THERMO VISUAL VIDEO FUSION

Anish Modi

Yash Motani

Pooja Manglani

Saurin Shah

Guided by

Prof. Anjali Malviya

Department of Information Technology
Thadomal Shahani Engineering College
University of Mumbai

THERMO-VISUAL VIDEO FUSION

Project-B Report

Submitted In Partial Fulfillment Of The
Requirements For The Degree Of

BACHELOR OF ENGINEERING
IN
INFORMATION TECHNOLOGY
BY

Group No: 24

Seat No.	Name
066093	Anish Modi
066094	Yash Motani
066095	Pooja Manglani.
066096	Saurin Shah

Under the guidance
Of
Prof. Anjali Malviya
Assistant Professor, Department of Information Technology

Thadomal Shahani Engineering College
University of Mumbai

Project entitled: Thermo-Visual Video Fusion.

Submitted by: Anish Modi
Yash Motani
Pooja Manglani
Saurin Shah

Submitted in Partial Fulfillment of the
Requirements for the Degree of
BACHELOR OF ENGINEERING
IN
INFORMATION TECHNOLOGY

Guide: _____

Examiner 1: _____

Examiner 2: _____

_____
Head of Department

_____
Principal

Date: _____

Table of Contents

1. Introduction

1.1 Abstract

In this report we consider the problem of fusing two video streams acquired by an RGB camera and a sensor operating in the Long Wave Infrared (LWIR). The application of interest is area surveillance and the fusion process aims at enhancing the human perception of the monitored scene. We propose a fusion procedure where the background and the moving objects are separated and fused by means of different strategies. With respect to standard video fusion techniques this approach has the advantage of reducing the computational load and mitigating the rapid brightness variations in the fused video. It is also less sensitive to the presence of noise. We discuss the experimental results obtained on a typical area surveillance scenario and demonstrate the effectiveness of the proposed method. For this purpose, the analysis is carried out both subjectively, in terms of visual quality of the fused video stream and objectively, in terms of standard image quality indexes.

1.2 Introduction

The objective in video fusion is to reduce uncertainty and minimize redundancy in the output while maximizing relevant information particular to an application or task. Given the same set of input videos, different fused videos may be created depending on the specific application and what is considered relevant information. There are several benefits in using videos fusion: wider spatial and temporal coverage, decreased uncertainty, improved reliability, and increased robustness of system performance of particular interest is the application of video fusion in real-time multi-sensor imaging such as that used in military, civilian avionics, medical imaging and surveillance. In this context continuous (video) streams of up to 30 images from different sensors are fused into a single output fusion stream each second to increase the examination accuracy and evaluation specificity.

1.3 Scope

Different videos of the same scene provides different information. A particular video may not contain information that another video may contain. Using video fusion we can take the important and relevant information from individual videos and then merge or fuse it. By doing so we not only edit the unnecessary part but we understand the video with greater detail. For example, the IR video does not contain detail of the background of the scene but it contains information about the movement of objects. Whereas video captured from a CCTV may not be able to distinguish the object with the background in certain situation but it contains information about the background of the scene. By fusing these two images we get a video in which we have background details as well as details of motion of the object. This application is extended in fusing videos by various satellites which include IR spectrum, images, etc.

1.4 Motivation

The main motivation behind this report is the surveillance and night time vision. As we know during dim light we cannot see moving figures clearly through a visual camera though we can see the background and through an IR camera we cannot see the background well but we can see the moving object. Hence we generate a fused video which contains important information from both the cameras. This is a big boon in the field of surveillance at night to track intruders.

2. Video Fusion

2.1 Introduction

Video fusion is used to reduce uncertainty and minimize redundancy in the output while maximizing relevant information particular to an application or task. Given the same set of input videos, different fused videos may be created depending on the specific application and what is considered relevant information.

2.1.1 Video

Video refers to recording, manipulating, and displaying moving images, especially in a format that can be presented on a television. It refers to displaying images and text on a computer monitor. The video adapter, for example, is responsible for sending signals to the display device. A recording produced with a video recorder (camcorder) or some other device that captures full motion.

2.1.2 Number of Frames/Second

Frame rate, the number of still pictures per unit of time of video, ranges from six or eight frames per second (frame/s) for old mechanical cameras to 120 or more frames per second for new professional cameras. PAL and SECAM standards specify 25 frame/s, while NTSC specifies 29.97 frame/s. Film is shot at the slower frame rate of 24photograms/s, which complicates slightly the process of transferring a cinematic motion picture to video. The minimum frame rate to achieve the illusion of a moving image [persistence of vision] is about fifteen frames per second.

2.1.3 Interlacing

Video can be interlaced or progressive. Interlacing was invented as a way to achieve good visual quality within the limitations of a narrow bandwidth. The horizontal scan lines of each interlaced frame are numbered consecutively and partitioned into two fields: the odd field (upper field) consisting of the odd-numbered lines and the even field (lower field) consisting of

the even-numbered lines. NTSC, PAL and SECAM are interlaced formats. Abbreviated video resolution specifications often include an i to indicate interlacing. For example, PAL video format is often specified as 576i50, where 576 indicates the vertical line resolution, i indicates interlacing, and 50 indicates 50 fields (half-frames) per second.

2.1.4 Aspect Ratio

Comparison of common cinematography and traditional television (green) aspect ratios. Aspect ratio describes the dimensions of video screens and video picture elements. All popular video formats are rectilinear, and so can be described by a ratio between width and height. The screen aspect ratio of a traditional television screen is 4:3, or about 1.33:1. High definition televisions use an aspect ratio of 16:9, or about 1.78:1. The aspect ratio of a full 35 mm film frame with soundtrack (also known as the Academy ratio) is 1.375:1.Ratios where the height is taller than the width are uncommon in general everyday use, but do have application in computer systems where the screen may be better suited for a vertical layout. The most common tall aspect ratio of 3:4 is referred to as portrait mode and is created by physically rotating the display device 90 degrees from the normal position. Other tall aspect ratios such as 9:16 are technically possible but rarely used.

2.1.5 Bit Rate

Bit rate is a measure of the rate of information content in a video stream. It is quantified using the bit per second (bit/s or bps) unit or Megabits per second (Mbit/s). A higher bit rate allows better video quality. For example Video, with a bit rate of about 1 Mbit/s, is lower quality than DVD, with a bit rate of about 5 Mbit/s. HDTV has a still higher quality, with a bit rate of about 20 Mbit/s. Variable bit rate (VBR) is a strategy to maximize the visual video quality and minimize the bit rate. On fast motion scenes, a variable bit rate uses more bits than it does on slow motion scenes of similar duration yet achieves a consistent visual quality. For real-time and non-buffered video streaming when the available bandwidth is fixed, e.g. in videoconferencing delivered on channels of fixed bandwidth, a constant bit rate (CBR) must be used.

2.2 Applications

In recent years, significant attention has focused on multisensory data fusion for both military and nonmilitary applications [2]. Data fusion techniques combine data from multiple sensors and related information to achieve more specific inferences than could be achieved by using a single, independent sensor.

The concept of multisensory data fusion is hardly new. As humans and animals have evolved, they have developed the ability to use multiple senses to help them survive. For example, assessing the quality of an edible substance may not be possible using only the sense of vision; the combination of sight, touch, smell, and taste is far more effective. Similarly, when vision is limited by structures and vegetation, the sense of hearing can provide advanced warning of impending dangers. Thus, multisensory data fusion is naturally performed by animals and humans to assess more accurately the surrounding environment and to identify threats, thereby improving their chances of survival.

While the concept of data fusion is not new, the emergence of new sensors, advanced processing techniques, and improved processing hardware have made real-time fusion of data increasingly viable. Applications for multi-sensor data fusion are widespread. Military applications include automated target recognition (e.g., for smart weapons), guidance for autonomous vehicles, remote sensing, battlefield surveillance, and automated threat recognition systems, such as identification-friend-foe-neutral (IFFN) systems. Nonmilitary applications include monitoring of manufacturing processes, condition based maintenance of complex machinery, robotics, and medical applications. Techniques to combine or fuse data are drawn from a diverse set of more traditional disciplines, including digital signal processing, statistical estimation, control theory, artificial intelligence, and classic numerical methods. Historically, data fusion methods were developed primarily for military applications.

However, in recent years, these methods have been applied to civilian applications and a bidirectional transfer of technology has begun.

3. Thermo-Visual Video Fusion

3.1 Thermo-Visual Video Fusion Using Probabilistic Graphical Model for Human Tracking

This technique presents a graphical model approach that fuses thermal infrared (IR) and visible spectrum video for human tracking. It is able to capture and exploit the statistical structure of the IR and the visible data separately, as well as their mutual dependencies. Model parameters are learned from data using the expectation maximization (EM) algorithm.

A challenging domain of vital military importance is the surveillance of non cooperative and camouflaged targets within cluttered outdoor setting. The advantages of capturing thermal infrared (IR) video in parallel with standard visible spectrum CCTV for human tracking in complex outdoor environments. These two sensors are intuitively complementary, since they capture object information in emitted and reflected radiation, respectively. IR video aids visible analysis in low-lighting conditions and when an object has similar color to the background. Visible spectrum CCTV can provide chrominance information on the object appearance. An ensemble of linear classifiers are trained online in a least-squares manner, to distinguish between object and background pixel features. This allows any type of pixel data to be added to assist the tracking. They link foreground regions in consecutive frames and do not model the appearances of tracked objects, therefore their method requires many complex ad-hoc rules to account for the splitting and merging of foreground regions. A probabilistic graphical model is generated to describe not only the statistics of each individual modality but also the joint statistical characteristics of them. In other words, the correlation between the two modalities can be exploited in a systematic manner by using the proposed model. The model parameters are learned from video sequences using an expectation maximization (EM) algorithm. The object trajectory is then inferred from the data via Bayes's rule.

3.2 Thermal-Visible Video Fusion for Moving Target Tracking and Pedestrian Classification

This presents a fusion-tracker and pedestrian classifier for color and thermal cameras. The tracker builds a background model as a multi-modal distribution of colors and temperatures. It is constructed as a particle filter that makes a number of informed reversible transformations to sample the model probability space in order to maximize posterior probability of the scene model. Observation likelihoods of moving objects account their 3D locations with respect to the camera and occlusions by other tracked objects as well as static obstacles. After capturing the coordinates and dimensions of moving objects we apply a pedestrian classifier based on periodic gait analysis. To separate humans from other moving objects, such as cars, we detect, in human gait, a symmetrical double helical pattern, that can then be analyzed using the Frieze Group theory. The results of tracking on color and thermal sequences demonstrate that our algorithm is robust to illumination noise and performs well in the outdoor environments.

4. DESIGN AND IMPLEMENTATION

4.1 Block Diagram

[Fig 3] shows the flow chart of the proposed VF system. Input data are a thermal sequence (8 12 micrometers), and an RGB video. The two cameras are static and the videos are aligned. Background estimation is performed in both the input sequences.

For this purpose, we use only the luminance component in the RGB video. The segmentation of the scene into background and moving objects is accomplished by the Moving Object Extraction (MOE) block, which only uses the IR video and the IR estimated background. The advantage of applying MOE to the IR sequence is that its quality is less affected by low visibility conditions, such as fog, night and rain. The MOE block returns a binary mask that is applied both to the RGB (luminance) and to the IR video. Fusion is performed separately on the background and on the moving objects, and a single video is finally recomposed. The process ends with a color transfer from the visible camera, to the fused sequence. In the next section we describe in more detail the background extraction methodology and the fusion technique.

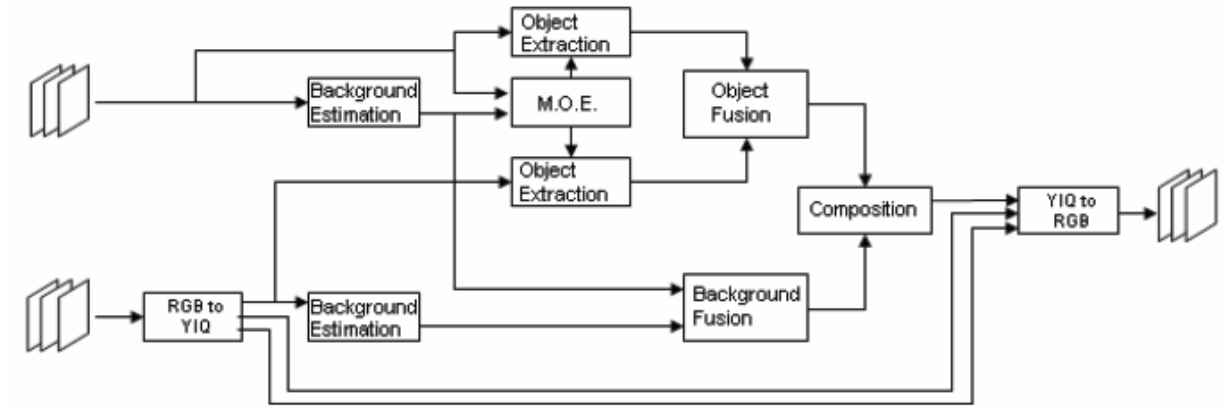


Figure 1. Flow Chart of Proposed System.

4.1.1 Background Estimation

In the literature, many techniques for background estimation have been presented. The popular median filtering algorithm, which sets the estimated background equal to the median value of the input frames, assumes that the background is visible for at least half the filter length. In the extraction process computes a map analyzing the history of each pixel, in terms of temporal stability and variability. Jabri et al. combine the information associated with color and edges in a frame. In a labeling algorithm is proposed. First, it assigns a cost function to each pixel on the basis of its texture and temporal variation, then it chooses the intensity value that minimizes the cost function. The performance of the techniques described above is strongly affected by the proper setting of design parameters. Their values, which depend on the specific properties of the observed video, are often chosen empirically and may dramatically influence the final result. For this reason we propose a methodology which does not require the setting of any threshold or parameter. The adopted technique estimates the background on a given pixel as the most probable value, in terms of number of occurrences in a temporal window.

(i) Periodic background estimation.

For each pixel of the observed scene, we count the number of occurrences of each intensity level during a time interval *T*. The occurrences are stored in a two dimensional matrix *N* of dimensions 256 x (*Nc.Nr*), where *Nc* and *Nr* are the number of columns and rows of the video frames, respectively, and 256 is the number of intensity levels of the input image. The element $N(i, k)$ represents the number of times the pixel $x(r, c)$ of position $i = (r-1) \cdot Nc + c$, has assumed the intensity value *k*, during the current time slot. Every second the background is estimated in each pixel by selecting the intensity value with the maximum number of occurrences. In other words, every second, the background image is computed as follows:

$$B(r,c) = \max_k \{N(i,k)\} \qquad \forall r, c = i$$

Finally, at the end of every second the matrix *N* is set to zero, and the procedure is started again, maintaining the estimated background for the next second.

4.2 Conversion From RGB To NTSC(YIQ) And From NTSC(YIQ) To RGB

(a) RGB to YIQ

yiqmap = rgb2ntsc(rgbmap) converts the m-by-3 RGB values in rgbmap to NTSC color space. yiqmap is an m-by-3 matrix that contains the NTSC luminance (Y) and chrominance (I and Q) color components as columns that are equivalent to the colors in the RGB colormap. YIQ = rgb2ntsc(RGB) converts the true-color image RGB to the equivalent NTSC image YIQ. In the NTSC color space, the luminance is the grayscale signal used to display pictures on monochrome (black and white) televisions. The other components carry the hue and saturation information. rgb2ntsc defines the NTSC components using

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.211 & -0.523 & 0.312 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

(b) YIQ to RGB

rgbmap = ntsc2rgb(yiqmap) converts the m-by-3 NTSC (television) color values in yiqmap to RGB color space. If yiqmap is m-by-3 and contains the NTSC luminance (Y) and chrominance (I and Q) color components as columns, then rgbmap is an m-by-3 matrix that contains the red, green, and blue values equivalent to those colors. Both rgbmap and yiqmap contain intensities in the range 0 to 1.0. The intensity 0 corresponds to the absence of the component, while the intensity 1.0 corresponds to full saturation of the component. RGB = ntsc2rgb(YIQ) converts the NTSC image YIQ to the equivalent true-color image RGB. ntsc2rgb computes the RGB values from the NTSC components

4.3 Fusion Techniques

(a) Block Fusion.

In this technique we consider the neighboring pixel values and find the average of them. The maximum average value of the 2 images is taken in the output image.

(b) Simple Average Fusion.

In the output image, for each pixel we take the average of the pixel values of the input images

.

(c) Maximum Fusion

The maximum pixel value of the 2 images is taken as the output image pixel value

(d) Wavelet Fusion

This consists of 2 steps namely decomposition and reconstruction.

(i) Decomposition

Two-dimensional Discrete Wavelet Transform (DWT) leads to a decomposition of approximation coefficients at level j in four components: the approximation at level j + 1, and the details in three orientations (horizontal, vertical, and diagonal). The following chart describes the basic decomposition steps for images [Fig 4].

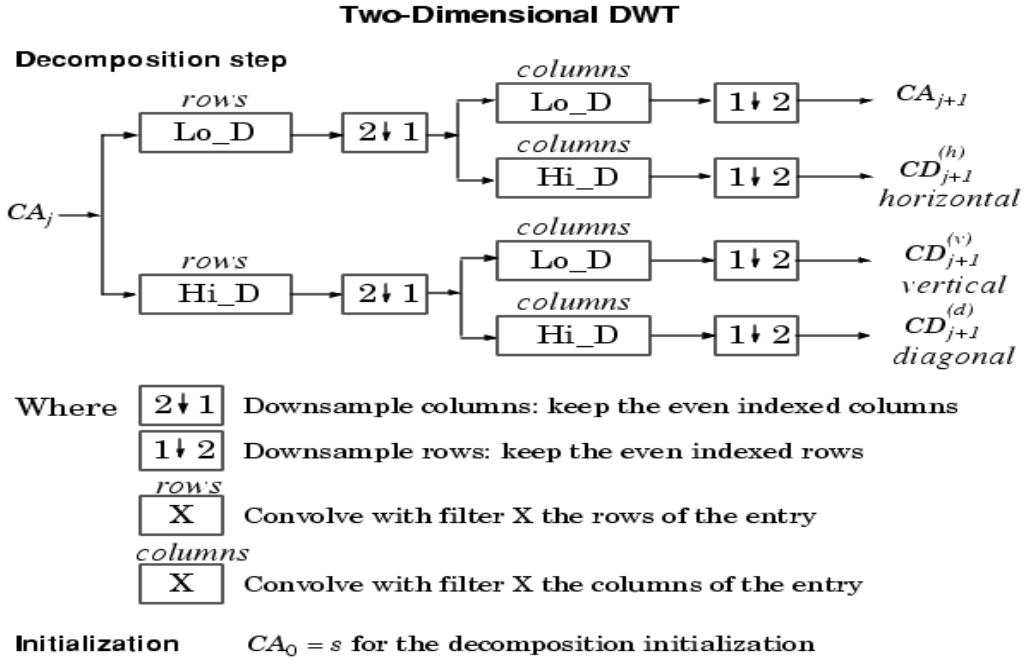


Figure 2. Basic decomposition steps for images

(ii) Reconstruction

After manipulating the Wavelet Transform we thus get the output image by reconstructing which is the inverse wavelet transform. The basic steps of reconstruction are shown in the [Fig 5]

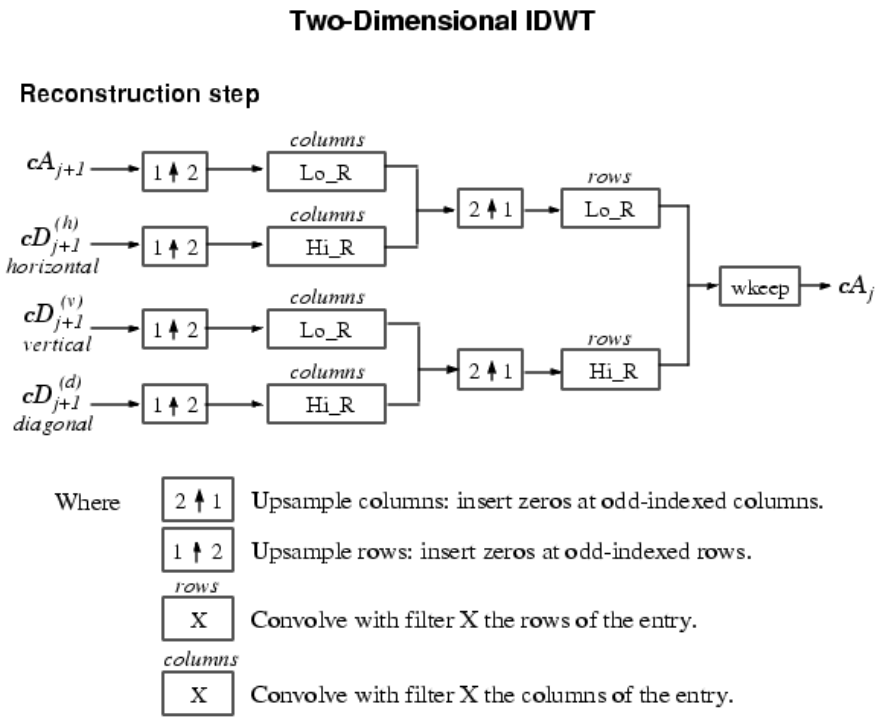


Figure 3. Basic steps of Reconstruction

We are using 2 wavelet fusion techniques which are :

1) Wavelet Maximum Fusion

Once the images have been separated into their wavelet coefficients the 'Maximum Fusion Algorithm' is applied to each of the coefficients. After this the final output image is reconstructed from the combined coefficients.

2) Wavelet Consistency Fusion

Once the images have been separated into their wavelet coefficients the consistency for the 1st coefficient is checked and accordingly we take the pixel value of the output for 1st coefficient. For the 2nd, 3rd and 4th coefficient again 'Maximum Fusion Algorithm' is applied. After this the final output image is reconstructed from the combined coefficients.

4.4 Performance Evaluation

To evaluate the video quality, we have adopted two quality indexes: Xydeas-Petrovic and Wang-Bovik indexes. These indexes use local measures to estimate the level of salient information transferred from the input images into the output image. The first index, called Xydeas-Petrovic index hereinafter, is based on the observation that the human visual system is particularly sensitive to the edges in the image. Therefore, the performance of the fusion method is evaluated as the quantity of information associated to the edges which is transferred from the input images to the fused one. First, the edge information is extracted from the input images; then, the edge strength and orientation are calculated. These features are subsequently used for the index evaluation. The second index, called Wang-Bovik index hereinafter, is based on the universal quality index. The Wang-Bovik index measures the similarity of two images by using three parameters: luminance, contrast and structure. To adapt such index to a fusion problem, the fused image is compared to each source image and a local weighted average of the Wang-Bovik index is evaluated. In this work, the local weights have been chosen as the gradient of the source images to make edge information more relevant.

4.4.1 Xydeas-Petrovic Index

A measure for objectively assessing the pixel level fusion performance is the Xydeas-Petrovic approach [1]. The proposed metric reflects the quality of visual information obtained from the fusion of input images and can be used to compare the performance of different image fusion algorithms. Experimental results clearly indicate that this metric is perceptually meaningful. Given the input and single fused output images, we address the problem of measuring fusion performance objectively. A performance measurement framework is defined which quantifies the fusion process and is subsequently used to compare the performance of different pixel level fusion systems. Furthermore, experimental results of this metric are shown

to be in agreement with preference scores obtained from informal subjective tests. This clearly indicates that the proposed fusion measure is perceptually meaningful.

The goal in pixel level image fusion is to combine and preserve in a single output image all the important visual information that is present in a number of input images. Thus an objective fusion measure should

(i) extract all the perceptually important information that exists in the input images.

(ii) measure the ability of the fusion process to transfer as accurately as possible this information into the output image.

Consider two input images A and B, and a resulting fused image F. Note that the following methodology can be easily applied to more than two input images. A Sobel edge operator is applied to yield the edge strength g(n,m) and orientation a(n,m) information for each pixel p(n,m), $1 \leq n \leq N$ and $1 \leq m \leq M$. Thus for an input image A:

$$g_A(n,m) = \sqrt{s_A^x(n,m)^2 + s_A^y(n,m)^2}$$

$$\alpha_A(n.m) = tan^{-1}\left(\frac{s_A^y(n,m)}{s_A^x(n,m)}\right)$$

Where $s_A^x(n,m)$ and $s_A^y(n,m)$ are the output of the horizontal and vertical Sobel templates centered on pixel $p_A(n,m)$ and convolved with the corresponding pixels of image A. The relative strength and orientation values of $G^{AF}(n,m)$ and $A^{AF}(n,m)$ of an input image A with respect to F are formed.

These are used to derive the edge strength and orientation preservation values $Q_g^{AF}(n,m)$ and $Q_\alpha^{AF}(n,m)$ model the perceptual loss of information in F, in terms of how well the strength and orientation values of a pixel p(n,m) in A are represented in the fused image. The constants Γ_g, κ_g, σ_g and Γ_α, κ_α, σ_α determine the exact shape of the sigmoid functions used to form the edge strength and orientation preservation values.

4.4.2 Wang-Bovik Index

We propose a new universal objective image quality index, which is easy to calculate and applicable to various image processing applications [6]. Instead of using traditional error summation methods, the proposed index is designed by modeling any image distortion as a combination of three factors: loss of correlation, luminance distortion, and contrast distortion. Mathematically defined measures are easy to calculate and usually have low computational complexity. They are independent of viewing conditions and individual observers. Although it is believed that the viewing conditions play important roles in human perception of image quality, they are, in most cases, not fixed and specific data is generally unavailable to the image analysis system. In addition, it becomes the user's responsibilities to measure the viewing conditions and to calculate and input the condition parameters to the measurement systems. By contrast, a viewing condition-independent measure delivers a single quality value that gives a general idea of how good the image is. Universal means that the quality measurement approach does not depend on the images being tested, the viewing conditions or the individual observers. More importantly, it must be applicable to various image processing applications and provide meaningful comparison across different types of image distortions.

The Wang-Bovik index measures the similarity of two images by using three parameters: luminance, contrast and structure.

$$Q = \frac{\sigma_{xy}}{\sigma_x \sigma_y} \frac{2\bar{x}\,\bar{y}}{(\bar{x}^2 + \bar{y}^2)} \frac{2\sigma_x \sigma_y}{\left(\sigma_x^2 + \sigma_y^2\right)}$$

The first component is the correlation coefficient between x and y, which measures the degree of linear correlation between x and y, and its dynamic range is [1,-1]. The second component measures how close the mean luminance is between x and y, and its dynamic range is [0,1]. The third component measures how similar the contrasts of the images are, σ_x and σ_y can be viewed as estimate of the contrast of x and y, its dynamic range is [0,1].

4.5 Algorithms

Step 1. Acquiring Frames From The Video

1.1 Break the Video in to frames using the "aviread" function.

1.2. Passing each frame through a median filter having a (3 X 3) mask to remove the noise.

Step 2: Extracting Y component From The Visual Frames

2.1. Use the rgb2ntsc function to get the YIQ component directly.

Step 3: Background Estimation

3.1 We perform background estimation for each second, 25 frames.

3.2 We maintain a table in the form of an array with rows columns and the pixel values.

3.3 For each frame store the pixel value in a temporary variable, if any pixel value is 0 make it 1.

3.4 We increment the value in the table for the pixel value we read.

3.5 Now we take the maximum occurring value for each pixel from the table which will be a new image which will be the estimated background.

Step 4. Moving Object Extraction (MOE)

4.1 MOE is a binary mask which is obtained from the IR frames and the IR background estimated image.

4.2 We take IR frames and subtract it from the extracted background image and if the value is greater than threshold 40 then we take the pixel value as 255 else it is 0.Thus we get the object in white and rest in black.

$$MOE = \begin{cases} 255; & if\ (IR - BG) > |40| \\ 0 & otherewise \end{cases}$$

Step 5. Object Extraction

5.1 For object extraction of both infrared and visual frames we use the MOE binary mask.

5.2 In the MOE if the pixel value is 255 we map the corresponding pixel values from the IR frame into a new image and the remaining pixel values of the the new image are 0. We perform the same for visual

$$IR\ Object = \begin{cases} IR\ Pixel\ Value; & if\ MOE = 255 \\ 0 & Otherwise \end{cases}$$

$$Visual\ Object\ = \begin{cases} Visual\ Pixel\ Value; & if\ MOE = 255 \\ 0 & Otherwise \end{cases}$$

Step 6. Fusion

(i) Block Fusion

1. Take 2 input images.
2. For every pixel p(i, j) of each image its neighboring pixels are added and a block average is calculated.
3. Compare the value with all the images.
4. Then the pixel with the maximum block average is copied to the output image.

5. This is repeated for all pixel values.

(ii) Simple Average Fusion.

1. Take 2 input images
2. The value of the pixel p(i, j) of each image is taken and added.
3. This sum is then divided by the number of images (2) to obtain the average.
4. This average value is assigned to the respective pixel of the output image.
5. This is repeated for all pixel values.

(iii) Maximum Method of Fusion.

1. Take 2 input images.
2. The value of the pixel p(i , j) of each image is taken and compared to each other.
3. The greatest pixel value is assigned to the corresponding pixel of the output image.
4. This is repeated for all pixel values.

(iv) Wavelet Maximum

1. Take the two input images to be fused.
2. Decompose the images in wavelet coefficients.
3. The value of the pixel p(i, j) of 1^{st} wavelet coefficient of each image is taken and compared to each other.
4. The greatest pixel value is assigned to the corresponding pixel of the 1^{st} wavelet coefficient of output image.
5. This is repeated for 4 coefficients.
6. The 4 coefficients of the output image is then reconstructed into the output image.

(v) Wavelet Consistency

1. Take the two input images to be fused.
2. Decompose the images in wavelet coefficients.

3. The absolute value of the pixel P (i, j) of 1st wavelet coefficient of each image is taken and compared to each other, if 1st is greater you store variable(x) value as 1 else 0 for corresponding pixel.
4. We check the consistency of the pixel value by adding the neighboring pixel values and the variable value(x) corresponding to the pixel.
5. If the sum is less than 2 the output variable(y) value is assigned 0, if it is greater than 7 it is assigned 1, else the value is assigned 2 corresponding to each pixel.
6. If the output variable value(y) is 0 the corresponding pixel value of 1st wavelet coefficient of output is taken as the 1st wavelet coefficient of 1st image, else if it is 1 the output value is taken as 1st wavelet coefficient of 2nd image, else average of 2 images.
7. The value of pixel P(i,j) of 2nd wavelet coefficient of each image is taken and compared to each other.
8. The greatest pixel value is assigned to the corresponding pixel of the 2nd wavelet coefficient of output image.
9. This is repeated for 3rd and 4th coefficients.
10. The 4 coefficients of the output image is then reconstructed into the output image.

Step 7. Converting NTSC to RGB

7.1. Use the ntsc2rgb function to get the RGB component directly.

Eg. YIQ = rgb2ntsc(RGB)

Step 8. Composition

8.1 In this step we map the values of the fused object image on the fused extracted background.

$$Final\ output = \begin{cases} fused\ object; & if\ fused\ object! = 255 \\ fused\ background & Otherwise \end{cases}$$

4.5.1 Algorithms for Performance Evaluation

(i) Xydeas Petrovic

1. We have the input images A and B (suppose) and after running it we will have the fused image. Run the Sobel mask for x and y gradients on the input image and find the edge strength and orientation using the formula:

$$g_A(n,m) = \sqrt{s_A^x(n,m)^2 + s_A^y(n,m)^2}$$

$$\alpha_A(n.m) = tan^{-1}\left(\frac{s_A^y(n,m)}{s_A^x(n,m)}\right)$$

2. Find the relative strength and orientation values:

$$G^{AF}(n,m)=\begin{cases}\frac{g_F(n,m)}{g_A(n,m)} & \text{if } g_A(n,m)>g_F(n,m)\\ \frac{g_A(n,m)}{g_F(n,m)} & otherwise\end{cases}$$

$$A^{AF}(n,m)=1-\frac{|\alpha_A(n.m)-\alpha_F(n.m)|}{\pi/2}$$

3. The edge strength and orientation values are calculated as:

$$Q^{AF}{}_g(n,m) = \frac{\Gamma_g}{1+\ e^{k_g(G_{n,m}^{AF}-\sigma_g)}}$$

$$Q^{AF}{}_\alpha(n,m) = \frac{\Gamma_\alpha}{1+\ e^{k_\alpha(A_{n,m}^{AF}-\sigma_\alpha)}}$$

In this step we use many constants like Γ_g, κ_g, σ_g and Γ_α, κ_α, σ_α and their values are;

L=1, Γ_g=0.9994, κ_g=-15, σ_g=0.5, Γ_α =0.9879, κ_α =-22, σ_α =0.8

4. Edge information preservation can be obtained by using the above 2 values as:

$$Q^{AF}(n,m) = Q^{AF}{}_g(n,m)Q^{AF}{}_\alpha(n,m)$$

After we get $Q^{AF}(n,m)$ and $Q^{BF}(n,m)$ from the input images A and B a weighted performance metric $Q_p^{AB/F}$ is calculated as follows

$$Q_P^{AB/F} = \frac{\sum_{n=1}^{N}\sum_{m=1}^{M} Q^{AF}(n,m)w^A(n,m) + Q^{BF}(n,m)w^B(n,m)}{\sum_{i=1}^{N}\sum_{j=1}^{M}(w^A(i,j) + w^B(i,j))}$$

$Q^{AF}(n,m)$ *and* $Q^{BF}(n,m)$ are weighted by $w^A(n,m) and\ w^B(n,m)$ where $w^A(n,m) = [g_A(n,m)]^L$ and L is a constant.

Thus we plot Xydeas-Petrovic quality index for 75 frames[fig14].

The mean formulated for the video is: 0.4522

(ii) Wang Bovik

1. Let $x = \{x_i, i = 1,2,\dots,N\}$ be the original image signal and $y = \{y_i, i = 1,2,\dots,N\}$ be the fused image signal. Then we calculate the following values as follows:

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \qquad \bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$$

$$\sigma_x{}^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2 \qquad \sigma_y{}^2 = \frac{1}{N-1}\sum_{i=1}^{N}(y_i - \bar{y})^2$$

$$\sigma_{xy} = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})$$

2. Further we calculate quality index as:

$$Q = \frac{4\sigma_{xy}\bar{x}\,\bar{y}}{(\sigma_x^2 + \sigma_y^2)[(\bar{x})^2 + (\bar{y})^2]}$$

Thus we plot the graph for the wang bovik quality index for IR frames with fused frames[fig17] and bovik quality index for Visual frames with fused frames [fig18] .

5. Results and Conclusion

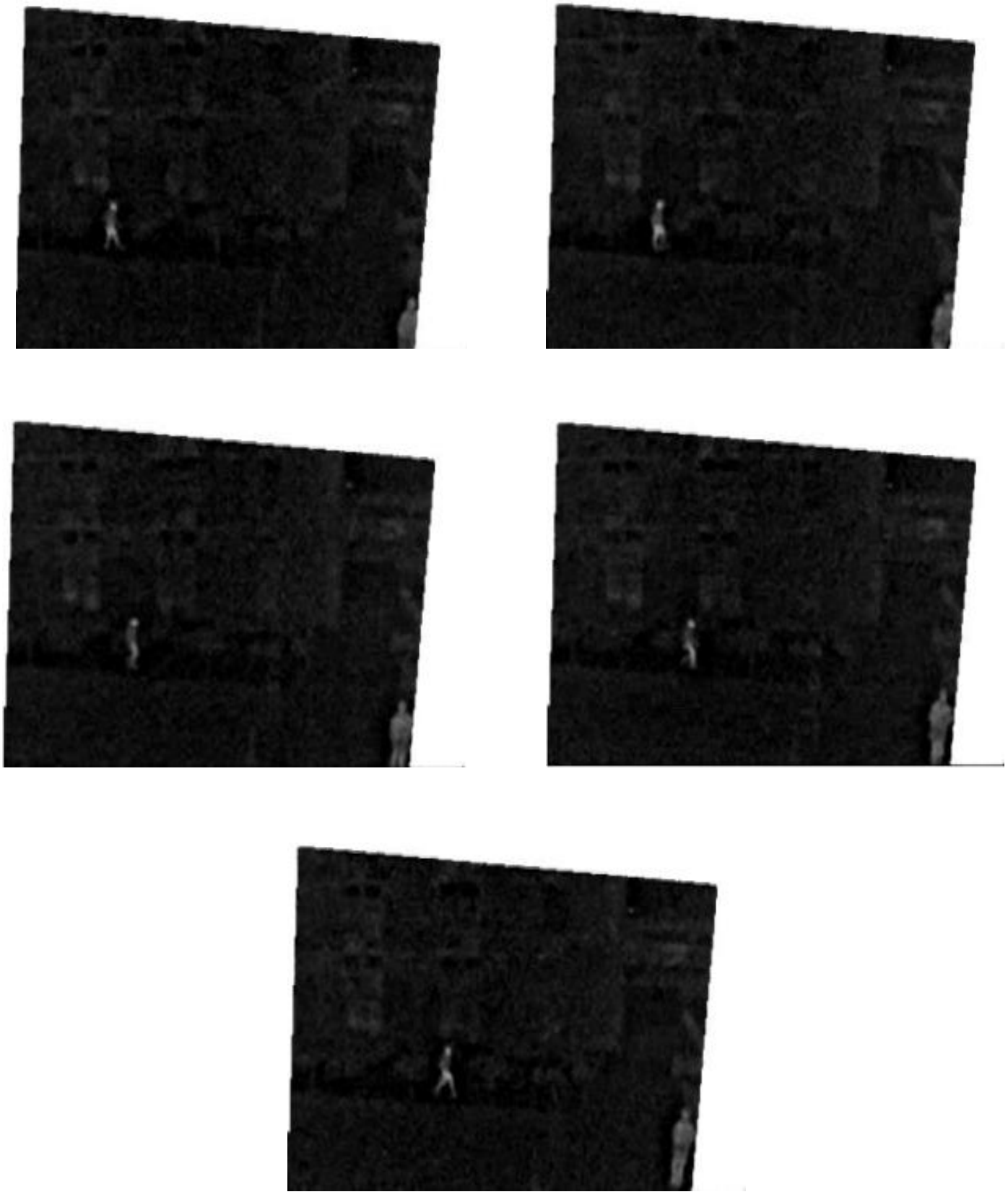


Figure 4. IR Camera Frames (10, 20, 30, 40, 50)





Figure 5. Visible Camera Frames (10, 20, 30, 40, 50)

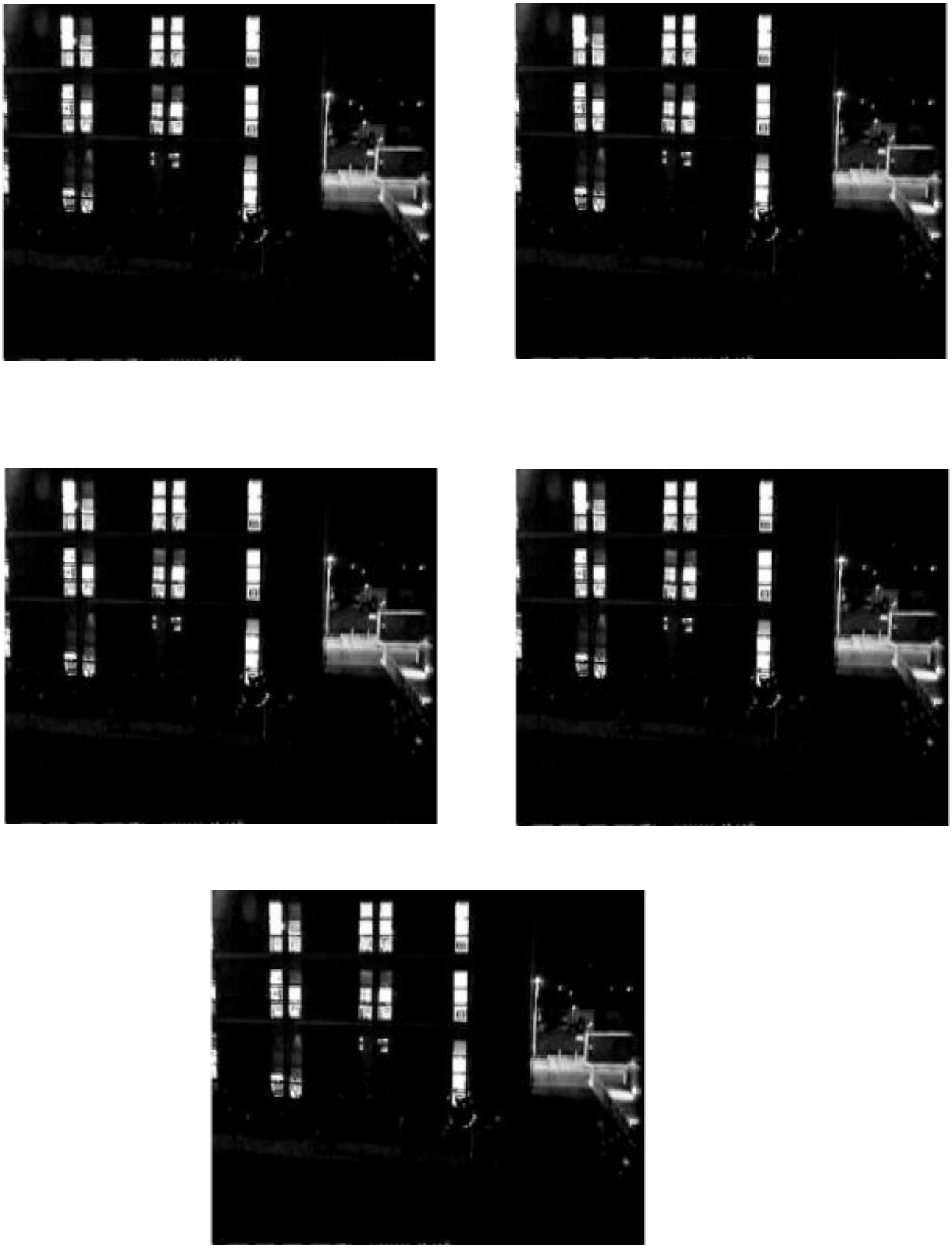


Figure 6. Y component of visual camera frames (10, 20, 30, 40, 50)

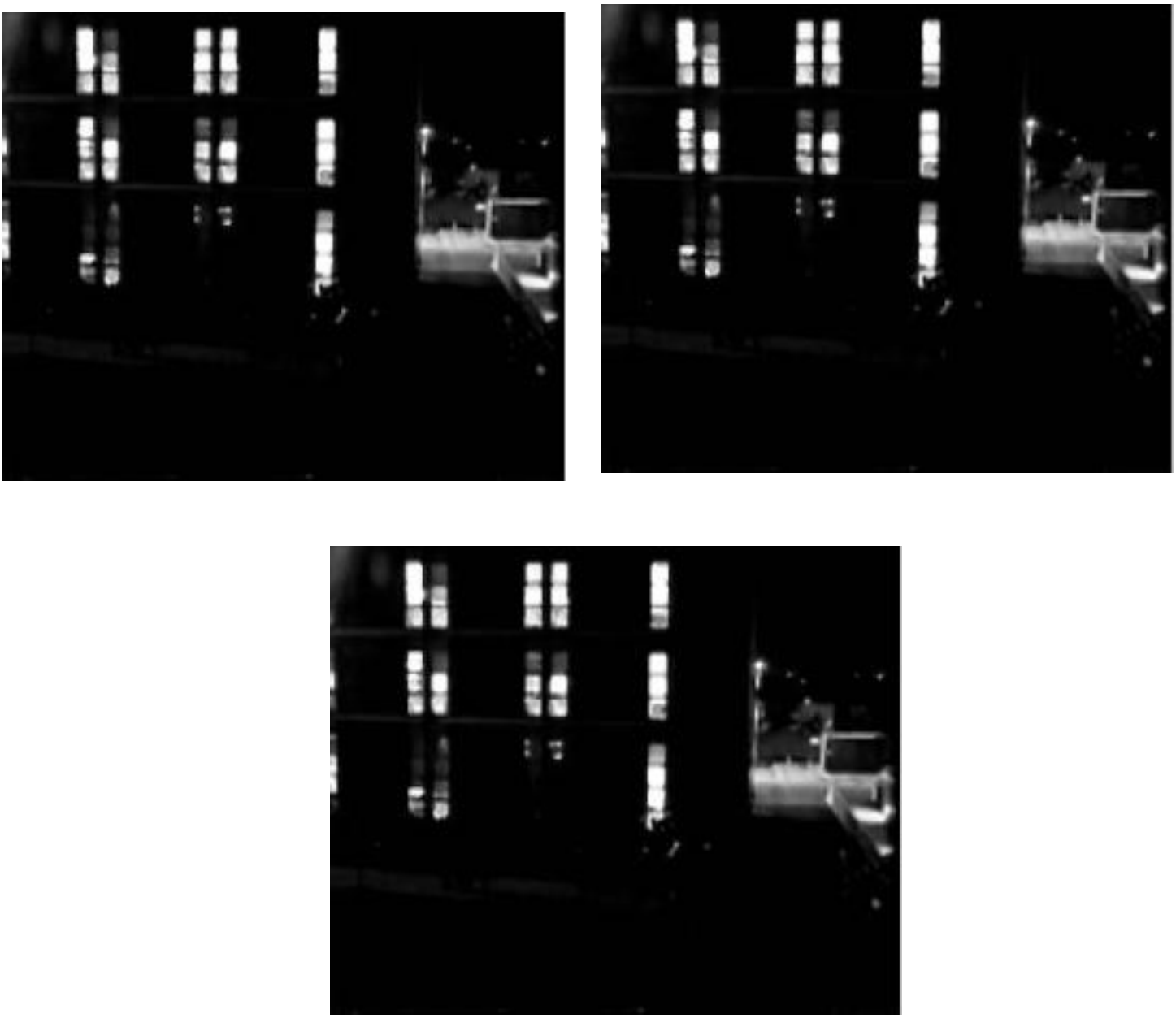


Figure 7. Background Estimation for Visual Camera Frames

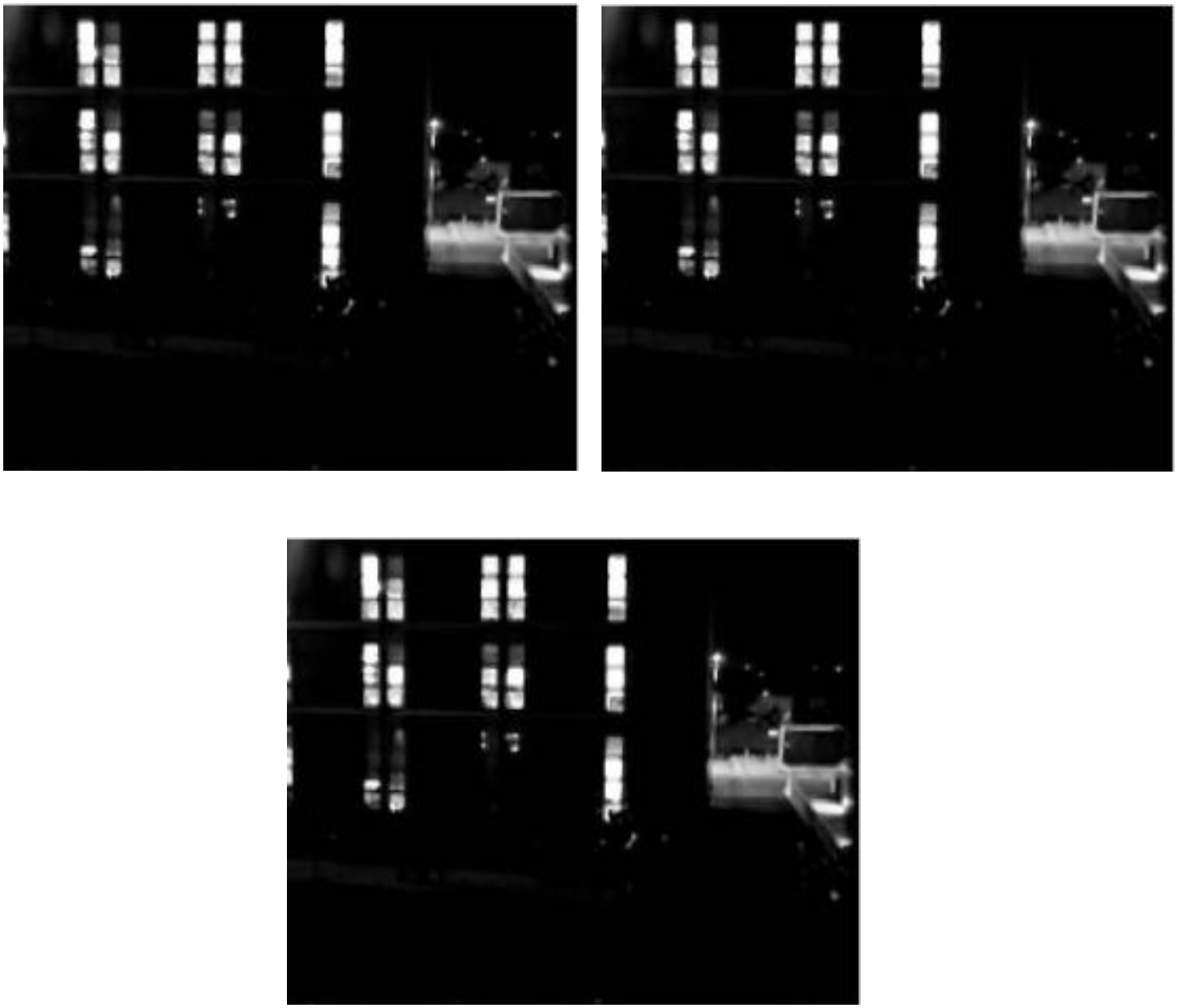


Figure 8. Fused Background

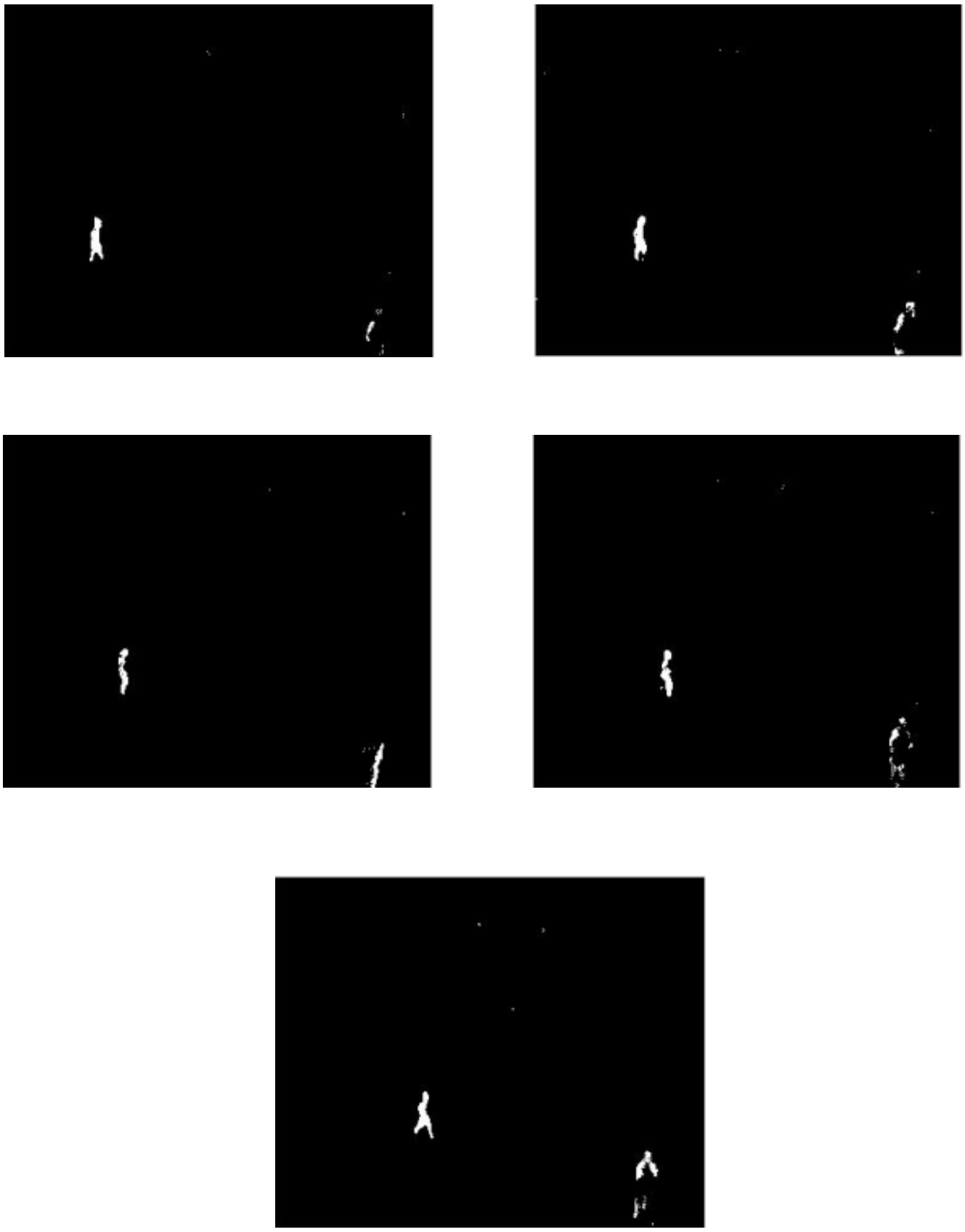


Figure 9. Object Extraction: Binary Mask (10, 20, 30, 40, 50)

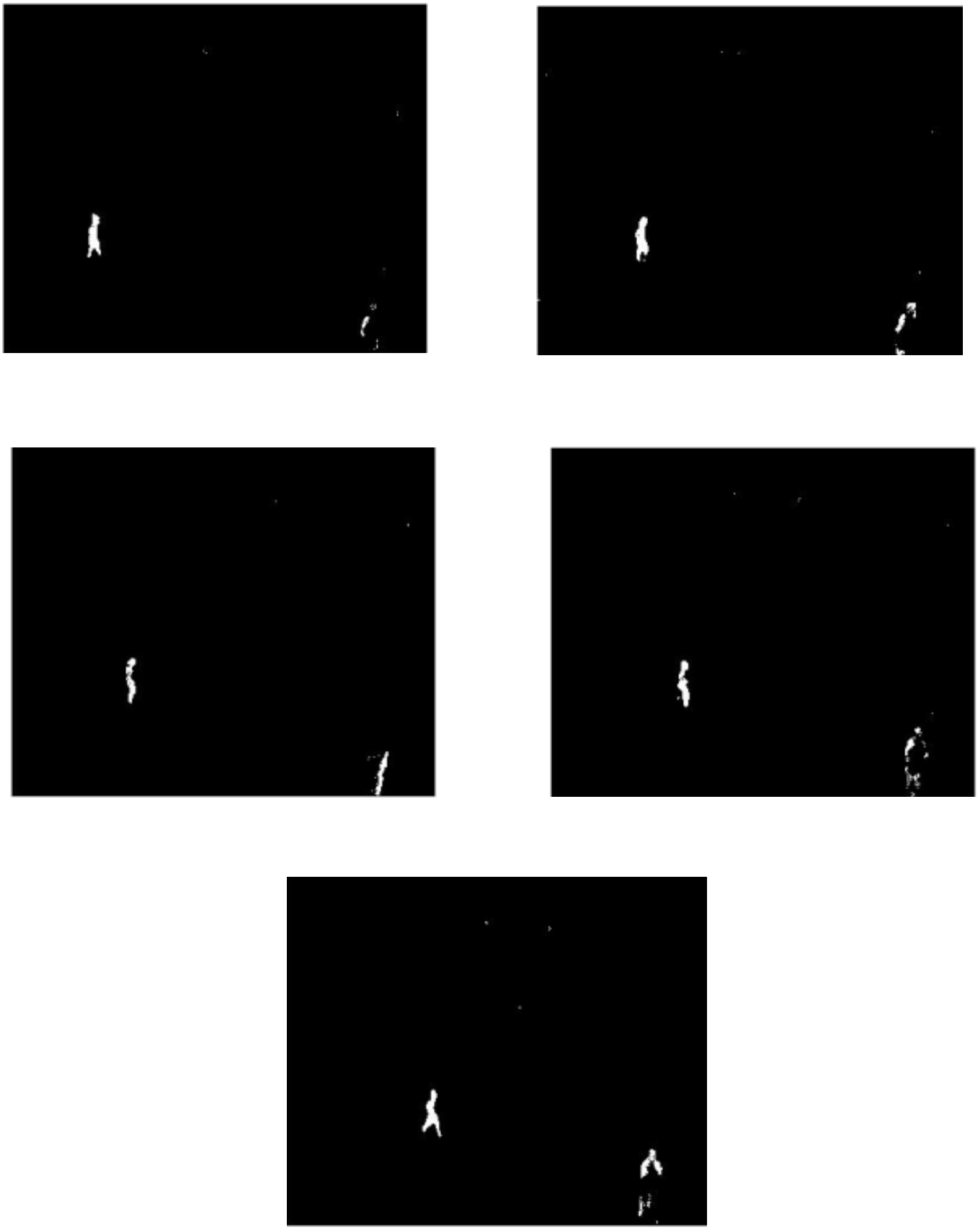


Figure 10. Object Extraction: IR object (10, 20, 30, 40, 50)

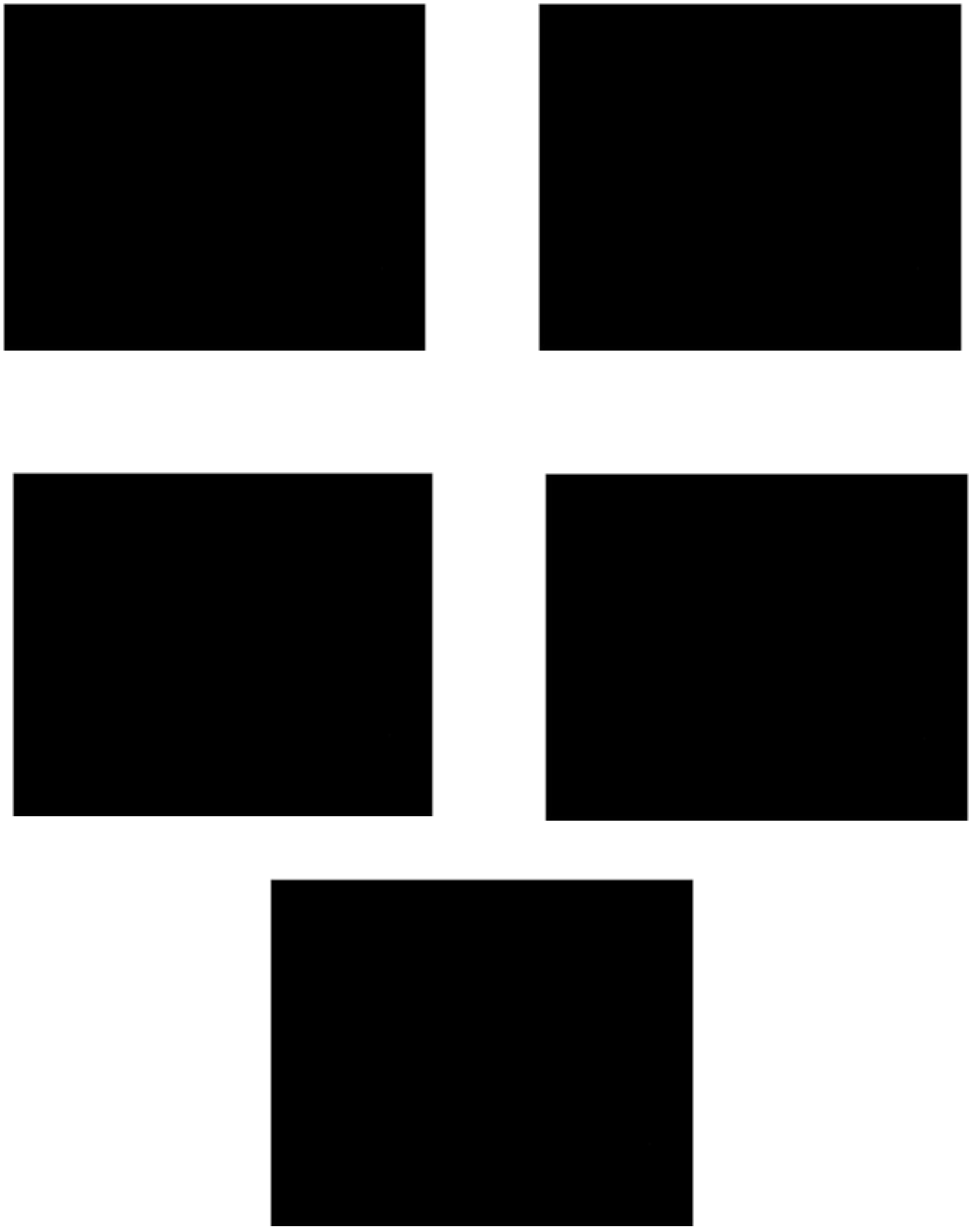


Figure 11. Object Extraction: Y Component of Visual Object (10, 20, 30, 40, 50)

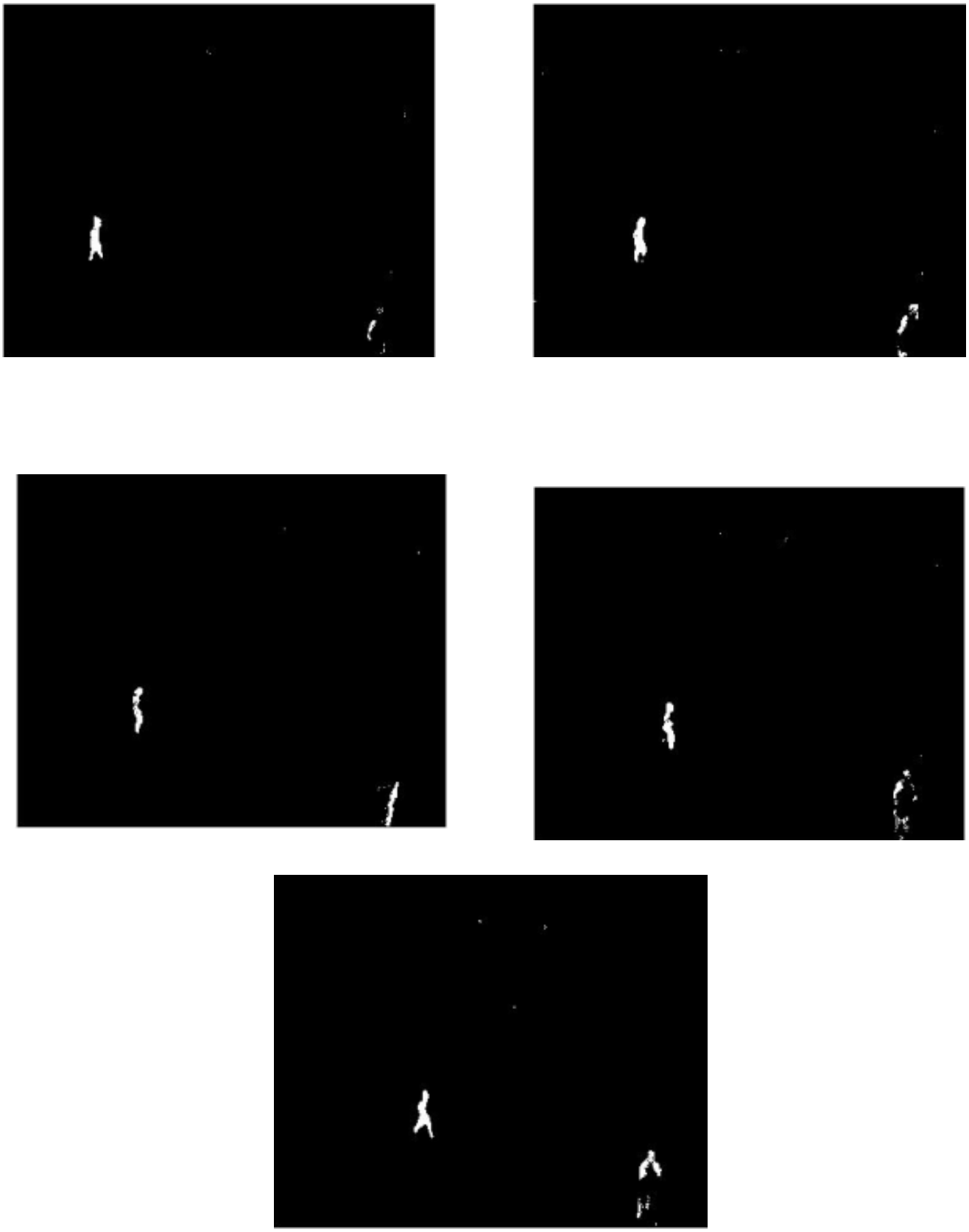
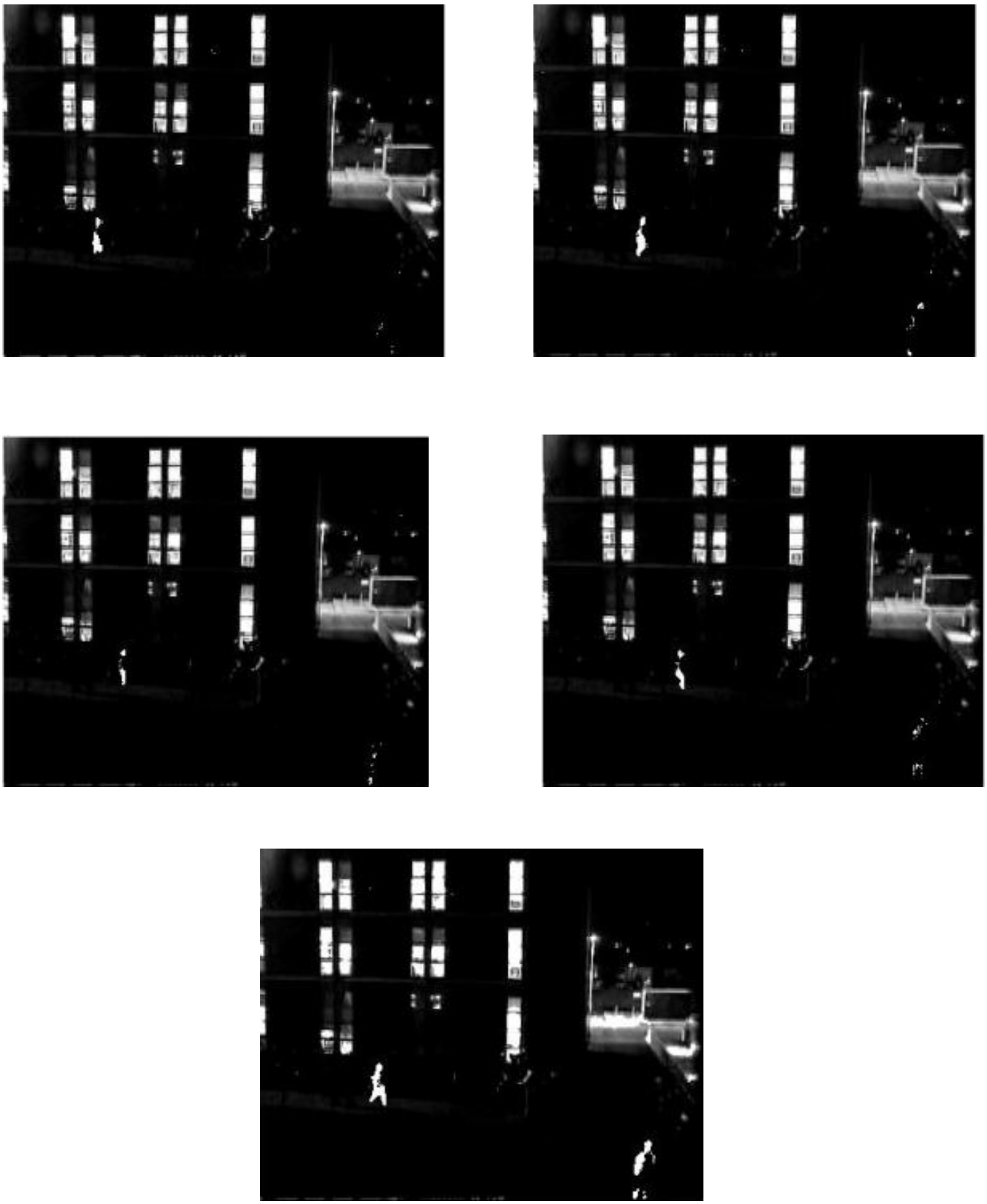


Figure 12. Object Extraction: Fused Objects (10, 20, 30, 40, 50)



Figured 13. Fused Y-Background and Objects (10, 20, 30, 40, 50)

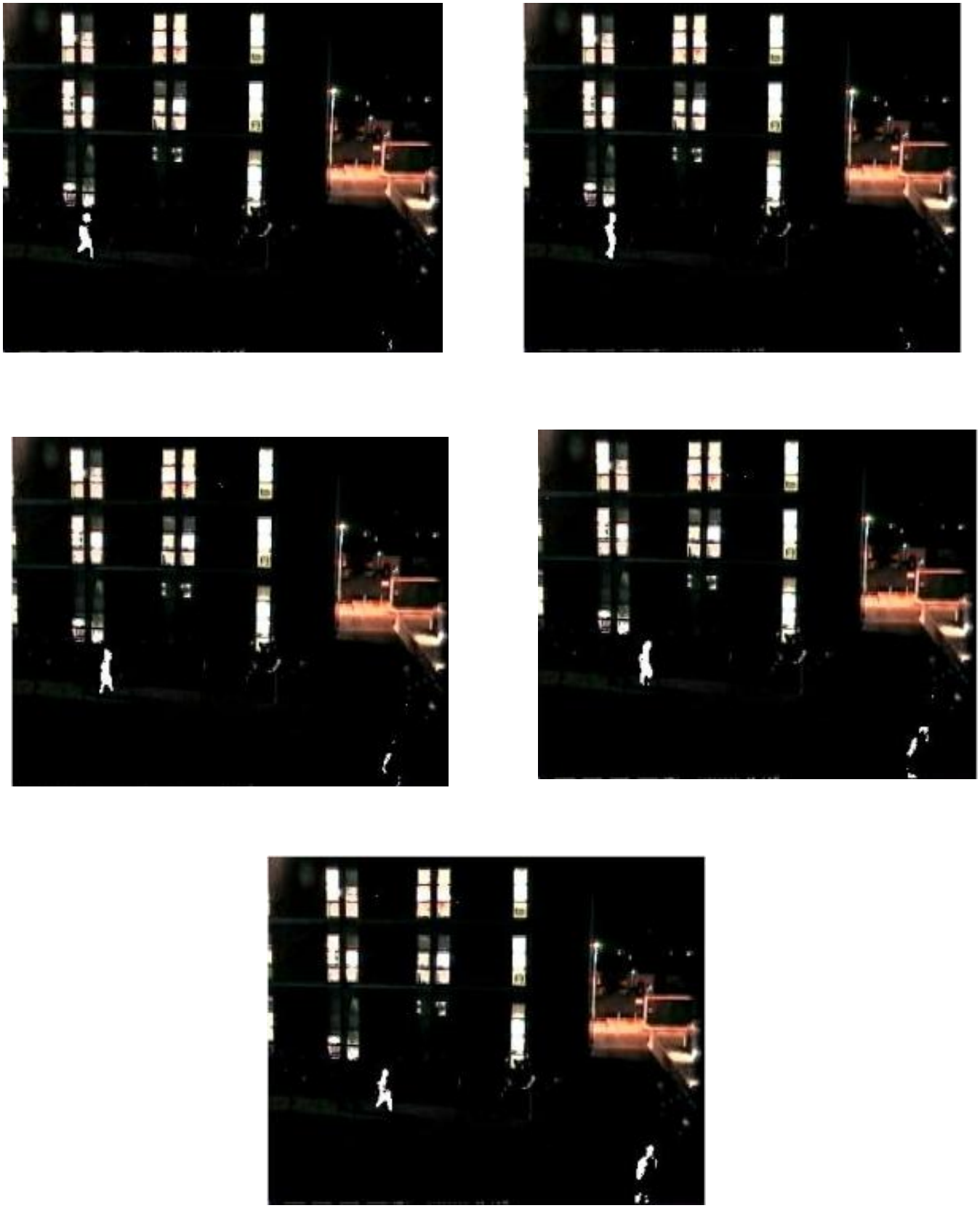


Figure 14. Final Fused YIQ-Background and Object (10, 20, 30, 40, 50)

Performance Evaluation Results

Wang Bovik Indices

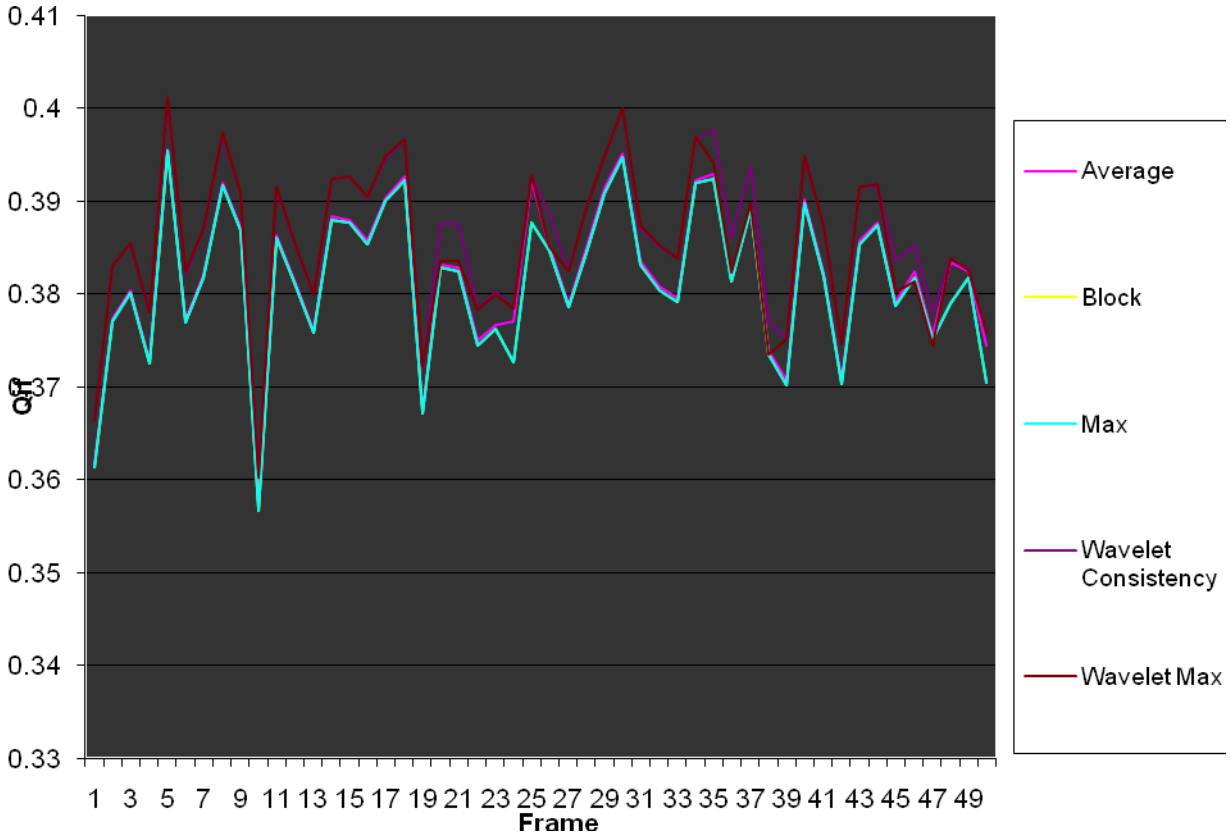


Figure 15. Quality Index for Infrared frames

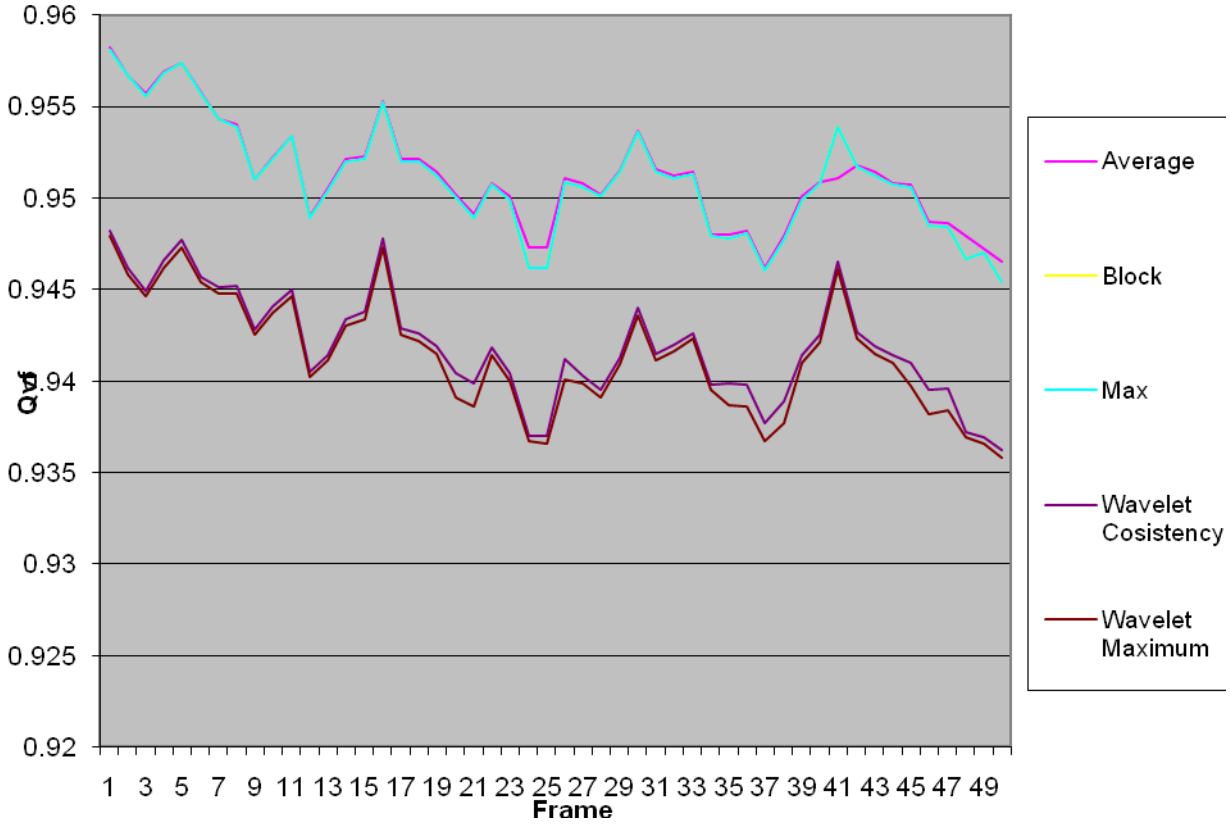


Figure 16. Quality Index for Visual frames

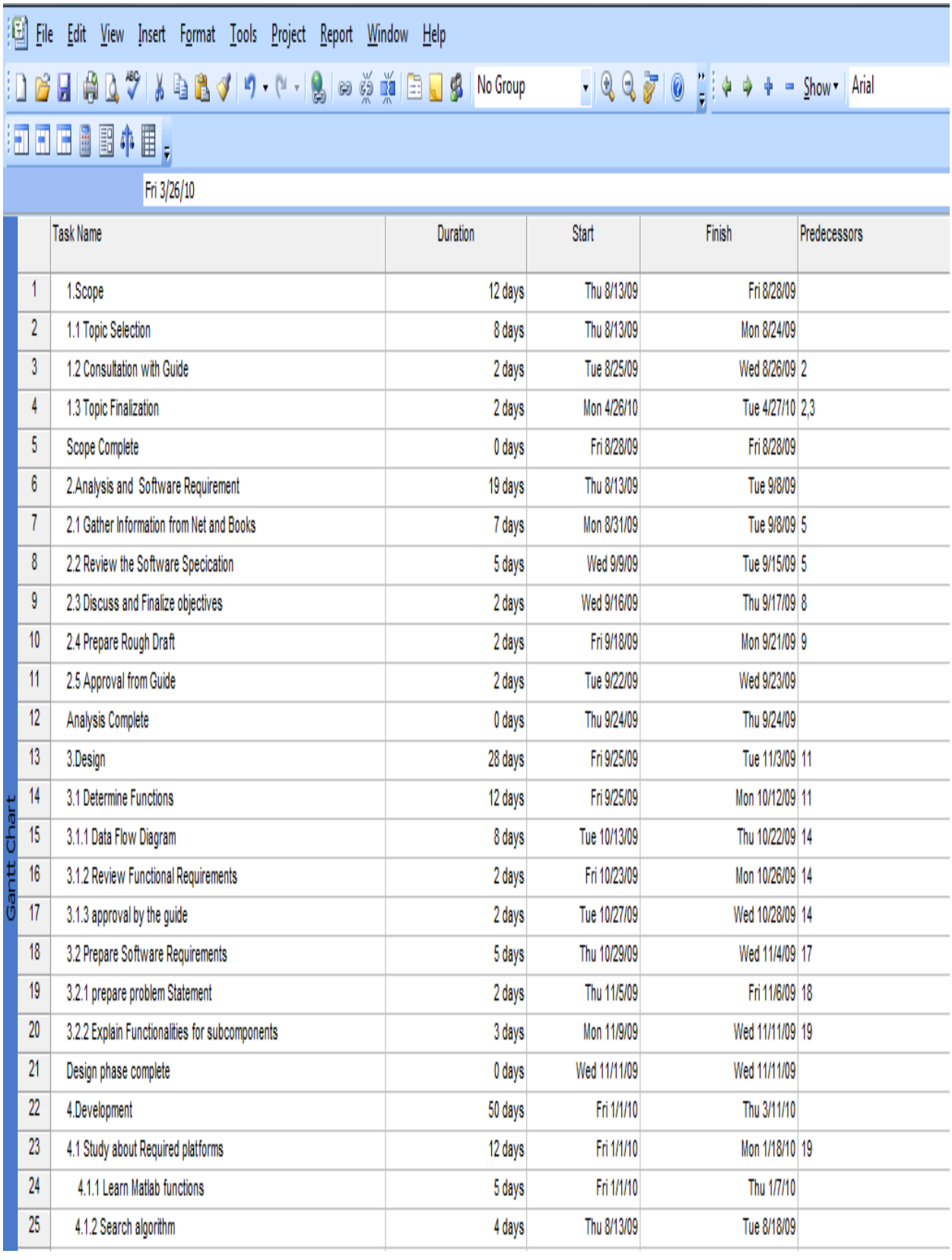
Table 1. Performance Results for The Different Fusion Techniques.

	Index			
		Wang-Bovik Q_{if}	Wang-Bovik Q_{vf}	Xydeas-Petrovic Q_{vif}
Type of Fusion	Simple Average	0.382024	0.951214	0.38916
	Simple Block	0.381318	0.951068	0.53236
	Simple Maximum	0.381318	0.951068	0.53218
	Wavelet Maximum	03.85978	0.941952	0.36498
	Wavelet Consistency	0.38527	0.94142	0.39374

Wang Bovik method of testing primarily concentrates on correlation of the visual and infrared video to that of fused video. The amount of information used in the final fused video is primarily of the visual video as it contains information of the background whereas infrared is mainly used for obtaining the object hence correlation is not high. The results of the Wang Bovik method of testing indicate that Average Method of fusion gives the best result for quality index of visual video which is 0.9512. Block and Maximum come very close with 0.9511. The method of Wavelet Maximum gives us the lowest Quality index which is 0.9414. As for the infrared video the Wavelet Maximum gives us the best result with 0.3896 and Block Fusion and Maximum give us the least result with index as 0.3814. Xydeas-Petrovic is an index that measures the quality of the object introduced in the final result. As per Xydeas-Petrovic the best method is Simple block with value as 0.5323 and Simple Maximum is very close with 0.5321.

Using both the results we can conclude that Simple Block and Simple Maximum methods of fusion provide the best results.

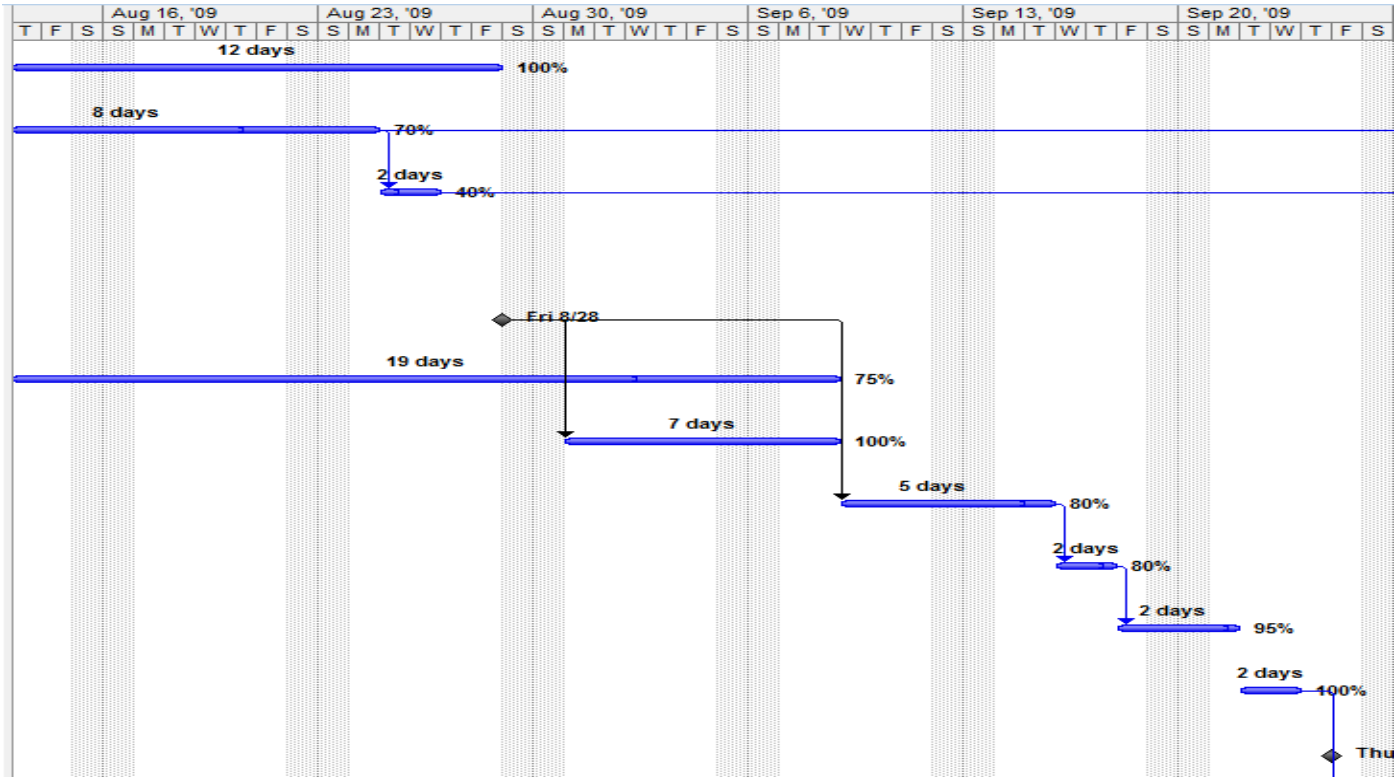
6. Project Time Lines

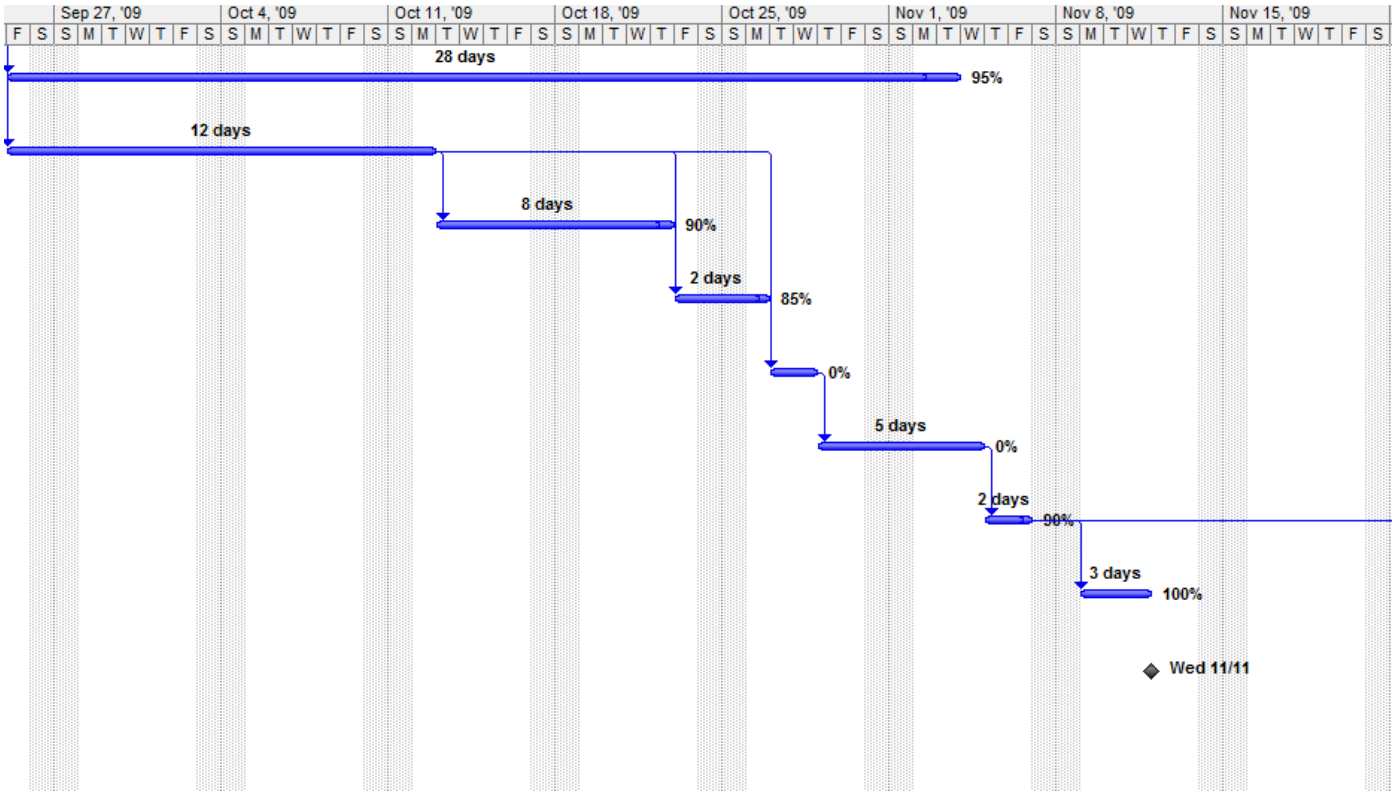


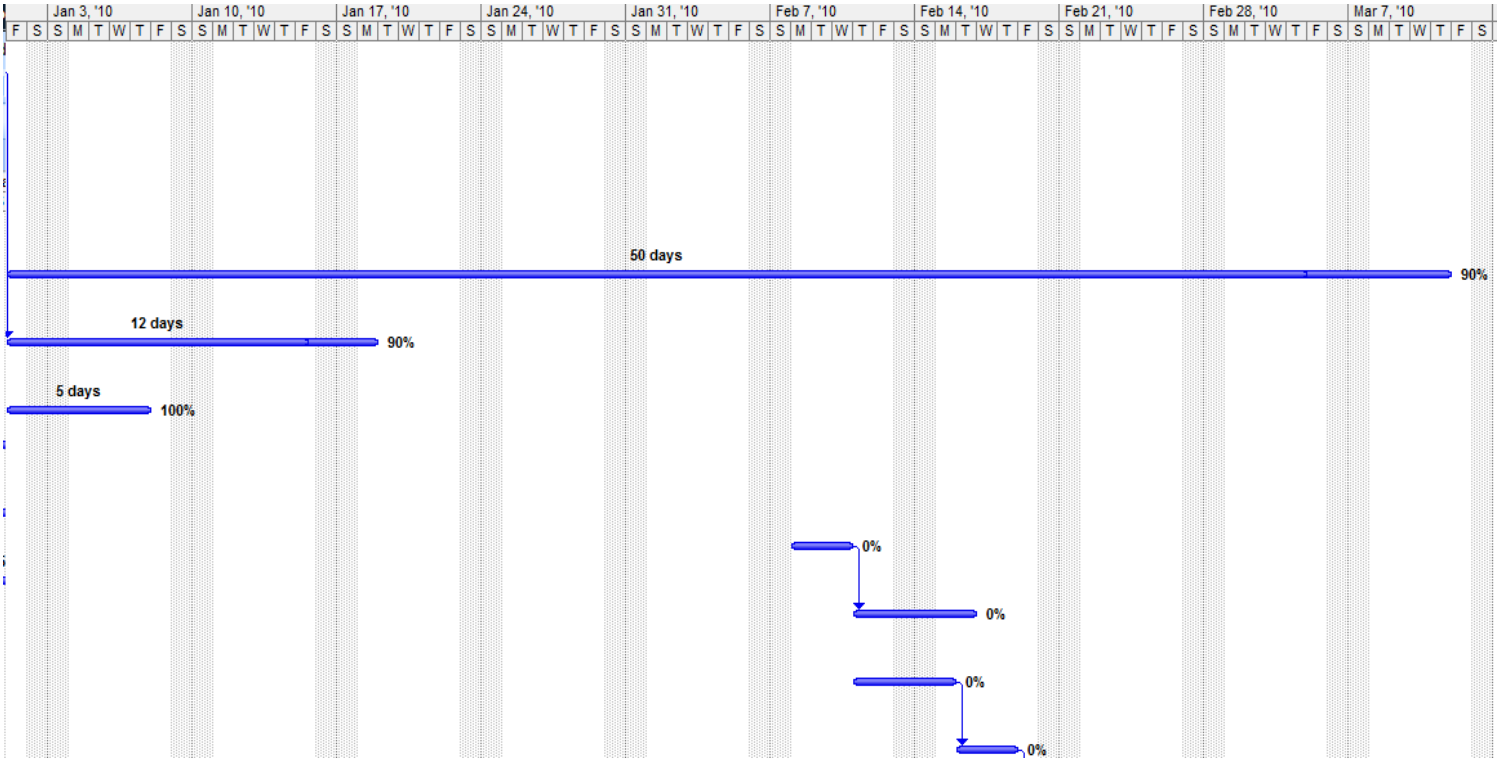
	Task Name	Duration	Start	Finish	Predecessors
1	1.Scope	12 days	Thu 8/13/09	Fri 8/28/09	
2	1.1 Topic Selection	8 days	Thu 8/13/09	Mon 8/24/09	
3	1.2 Consultation with Guide	2 days	Tue 8/25/09	Wed 8/26/09	2
4	1.3 Topic Finalization	2 days	Mon 4/26/10	Tue 4/27/10	2,3
5	Scope Complete	0 days	Fri 8/28/09	Fri 8/28/09	
6	2.Analysis and Software Requirement	19 days	Thu 8/13/09	Tue 9/8/09	
7	2.1 Gather Information from Net and Books	7 days	Mon 8/31/09	Tue 9/8/09	5
8	2.2 Review the Software Specication	5 days	Wed 9/9/09	Tue 9/15/09	5
9	2.3 Discuss and Finalize objectives	2 days	Wed 9/16/09	Thu 9/17/09	8
10	2.4 Prepare Rough Draft	2 days	Fri 9/18/09	Mon 9/21/09	9
11	2.5 Approval from Guide	2 days	Tue 9/22/09	Wed 9/23/09	
12	Analysis Complete	0 days	Thu 9/24/09	Thu 9/24/09	
13	3.Design	28 days	Fri 9/25/09	Tue 11/3/09	11
14	3.1 Determine Functions	12 days	Fri 9/25/09	Mon 10/12/09	11
15	3.1.1 Data Flow Diagram	8 days	Tue 10/13/09	Thu 10/22/09	14
16	3.1.2 Review Functional Requirements	2 days	Fri 10/23/09	Mon 10/26/09	14
17	3.1.3 approval by the guide	2 days	Tue 10/27/09	Wed 10/28/09	14
18	3.2 Prepare Software Requirements	5 days	Thu 10/29/09	Wed 11/4/09	17
19	3.2.1 prepare problem Statement	2 days	Thu 11/5/09	Fri 11/6/09	18
20	3.2.2 Explain Functionalities for subcomponents	3 days	Mon 11/9/09	Wed 11/11/09	19
21	Design phase complete	0 days	Wed 11/11/09	Wed 11/11/09	
22	4.Development	50 days	Fri 1/1/10	Thu 3/11/10	
23	4.1 Study about Required platforms	12 days	Fri 1/1/10	Mon 1/18/10	19
24	4.1.1 Learn Matlab functions	5 days	Fri 1/1/10	Thu 1/7/10	
25	4.1.2 Search algorithm	4 days	Thu 8/13/09	Tue 8/18/09	

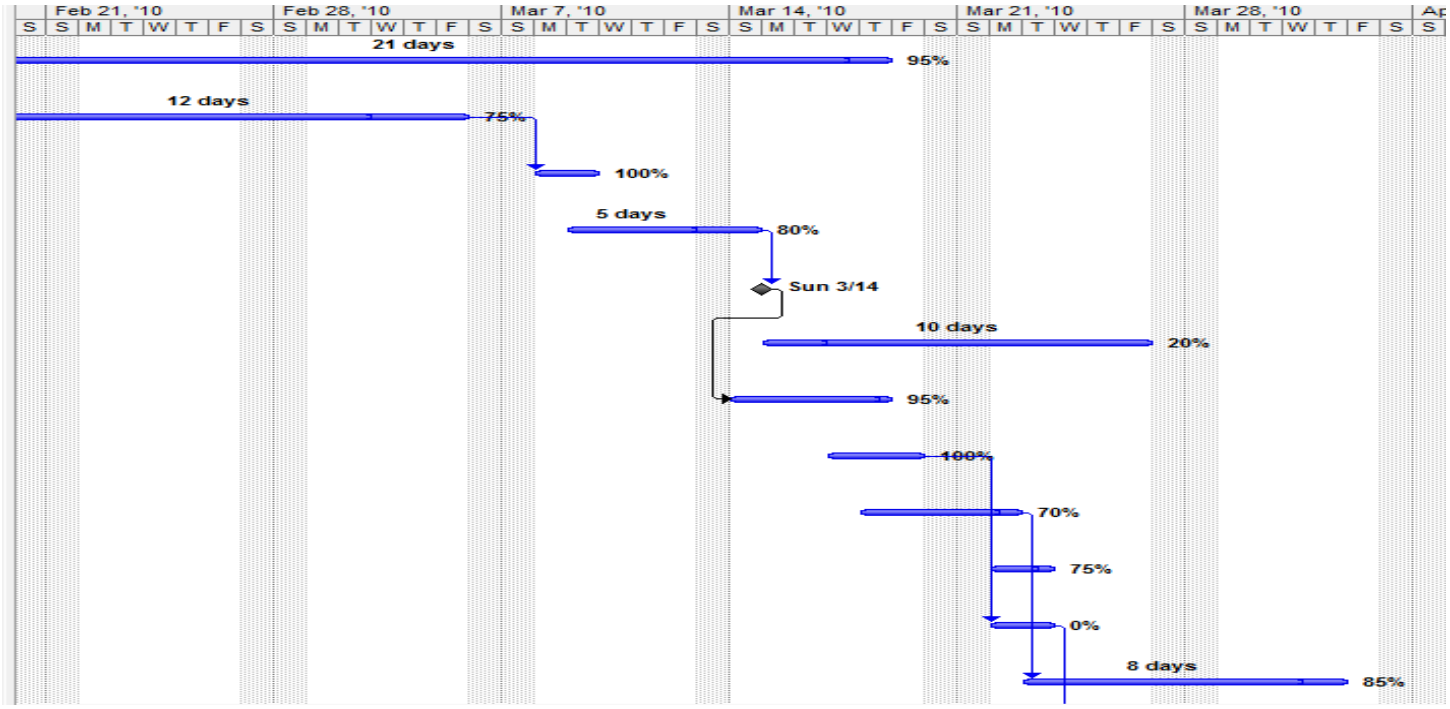
Gantt Chart

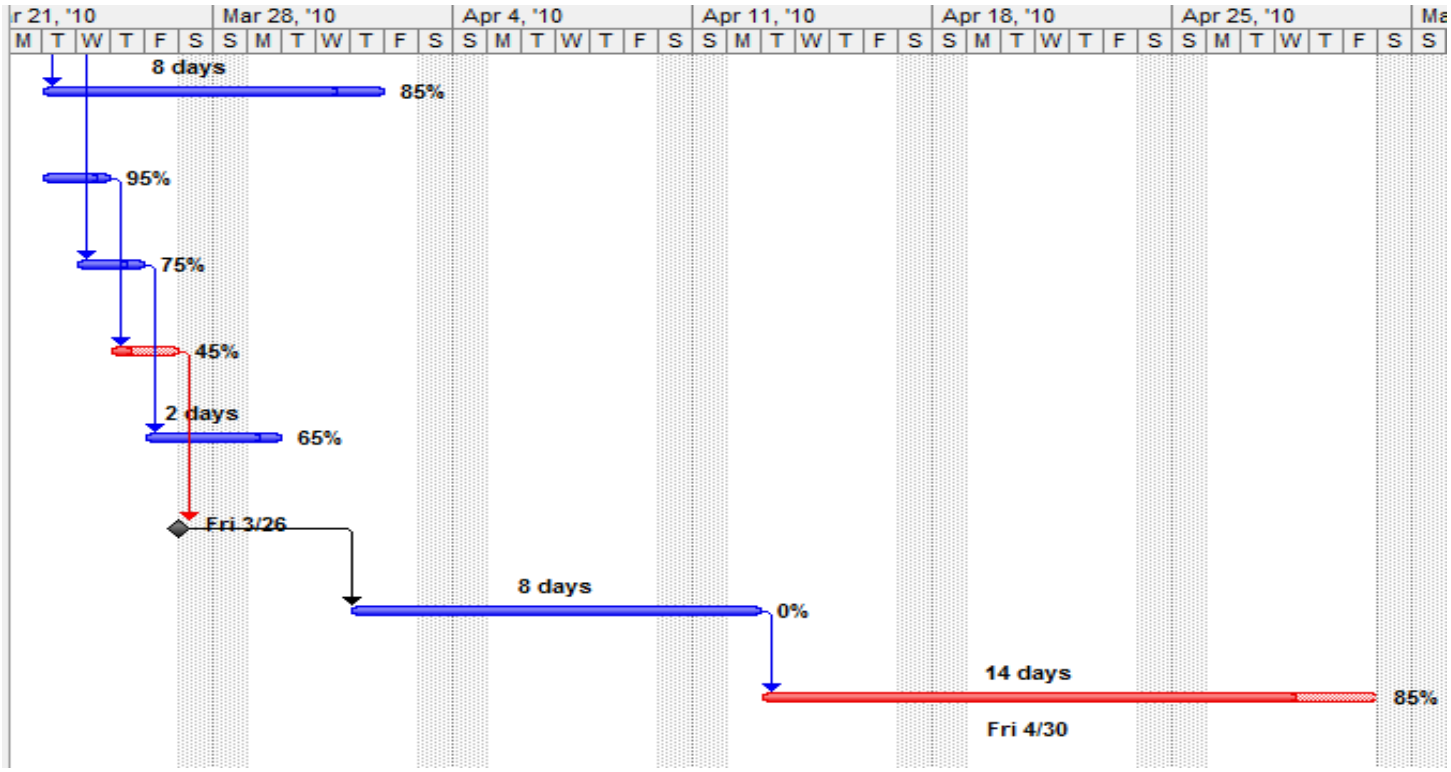
26	4.2 Review and Finalize Algorithms	3 days	Mon 2/8/10	Wed 2/10/10	
27	4.3 Design Layout	4 days	Thu 2/11/10	Tue 2/16/10	26
28	4.3.1 Subcomponents of design module	3 days	Thu 2/11/10	Mon 2/15/10	
29	4.3.2 Review and finalize Design	3 days	Tue 2/16/10	Thu 2/18/10	28
30	4.4 Develop Code	21 days	Fri 2/19/10	Thu 3/18/10	29
31	4.4.1 Implement Functionalities of the sub modules	12 days	Thu 2/18/10	Fri 3/5/10	
32	4.4.2 Connect with backend	2 days	Mon 3/8/10	Tue 3/9/10	31
33	4.4.3 Search algorithm Implementation	5 days	Tue 3/9/10	Sun 3/14/10	
34	Development phase Complete	0 days	Sun 3/14/10	Sun 3/14/10	33
35	5. Testing	10 days	Mon 3/15/10	Fri 3/26/10	
36	5.1 Unit testing	5 days	Sun 3/14/10	Thu 3/18/10	34
37	5.1.1 Develop Unit Test Plans	3 days	Wed 3/17/10	Fri 3/19/10	
38	5.1.2 Test individual components	3 days	Thu 3/18/10	Mon 3/22/10	
39	5.1.3 Modify	2 days	Mon 3/22/10	Tue 3/23/10	
40	5.1.4 Re-Test	2 days	Mon 3/22/10	Tue 3/23/10	37
41	5.2 Integration Testing	8 days	Tue 3/23/10	Thu 4/1/10	38
42	5.2.1 Integrate individual components	2 days	Tue 3/23/10	Wed 3/24/10	
43	5.2.2 test integrated components	2 days	Wed 3/24/10	Thu 3/25/10	40
44	5.2.3 Modify	2 days	Thu 3/25/10	Fri 3/26/10	42
45	5.2.4 Retest	2 days	Fri 3/26/10	Mon 3/29/10	43
46	Testing complete	0 days	Fri 3/26/10	Fri 3/26/10	44
47	6.Documentation	8 days	Thu 4/1/10	Mon 4/12/10	46
48	7.Implementation	14 days	Tue 4/13/10	Fri 4/30/10	47


Aug 16, '09
Aug 23, '09
Aug 30, '09
Sep 6, '09
Sep 13, '09
Sep 20, '09
12 days
100%
8 days
70%
2 days
40%
Fri 8/28
19 days
75%
7 days
100%
5 days
80%
2 days
80%
2 days
95%
2 days
100%
Thu


Sep 27, '09
Oct 4, '09
Oct 11, '09
Oct 18, '09
Oct 25, '09
Nov 1, '09
Nov 8, '09
Nov 15, '09
28 days
95%
12 days
8 days
90%
2 days
85%
0%
5 days
0%
2 days
90%
3 days
100%
Wed 11/11


Jan 3, '10
Jan 10, '10
Jan 17, '10
Jan 24, '10
Jan 31, '10
Feb 7, '10
Feb 14, '10
Feb 21, '10
Feb 28, '10
Mar 7, '10
50 days
90%
12 days
90%
5 days
100%
0%
0%
0%
0%


Feb 21, '10
Feb 28, '10
Mar 7, '10
Mar 14, '10
Mar 21, '10
Mar 28, '10
21 days
95%
12 days
75%
100%
5 days
80%
Sun 3/14
10 days
20%
95%
100%
70%
75%
0%
8 days
85%


r 21, '10
Mar 28, '10
Apr 4, '10
Apr 11, '10
Apr 18, '10
Apr 25, '10
M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S
8 days
85%
95%
75%
45%
2 days
65%
Fri 3/26
8 days
0%
14 days
85%
Fri 4/30

7. Task Distribution

The following table shows the involvement of the different members in the respective tasks of the project.

TASK NAME	DURATION	START DATE	FINISH DATE	PEOPLE INVOLVED
1. Scope	12 days	17/8/2009	29/8/2009	4
1.1 Topic Selection	8 days	17/8/2009	25/8/2009	4
1.2 Consultation with Guide	2 days	26/8/2009	27/8/2009	5
1.3 Topic Finalization	2 days	28/8/2009	29/8/2009	4
Scope Complete	0 days	29/8/2009	29/8/2009	4
2. Analysis and Software Requirement	19 days	30/8/2009	20/9/2009	4
2.1 Gather Information from Net and Books	7 days	30/8/2009	7/9/2009	4
2.2 Review the Software Specication	5 days	8/9/2009	13/9/2009	4
2.3 Discuss and Finalize objectives	2 days	14/9/2009	15/9/2009	4
2.4 Prepare Rough Draft	2 days	16/9/2009	17/9/2009	4
2.5 Approval from Guide	2 days	18/9/2009	20/9/2009	5
Analysis Complete	0 days	20/9/2009	20/9/2009	4

3.Design	28 days	22/9/2009	28/10/2009	4
3.1 Determine Functions	12 days	22/9/2009	6/10/2009	4
3.1.1 Data Flow Diagram	8 days	7/10/2009	15/10/2009	4
3.1.2 Review Functional Requirements	2 days	18/10/2009	20/10/2009	4
3.1.3 approval by the guide	2 days	21/10/2009	22/10/2009	4
3.2 Prepare Software Requirements	5 days	23/10/2009	28/10/2009	4
3.2.1 prepare problem Statement	2 days	23/10/2009	24/10/2009	4
3.2.2 Explain Functionalities for subcomponents	3 days	25/10/2009	28/10/2009	4
Design phase complete	0 days	28/10/2009	28/10/2009	4
4.Development	50days	2/1/2010	10/3/2010	4
4.1 Study about Required platforms	16 days	2/1/2010	18/1/2010	4
4.1.1 Learn Functions In Matlab	5 days	2/1/2010	6/1/2010	4
4.1.2 Search algorithm	4 days	19/1/2010	22/1/2010	4
4.2 Review and Finalize Algorithms	5 days	30/1/2010	5/2/2010	4
4.3 Design Layout	6 days	6/2/2010	12/2/2010	4
4.3.1 Subcomponents of design module	3 days	6/2/2010	8/2/2010	4

4.3.2 Review and finalize Design	3 days	8/2/2010	10/2/2010	4
4.4 Develop Code	21 days	11/2/2010	10/3/2010	4
4.4.1 Implement Functionalities of the sub modules	12 days	11/2/2010	20/2/2010	4
4.4.2 Connect with backend	2 days	22/2/2010	23/2/2010	4
4.4.3 Search algorithm Implementation	7 days	25/3/2010	5/3/2010	4
Development phase Complete	0 days	5/3/2010	5/3/2010	4
5.Testing	16 days	6/3/2010	25/3/2010	4
5.1 Unit testing	10 days	6/3/2010	16/3/2010	4
5.1.1 Develop Unit Test Plans	3 days	6/3/2010	9/3/2010	4
5.1.2 Test individual components	3 days	10/3/2010	12/3/2010	4
5.2 Integration Testing	8 days	17/3/2010	30/3/2010	4
5.2.1 Integrate individual components	2 days	17/3/2010	18/3/2010	4
5.2.2 test integrated components	2 days	19/3/2010	20/3/2010	4
5.2.3 Modify	2 days	21/3/2010	28/3/2010	4
5.2.4 Retest	2 days	29/3/2010	30/3/2010	4
Testing complete	0 days	30/3/2010	30/3/2010	4
6.Documentation	7 days	10/4/2010	20/4/2010	4
7.Implementation	10 days	21/4/2010	30/4/2010	4

8. Future Work

8.1 Sensor Imagery for Night Vision: Color Visualization

Fusion of imagery from multiple sensors to create a color night vision capability. The fusion system architectures are based on biological models of the spatial and opponent color processes in the human retina and visual cortex, implemented as shunting center-surround feed-forward neural networks. Real-time implementation of the dual sensor fusion system combines imagery from either a low-light CCD camera or a short-wave infrared camera, with thermal long-wave infrared imagery. In the future this fusion architecture will include imagery from all three of these sensors, Visible/SWIR/LWIR, as well as a four sensor system using Visible/SWIR/MWIR/LWIR cameras and that too in real time. It also demonstrate how results from these multi-sensor fusion systems are used as inputs to an interactive tool for target designation, learning, and search based on a Fuzzy ARTMAP neural network.

9. Technologies Used

10.1 Matlab

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include:

- Math and computation
- Algorithm development
- Modeling, simulation, and prototyping
- Data analysis, exploration, and visualization
- Scientific and engineering graphics
- Application development, including graphical user interface building

In industry, MATLAB is the tool of choice for high-productivity research, development, and analysis. Toolboxes are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular classes of problems. Areas in which toolboxes are available include signal processing, control systems, neural networks, fuzzy logic, wavelets, simulation, and many others.

The MATLAB system consists of five main parts:

1. *Development Environment*: This is the set of tools and facilities that help us to use MATLAB functions and files. Many of these tools are graphical user interfaces. It includes the MATLAB desktop and Command Window, a command history, and browsers for viewing help, the workspace, files, and the search path.

2. *The MATLAB Mathematical Function Library*: This is a vast collection of computational algorithms ranging from elementary functions like sum, sine, cosine, and complex arithmetic, to more sophisticated functions like matrix inverse, matrix eigen values, Bessel functions, and fast Fourier transforms.
3. *The MATLAB Language*: This is a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features. It allows both "programming in the small" to rapidly create quick and dirty throw-away programs, and "programming in the large" to create complete large and complex application programs.
4. *Handle Graphics*: This is the MATLAB graphics system. It includes high-level commands for two-dimensional and three-dimensional data visualization, image processing, animation, and presentation graphics. It also includes low-level commands that allow you to fully customize the appearance of graphics as well as to build complete graphical user interfaces on your MATLAB applications.
5. *The MATLAB Application Program Interface (API)*: This is a library that allows you to write C and Fortran programs that interact with MATLAB. It include facilities for calling routines from MATLAB (dynamic linking), calling MATLAB as a computational engine, and for reading and writing MAT-files.

10. Acknowledgements

This project would not have been made possible without the able guidance and expert advice of our guide Prof. Anjali Malviya. We would also like to appreciate all the facilities provided to us by the IT department of Thadomal Shahani Engineering College.

11. References

[1] C. Xydeas, V. Petrovic, "Objective Pixel-Level Image Fusion Performance Measure", *IEE Electronics Letters*, vol. 36, no. 4, pp. 308-309, 2000.

[2]David L. Hall, James Llinas ''*Handbook Of Multisensor Data Fusion''* , 2001,CRC Press LLC

[3] D.A. Fay, A.M. Waxman, M. Aguilar, D.B. Ireland, J.P. Racamato, W.D. Ross, W.W. Streilein, and M.I. Braun "*Fusion of Multi-Sensor Imagery for Night Vision: Color Visualization, Target Learning and Search",* Massachusetts Institute of Technology Lincoln Laboratory.

[4] Dariu M. Gavrila, Martin Kunert, Ulrich Lages "*A multi-sensor approach for the protection of vulnerable traffic participants the PROTECTOR project",* at *IEEE Instrumentation and Measurement Technology Conference, Budapest, Hungary,* May 21-23, 2001.

[5] Vivek Maik, Jeongho Shin, Joonki Paik, "*Computer Analysis of Images and Patterns",* Volume 3691/2005, Springer Berlin / Heidelberg, 2005

[6] Z. Wang, A.C. Bovik, "A Universal Image Quality Index", *IEEE Signal Processing Letters*, vol. 9, no. 3,pp. 81-84,2002.