the property of motion, an object can be successfully tracked. Along with the problem of low resolution is handled by using *bilinear interpolation. Adaptive image Enhancement* provides a simple and elegant way to improve the quality of the images thus contributing to the betterment of results. Further the algorithm has been tested on a variety of low resolution video clips and the results have been found to be satisfactory.

In the next section we discuss some of the previous works carried out in this field. This is followed by the proposed framework and details of the algorithms used in Section III. In section IV, the results obtained by applying our algorithm on three different video clips have been elucidated. In the last section the results are analyzed and a thorough conclusion is drawn based on the experimental results.

RELATED WORK

Object tracking in videos and extracting content based information from videos had been desired and worked upon for a long time. In the fields like surveillance detection of moving object gains primary importance which has boosted interest in research in this field. Shirai et al [1] described a system based on the generalized gradient model for real time optical flow extraction and tracking an object. Born [2] presented an algorithm for visually tracking moving objects in a natural environment using adaptive resolution. Jang, Kim and Choi [3] presented a system for real time tracking of objects utilizing Kalman filter approach. Jang and Choi [4] further proposed a model based tracking algorithm which can extract trajectory information of a target object by detecting and tracking a moving object from a sequence of images. In this an active model is used which characterizes regional and structural features of a target

object such as shape, texture, color, and edge. Celenk and Reza [5] have designed a system of tracking object using local windows. Kartik et al [6] have proposed a system for object tracking using block matching method.

In order to provide a simple and effective method for object detection and tracking we have proposed an algorithm which seeks to track an object based on consecutive frame subtraction and then searching for object in localized regions thus improving the efficiency of the system.

THE PROPOSED FRAMEWORK

3.1. Overview

An infrared video is obtained using a low resolution camera. Frames are extracted from the given video clip and are enhanced using various methods. The resolution of the images is increased using bilinear interpolation. This is further enhanced using adaptive histogram equalization and contrast improvement. The enhanced images are further used to track an object.

3.1.1 Resolution Enhancement using Bilinear Interpolation

Interpolation is a technique for mathematically calculating missing values between known pixels. In our work we have used a bilinear interpolation function to guess these missing values.

$$\mathbf{F(x,y) = a_o + a_x x + a_y y + a_{xy} xy}$$

For each 2X2 block in the original image we get a 4X4 block in the new image. The values of the missing pixels, are calculated by the matrix manipulation as shown.

$$\begin{pmatrix} \\ \\ \\ \\ \end{pmatrix} \begin{pmatrix} F(x_1,y_1) \\ F(x_2,y_2) \\ F(x_3,y_3) \\ F(x_4,y_4) \end{pmatrix} = \begin{pmatrix} 1 & x_1 & y_1 & x_1y_1 \\ 1 & x_2 & y_2 & x_2y_2 \\ 1 & x_3 & y_3 & x_3y_3 \\ 1 & x_4 & y_4 & x_4y_4 \end{pmatrix} X \begin{pmatrix} a_o \\ a_x \\ a_y \\ a_{xy} \end{pmatrix}$$

From above equation we can calculate the coefficients { a_o , a_x , a_y , a_{xy} } which are used to predict the missing values as shown below.

1	3
4	2

↓

1.0000	.6667	2.3333	3.0000
2.0000	2.2222	2.4444	2.6667
3.0000	2.7778	2.5556	2.3333
4.0000	3.3333	2.6667	2.0000

Fig. 3: Implementing Bilinear interpolation

3.1.2 Enhancement of the frame

Homomorphic filtering or histogram equalizations have been used for the enhancement of images with shaded regions and images degraded by cloud cover. However, it is not easy to recover the details of very high and low luminance

regions in such images when the dynamic range of the recorded medium is smaller than that of the original images. Local contrast of very high and low luminance regions cannot be well represented by the dynamic range constraints. Moreover, small local contrasts in the very high and low luminance regions cannot be well detected by the human eye. Image enhancement which enhances contrasts that were hardly seen in the original image is possible by enhancing the local contrast as well as modifying the local luminance mean for the very high and/or low luminance regions to the level where the human eye can easily detect them. This image enhancement algorithm was proposed by T. Peli and J.S. Lim [7].

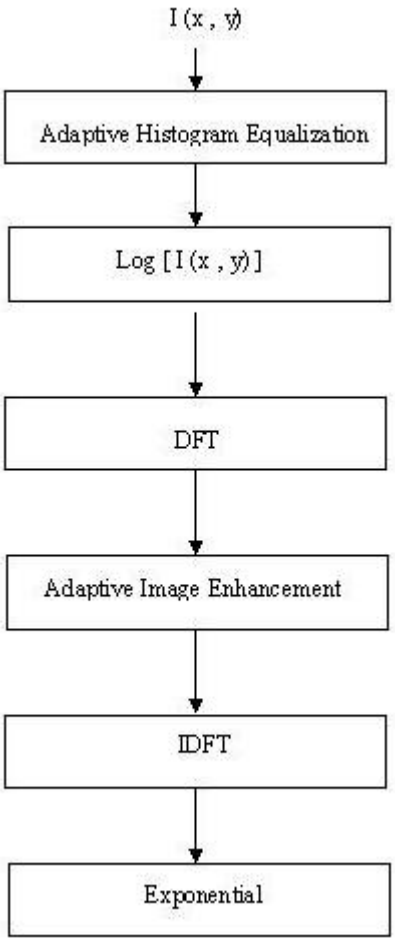


Fig 1: Block Diagram of Proposed Algorithm of image enhancement

An image can be denoted by a two dimensional function of the form f(x,y). The value or amplitude of f at spatial coordinates (x,y) is a positive scalar quantity whose physical meaning is determined by the source of image. When a image is generated from a physical process, its values are proportional to energy radiated by a physical source (e.g. electromagnetic waves, infrared waves). As a consequence, f(x,y) must be non zero and finite. The function f(x,y) may be characterized by two components:

i) The amount of source illumination incident on the scene
being viewed
ii) The amount of illumination reflected by objects in scene.
Appropriately these are called illumination and reflectance components and are denoted by i(x,y) and r(x,y) respectively. The two functions combined as a product to form f(x,y).

$$f(x,y)=i(x,y)*r(x,y)$$

The nature of i(x,y) is determined by illumination source, and r(x,y) is determined by the characteristics of the imaged objects.

The function f(x,y) can not be used directly to operate separately on the frequency components of illumination and reflectance because the Fourier transform of the product of two function is not separable. However if we define

$$z(x,y)=\ln [f(x,y)]$$
$$=\ln [i(x,y)] + \ln [r(x,y)]$$

Then

$$F\{z(x,y)\}=F\{ \ln [f(x,y)] \}$$
$$=F\{ \ln [i(x,y)] \} + F\{ \ln [r(x,y)]$$

Now we can operate on illuminance and reflectance components separately.

The illumination component of an image generally is characterized by slow spatial variation, while the reflectance component tends to vary abruptly, particularly at the junction of dissimilar components. These characteristics lead to associating low frequencies of the Fourier transform of the logarithm of an image with illumination and the high frequencies with reflectance. A good deal of control can be gained over the illumination and reflectance components by defining a filter function that affects low and high frequency components of the Fourier transform in different ways. The filter function should be such that it tends to decrease the contribution made by the low frequencies (illumination) and amplify the contribution made by high frequencies (reflectance). The net result is simultaneous dynamic range compression and contrast enhancement.

In our approach we have suppressed the low frequency components by 90%.and increased the clearly visible high frequency components by 110%. The hidden frequency components are locally enhanced depending on illumination of that particular region. The hidden frequency components are convolved with the function $F(f_l)$ where F is defined as

$$F(f_l)= 1+k f_l$$

The value of k is different for each 17 x 17 block.
Now the modified low frequency components, high frequency components and hidden frequency components are

added to give new enhanced image in the frequency domain whose inverse transform is taken to get the image in

spatial domain. Finally, as z(x,y) was formed by taking the logarithm of the original image f(x,y), the inverse

(exponential) operation yields the desired enhanced image. The process is shown in the block diagram below.

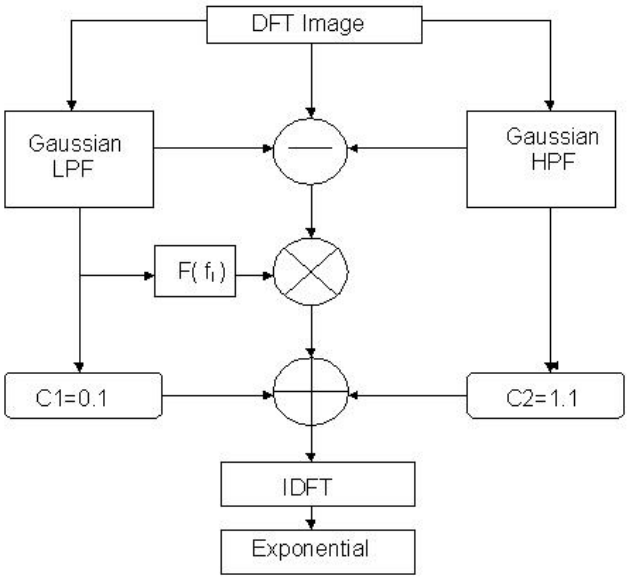


Fig 3: Implementation of Adaptive Image Enhancement

3.2.2 Object Tracking

The algorithm is based on a region-based frame difference motion detection technique. The purpose is to indicate the position of moving object in a video frame. By utilizing the region-based frame difference, it detects motion in the scene, and furthermore, it determines the position of moving regions.

The region of interest is marked in the first frame which contains the object to be detected. Frame difference technique is used, which is simple and yet powerful enough, to discriminate between moving objects and non-moving ones. The frame difference method is simply finding the absolute difference between two consecutive frames.. Supposing that the intensity of a pixel at location *(x, y)* and time ***f*** is represented by f **(x,** *y, t).* Then the difference of two consecutive frames can be represented as.

$$D(x,y,t) = f(x,y,t) - f(x,y,t+1)$$

The noise occurring in D(x,y,t) is removed by convolving it with gaussian low pass filter. Since noise is made up of high frequency components, so most of noise is removed. After that thresholding is done i.e. selecting maximum value pixel out of all pixels and making it 1, rest are made 0. In this way coordinates of point that is one are found out and hence object is located. After that region of interest is marked around that point and in next iteration it is expected that object will be located within that region (assuming motion of object is smooth and not abrupt). Therefore the coordinates of object within region of interest are taken as center for marking region of interest for next iteration. This procedure continues and with each iteration the object is always tracked. The block diagram of the object tracking algorithm is shown in the figure below.

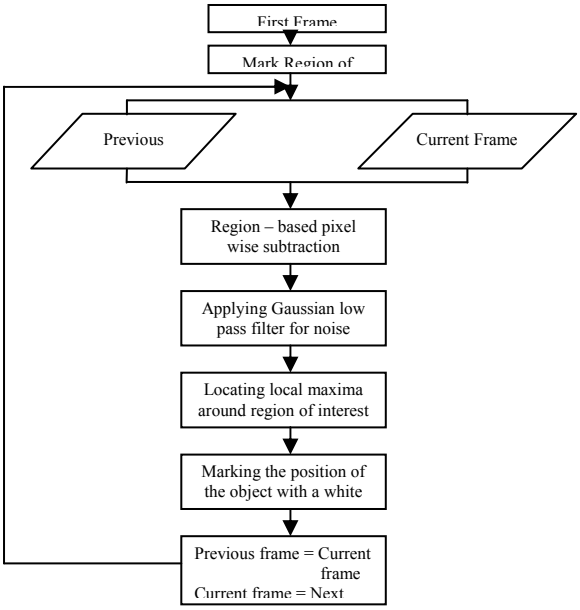


Fig 4 : Object Tracking Algorithm.

IV. EXPERIMENTS AND RESULTS

4.1 Results and analysis for Video clip

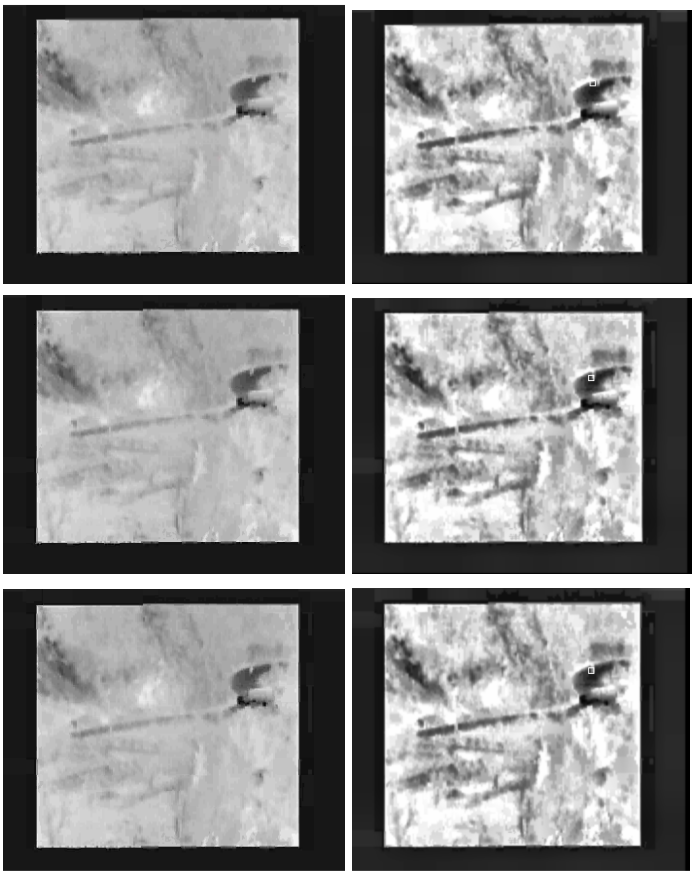


Fig 5a

Fig 5b

Fig 5a shows the extracted frames from the original video clip. Fig 5b shows the frames with the object detected marked with a white block.

The above figures show the extracted frames from the original video clip and the frames with the object marked. The histogram for the original frames and the frames from the final video are shown in figure 5c and 5d respectively .

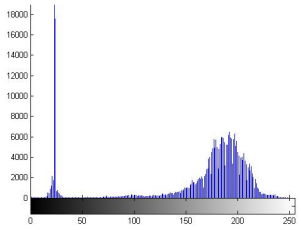


Fig 5c

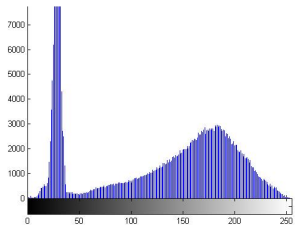


Fig 5d

calculated for the original and the enhanced frames. It was observed that the SNR values improved significantly after the application of our algorithm.

$$SNR = 20 \log_{10} (mean / variance)$$

where

$mean = \Sigma f(x, y) / M*N$

$variance = \Sigma (f(x , y)-mean)^2 / (M*N)$

The corresponding values obtained for the original frames and the enhanced frames are **4.0134dB** and **6.4042dB**.

The accuracy of our algorithm is clearly depicted by drawing the corresponding paths as shown in figure below.

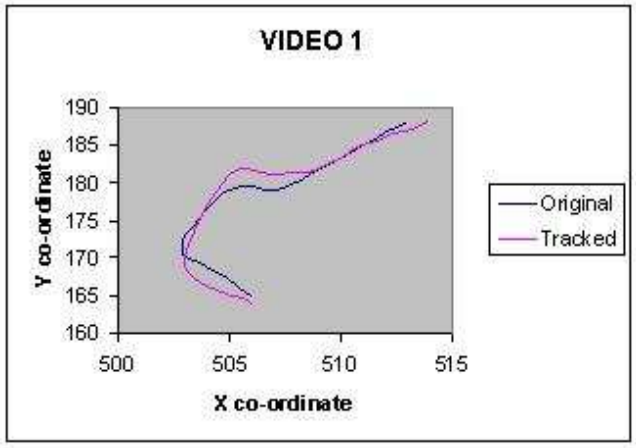


Fig 5e

In the figure above the Original line shows the actual path traversed by the object while the Tracked line shows the path obtained using our algorithm. It clearly shows the high degree of correlation between the actual path and the path obtained with our algorithm.

4.2 Some more experimental results

4.2.1 Results for video clip 2

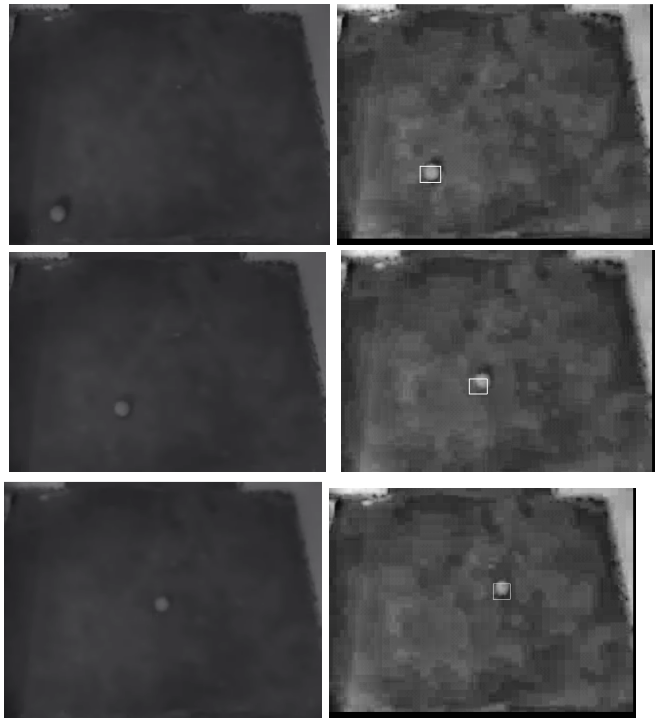


Fig 6a **Fig 6b**

Fig 6a shows the extracted frames from the original video clip. Fig 6b shows the frames with the object detected marked with a white block.

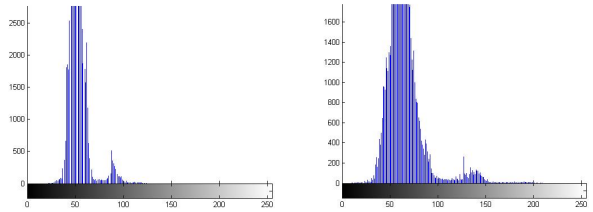


Fig 6c **Fig 6d**

Figure 6c shows the histogram of the original frame and figure 6d shows the histogram of enhanced frame.

SNR (Original frame) : 14.5426dB
SNR (Enhanced frame) : 15.5783dB

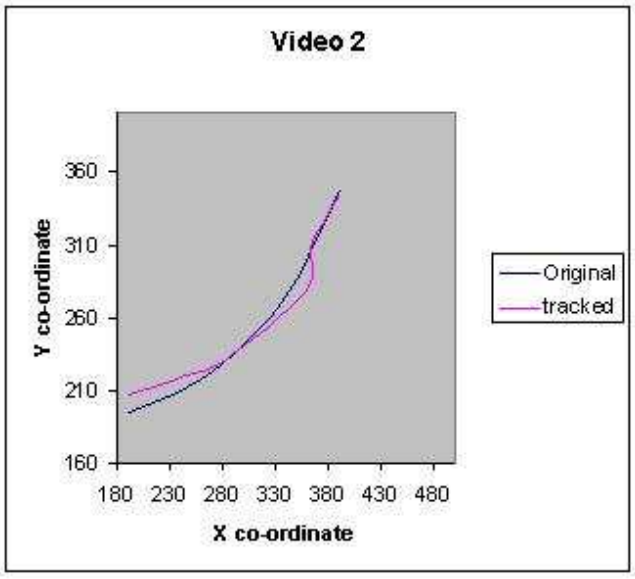


Fig 6e

Figure 6e shows the actual path along with the path tracked using our algorithm.

4.2.2 Results for video clip 3

The video clip for this experiment was of better quality and hence the results obtained for this clip are better as can be seen in the figures below:

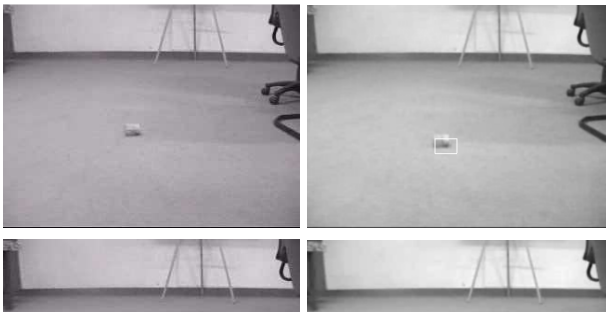


Fig 10a **Fig 10b**

Fig 7a **Fig 7b**

Fig 7a shows the extracted frames from the original video clip. Fig 7b shows the frames with the object detected marked with a white block.

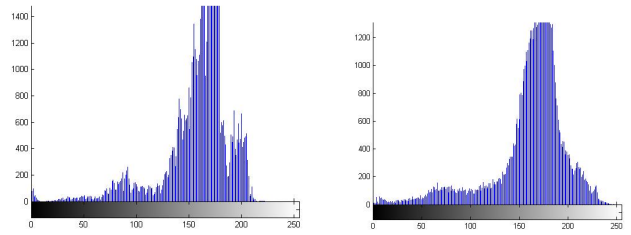


Fig 7c **Fig 7d**

Figure 7c shows the histogram of the original frame and figure 7d shows the histogram of enhanced frame.

SNR (Original frame) : 14.1578 dB
SNR (Enhanced frame) : 16.4821 dB

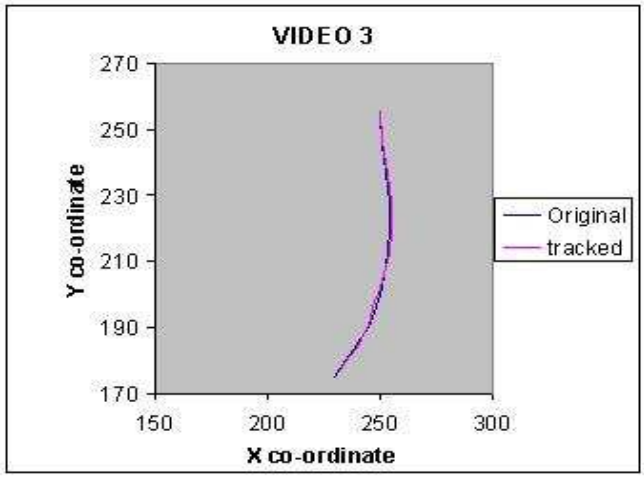


Fig 7e

Figure 7e shows the actual path along with the path tracked using our algorithm.

V. CONCLUSION

There is a huge interest on the market to make technical equipment "smart" and "self learning". An important component in such systems is the ability for a computer to track and identify moving objects. The problem of tracking the movement of a desired object that is captured by a real time video stream is of interest because of the many applications that can be derived from it. The task addressed in this work is to track the movement of an object in the given video sequences. We suppose that this object can be easily recognized and individualized in terms of relations between its characteristics and the characteristics of the background. The objective is to provide a software that can be used on a PC for performing object tracking along with video enhancement using bilinear interpolation. Different applications can be easily derived from this tool. One of the requirements that we had specified was that this project was to run on a PC with a MATLAB installed and Windows operating system. The program is able to track moving objects and it is structured as different blocks working together. Initially the spatial resolution and the contrast of the extracted frames of the video sequence are enhanced. The position of the object is now marked manually so as to obtain the "Region of Interest"**.** The object is marked using a white square superimposed on the object which makes the position of the object clear to the observer. Now the trajectory of the object is traced assuming that the motion of the object is smooth around the region of interest. The algorithm improves the detection and location of moving objects in the video images. It is very useful for the video based applications, such as automatic video surveillance.

REFERENCES

[1] Moving object perception and tracking by use of DSP
Shirai, Y.; Miura, J.; Mae, Y.; Shiohara, M.; Egawa, H.; Sasaki, S.;Computer Architectures for Machine Perception, 1993. Proceedings 15-17 Dec. 1993 Page(s):251 - 256

[2] Tracking moving objects using adaptive resolution
Born, C.;Neural Networks, 1996., IEEE International Conference on Volume 3, 3-6 June 1996 Page(s):1687 - 1692 vol.3

[3] Kalman filter incorporated model updating for real-time tracking Dae-Sik Jang; Gye-Young Kim; Hyung-Il Choi; TENCON '96. Proceedings. 1996 IEEE TENCON. Digital Signal Processing Applications Volume 2, 26-29 Nov. 1996 Page(s):878 - 882 vol.2

[4] Moving object tracking by optimizing active models
Daesik Jang; Hyung-Il Choi; Pattern Recognition, 1998. Proceedings. Fourteenth International Conference on Volume 1, 16-20 Aug. 1998 Page(s):738 - 740 vol.1

[5] Moving object tracking using local windows Celenk, M.; Reza, H.;Intelligent Control, 1988. Proceedings., IEEE International Symposium on 24-26 Aug. 1988 Page(s):180 - 185

[6] Object Tracking Using Block Matching.
H. Kartik, D. Schonfeld, P. Raffy, F. YassaIEEE conference on Image Processing, Vol. 3, pp. 945 - 948, Jul 2003.

[7] Adaptive Filtering for Image Enhancement,
T. Peli, J.S. Lim, Proc. ICASSP81, Atlanta, pp. 1117-1120, Mar. 1981.

List of Publications

1. Rajan Sehgal, Ms. Navjot Kaur, "Silicon CMOS Optical Reciever Circuits with Integrated Thin Film Semiconductor Detectors",in Proceedings of National Conference on "Trends in Electronics, Computers & Communication" (TRND'Z 06) held at Thanthai Periyar Government Institute of Technology Vellore (Tamilnadu) on 24-26 April,2006.

2. Rajan Sehgal, Dr. R.S Kaler " Video Image Enhancement & Object Tracking",

Communicated with National Conference on "Wireless Networks And Embedded Systems" (WNES-2006) to be haeld on 28th July 2006 at Chitkara Institute of Engg. & Technology, Rajpura , Patiala (Punjab)

3. Rajan Sehgal, Dr. R.S Kaler , Ms. Navjot Kaur "MEMS:An Emerging IC Technology" Communicated with National Conference on "Wireless Networks And Embedded Systems" (WNES-2006) to be held on 28th July 2006 at Chitkara Institute of Engg. & Technology, Rajpura , Patiala (Punjab)