DETECTION AND SEGMENTATION OF MOVING OBJECTS IN VIDEO USING OPTICAL VECTOR FLOW ESTIMATION

A Thesis Submitted
to the College of Graduate Studies and Research
in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in the Department of Electrical and Computer Engineering
University of Saskatchewan

by

Rishabh Malhotra

Saskatoon, Saskatchewan, Canada

PERMISSION TO USE

The author has agreed that the Libraries of this University may make this thesis freely available for inspection. Moreover, the author has agreed that permission for copying of this thesis, in any manner, in whole or in part, for scholarly purposes may be granted by the professors who supervised this thesis work or, in their absence, by the Head of the Department of Electrical and Computer Engineering or the Dean of the College of Graduate Studies and Research at the University of Saskatchewan. It is understood that due recognition will be given to the author and the University of Saskatchewan in any use of the material in this thesis. Copying, publication, or use of this thesis or parts thereof for financial gain shall not be allowed without the author's written permission.

Request for permission to copy or to make any other use of material in this thesis in whole or in part should be addressed to:

Head of the Department of Electrical Engineering
57 Campus Drive
University of Saskatchewan
Saskatoon, Saskatchewan, Canada
S7N 5A9

ACKNOWLEDGMENTS

I express my sincere gratitude to Prof. Kunio Takaya for his constant guidance and assistance throughout this research project. I deeply appreciate and thank him for his unrelenting support, patience and encouragement throughout the course of my M.Sc. program. I am grateful that I was given an opportunity to work on this research.

Secondly, I would like to thank my advisory committee members, Prof. Sherif Faried, Prof. Carl McCrosky, Prof. Daniel Teng and Prof. Wahid Khan for their constructive comments and suggestions on my research work and thesis. I would also like to thank Prof. Reza Fotouhi for acting as my external examiner.

I would also like to thank the management and staff of Telecommunications Research Laboratories (TRLabs) for providing me with financial assistance and the use of their facilities throughout my research. Thanks to the faculty, fellow students and staff at the Department of Electrical and Computer Engineering who have helped me.

I wish to thank my family and the very special people in my life. A big thank you to my wife Ritu for her endless support, love, encouragement and especially her patience without which I could not have accomplished this work.

Finally, thanks to the College of Engineering and College of Graduate Studies and Research, University of Saskatchewan, for providing me with this wonderful opportunity!

ABSTRACT

The objective of this thesis is to detect and identify moving objects in a video sequence. The currently available techniques for motion estimation can be broadly categorized into two main classes: block matching methods and optical flow methods.

This thesis investigates the different motion estimation algorithms used for video processing applications. Among the available motion estimation methods, the Lucas Kanade Optical Flow Algorithm has been used in this thesis for detection of moving objects in a video sequence. Derivatives of image brightness with respect to x-direction, y-direction and time t are calculated to solve the Optical Flow Constraint Equation. The algorithm produces results in the form of horizontal and vertical components of optical flow velocity, u and v respectively. This optical flow velocity is measured in the form of vectors and has been used to segment the moving objects from the video sequence. The algorithm has been applied to different sets of synthetic and real video sequences.

This method has been modified to include parameters such as neighborhood size and Gaussian pyramid filtering which improve the motion estimation process. The concept of Gaussian pyramids has been used to simplify the complex video sequences and the optical flow algorithm has been applied to different levels of pyramids. The estimated motion derived from the difference in the optical flow vectors for moving objects and stationary background has been used to segment the moving objects in the video sequences. A combination of erosion and dilation techniques is then used to improve the quality of already segmented content.

The Lucas Kanade Optical Flow Algorithm along with other considered parameters produces encouraging motion estimation and segmentation results. The consistency of the algorithm has been tested by the usage of different types of motion and video sequences. Other contributions of this thesis also include a comparative analysis of the optical flow algorithm with other existing motion estimation and segmentation techniques. The comparison shows that there is need to achieve a balance between accuracy and computational speed for the implementation of any motion estimation algorithm in real time for video surveillance.

Table of Contents

List of Figures

List of Tables

ABBREVIATIONS AND ACRONYMS

$2-D$	Two Dimensional
R, G and B	Red, Green and Blue pixel Intensity
ANN	Artificial Neural Network
MFNN	Multi Layer Feed Forward Network
MPEG	Moving Picture Experts Group
MSE	Mean Square Error
SfM	Structure from Motion
SE	Structuring Element

1. INTRODUCTION

1.1 Background of Research

There has been a significant development in the area of digital image and video technology in the last twenty years. Because of the ever improving computer technology and high resolution cameras, the interest in video surveillance and similar computer vision technologies has soared. Tracking and detecting motion of objects in a video sequence is one of the most important tasks of the emerging computer vision technologies.

Video surveillance has become a very popular mode of monitoring, especially in secure or restricted access areas. Sensors which detect the moving objects are used in almost all video surveillance applications. By using the estimated motion, objects can be segmented. Segmentation of moving objects can be beneficial to segregate regions of interest from the background. For example, if a security/surveillance system is being used to segment a particular object (or person) in a crowd, segmentation techniques can be very useful in such cases.

Several camera and web camera manufacturers are using the concept of video surveillance by motion detectors. These motion detectors help users to keep a watchful eye on any area of interest in their absence. The motion detector triggers a recording whenever it senses any motion. The recording automatically stops when the motion goes below a certain level (depending on the sensitivity of the camera). The recorded motion can be viewed in the form of image snapshots or video files (each with a time and date stamp indicating the start of motion detection).

1.2 Motion Detection

This research project includes two main steps used to detect the moving objects in a video sequence. The first step is to estimate the motion in a video sequence or a frame. This step is important to determine the object(s) that should be segregated from the frame under con-

sideration. The second step is to segment the moving objects (determined from the first step) from the stationary object(s) or background in the considered frame/image. The next two sections present the basic concepts of estimation of motion and the subsequent segmentation of the moving objects from a frame/video image.

1.2.1 *Estimation of Motion*

In simple terms, motion can be defined as displacement of moving objects between the frames under consideration. The motion estimation models that have been proposed till now are based on measuring this displacement between objects. Currently existing motion detection and measurement techniques usually employ at least two frames of video images from a video sequence. The motion is usually computed by comparing any two consecutive frames (from a video sequence) at a given time. Some of the most widely used methods are:

- Optical Flow Method
- Motion Vector Estimation
- Image Subtraction

Some of these methods have been discussed in detail in the next chapter.

1.2.2 *Segmentation of Moving Objects*

Segmentation of moving objects is done in order to separate the moving objects from their stationary background. Image segmentation has found wide applications in the field of medical and biomedical imaging. Various methods have been proposed and tested for segmentation of moving objects. Some of these methods [24] [25] can be broadly classified on the basis of the principle as:

- Clustering
- Edge Detection

- Histogram-Based
- Level Set
- Model Based
- Thresholding
- Neural Networks and other Artificial Intelligence Based Techniques

Some of these methods have been discussed in further detail in Chapter 2.

1.3 Objective and Methodology of the Research

The motivation of this research project comes from the application of this research in video surveillance where only one or more moving objects are of interest and thus should be segmented out from the stationary background. It is imperative that good motion detection results are obtained before the moving objects can be segregated from the stationary background. Different methods have been proposed for tracking moving objects in video sequences. The objective of this research project is to analyze the response of these different methods for various types of motions with the ultimate objective of segmenting the moving objects of a video sequence by using one of the available methods. The main application of this technology is in computer vision, specifically in video surveillance, video processing and 2.5 dimensional structure estimation from motion (SfM).

The objective of this research project is to identify and segment moving objects in a video clip. The methodology that has been followed to achieve this research objective can be broadly described in the following five main steps:

1.) Estimation of Motion

Different methods of motion estimation and segmentation of images have been studied and analyzed. The results obtained from these methods have been presented. Ultimately, the Lucas Kanade Optical Vector Flow Algorithm has been chosen to obtain the relative motion between any two video frames. The estimated motion is in the form of optical flow vectors. The obtained

optical flow vectors are then superimposed on the original frame(s) indicating motion or moving objects in the frame(s). The magnitude and direction of the optical flow vectors indicate the relative motion between the two frames under consideration.

Depending on the complexity of the chosen video sequence, Gaussian smoothing has been performed and the Lucas Kanade Optical Flow Algorithm has been applied to the different smoothed levels of Gaussian pyramid images. A comparative analysis has been performed on the obtained Gaussian levels to evaluate the results obtained by using different Gaussian pyramid levels. Further, depending on the amount of motion in the two image frames chosen, the neighborhood window sizes have been varied. The results obtained from this process have been compared and presented.

2.) Analysis of the response of Lucas Kanade Method for different Video Sequences

Practically, the objects in a video sequence can move in different directions. Therefore, video sequences for different types of motion have been considered and the results obtained from the selected algorithm have been evaluated. The included sequences have translational motion of moving objects, independent translational and rotary motion of moving objects, etc. The study of video sequences under different camera parameter variances including pan, tilt, zoom-in and zoom-out of camera motions has also been undertaken.

3.) Segmentation of identified Moving Objects

The method of thresholding has been chosen and the magnitude difference between the optical flow vectors of moving and stationary objects has been used to segment the moving objects. This has been done by applying an optical vector threshold to differentiate the moving objects from their relatively stationary counterparts in the video frame(s). Object segmentation is then performed on the chosen frames to clearly depict the moving objects.

4.) Enhancement of Segmented Content

Two techniques, known as erosion and dilation have been used to enhance the already segmented content obtained in step 3. Either of these techniques can be used to improve segmented image quality (depending on the considered image sequence). In this thesis, a combination of these techniques (in a four stage process) has been tested to analyze the effect of these techniques on

the segmented images. The results obtained by this process have been observed.

5.) Comparative Analysis with other Techniques

The chosen Lucas Kanade Optical Flow Algorithm has been compared with other existing techniques - block matching method and artificial neural network based technique. The motion estimation results obtained from block matching method have been compared with the optical flow results. Segmentation results using Neural Networks have been compared with results obtained by the considered thresholding algorithm. A comparative analysis of these results has been presented.

1.4 Organization of the Thesis

This thesis has been divided into six chapters and two appendices.

The first chapter presents the background of the research project and its objectives. A brief introduction to the estimation of motion in video sequences has been discussed. This chapter also introduces the concept of segmentation of moving objects after the detection of motion. Further, the outline of each chapter and its contents has been presented.

Chapter two describes the different types of motions that are usually seen in different video sequences. The various methods used to detect moving objects in an image/video sequence have been described. The methods include Block Matching methods, Optical Flow methods and other alternative algorithms based on neural networks and artificial intelligence. The challenges faced during accurate detection of motion have been discussed as well.

The Optical Flow technique (Lucas Kanade Method) that has been used in this research project has been discussed in detail in chapter three. This chapter also presents the mathematical calculations and modifications involved in this algorithm. It further discusses the important parameters of the algorithm which affect accurate detection of motion. Challenges faced during the implementation of an Optical Flow technique such as the aperture problem have also been discussed.

Chapter four presents the implementation of the Lucas Kanade Method for detection of

motion in image sequences. The results obtained by the application of the proposed technique to synthetic and real images have been presented. The proposed algorithm has been applied to different types of motion and the results have been shown. The concept of Gaussian Pyramids has been applied to these image sequences and the chapter also presents a comparative analysis of the obtained results.

The motion detection results obtained from different image sequences have been used for segmentation of moving objects in Chapter five. The chapter further presents the results of enhancement techniques for smoothing of the segmented background in the obtained results. A comparative analysis of the presented technique with other techniques such as block-matching method and neural network technique has been presented in this chapter.

Chapter six concludes the thesis and presents the summary of this research. Further, it provides suggestions for future work related to this research project.

Appendix A outlines the MATLAB commands that have been used in this project. It also includes the commands taken from the Image Processing Toolbox of MATLAB.

The MATLAB code and other parameters used for the implementation of the Lucas Kanade algorithm have been presented in Appendix B.

1.5 Summary

This chapter introduced the basic concept of motion detection in image/video sequences and the subsequent segmentation of moving objects from their background. It further outlines the objective of this research project. The organization of this thesis into six chapter and two appendices has been discussed.

2. MOTION ESTIMATION

2.1 Introduction

The previous chapter introduced the concept of motion estimation in video sequences. The different types of motion and the methods used to detect it have been discussed in detail in this chapter. Further, the challenges faced in the determination of accurate movement and motion detection between two frames of a video sequence have also been discussed.

2.2 Characterization of Motion

Objects in a video can move in different ways. Therefore, it is important to understand some of the types of motions that may exist in a video sequence. Some of the types of motion have been described in the subsequent sub-sections.

2.2.1 *Pan Motion*

This type of motion occurs when the object in a video sequence moves linearly from the left direction to right or vice versa. The motion in the video can also be as a result of the movement of the camera instead of the object (in similar directions). When such a motion occurs in a video, the estimated motion should be largely in a horizontal direction with relatively low or no motion detection in the vertical direction.

2.2.2 *Tilt Motion*

Tilt motion implies the movement of an object in the vertical direction. This means that the object in the video sequence moves either upwards or downwards. As in the previous case on Pan motion, this motion can be a result of the camera movement as well. Consequently, when a motion estimation algorithm is applied to such a video sequence, the results should be concentrated in the vertical direction as compared to the horizontal direction.

2.2.3 *Zoom Motion*

Zoom motion can be of two types: Zoom-in and Zoom-out. As the name suggests, zooming in occurs when an object in a video sequence in concentrated upon. In the subsequent frames of a video sequence, the object of interest increases or enlarges in size. In most of the cases, the other objects in the frame tend to move out in the next frame. Motion detection for zoom in cases results in estimation of motion over the whole frame. This is because the camera moves all over the frame while trying to zoom in onto one object. Further, the results should be concentrated towards the centre of the frame indicating the zooming in motion.

On the other hand, zoom out motion results in lesser concentration on objects of interest. Zoom out leads to a decrease in size of objects in the succeeding frames. This may lead to insertion of more objects as the camera zooms out. Evidently, the results obtained for such a motion should be spread over the whole frame, with the vectors moving away from the centre of the frame indicating zooming out motion of the camera.

Zoom-in and zoom-out motions can also be emulated if the object of interest moves closer or away respectively, from the camera while the camera focus remains the same.

2.2.4 *Rotary Motion*

Rotary motion indicates the motion of an object around a horizontal or vertical axis in a video sequence. Since an object moving in a rotary motion has a constant axis, the results obtained from such a data set are usually concentrated on the edges of the moving object. This is true especially in cases when the moving object has a smooth surface or a constant color. In cases of irregular colored objects, the motion results could be concentrated over the whole object.

Further, it should be noted that the objects in a video sequence can move in a combination of these motions. In such cases, the estimated motion is a resultant of the individual motion results.

2.3 Methods for Motion Detection

Different image processing techniques can be used to detect and track moving objects in a video sequence. These techniques can be categorized into two broad classes: feature based and motion based [1] [34].

Feature based techniques differentiate moving objects from the stationary ones on the basis of one or more object features such as shape, color, profile etc. The motion based techniques (Optical Flow technique) usually compare relative or absolute motion between moving and stationary objects.

Even though these two methods remain most popular, another type of technique which has gained a lot of interest in recent times is based on artificial intelligence. The subsequent sections discuss some of the methods of these three techniques in detail.

2.3.1 *Block Matching Methods*

Block matching methods are one of the most popular feature based methods used to estimate motion. The method is based on the measurement of absolute motion between objects in two video frames. All block matching methods start with the division of any video frame into macro-blocks of desired dimensions. The method usually uses two adjacent MPEG video frames, referred to as reference frame and target frame as shown in Figure 2.1. The two frames are then divided into macro-blocks. These corresponding macro-blocks between the two frames are compared to estimate motion in the form of vectors. The MPEG video image coding conducts the search by using macro blocks to search a best match of a block (belonging to the reference frame) in the target frame. Generally, a 16 pixel by 16 pixel macro block is used for gray scale images and an 8 pixel by 8 pixel macro block is used for colored images [27] [28].

The motion vector search algorithm uses an error function (e.g., sum of absolute differences, mean square error etc.) to search the best match macro-block in the target image frame with respect to the reference image frame. The displacement found by matching the reference and target macro blocks defines the motion vector of that pixel. The pixel could be any pixel belonging to the reference macro block (top left pixel of macro block, bottom left pixel of macro block etc.).

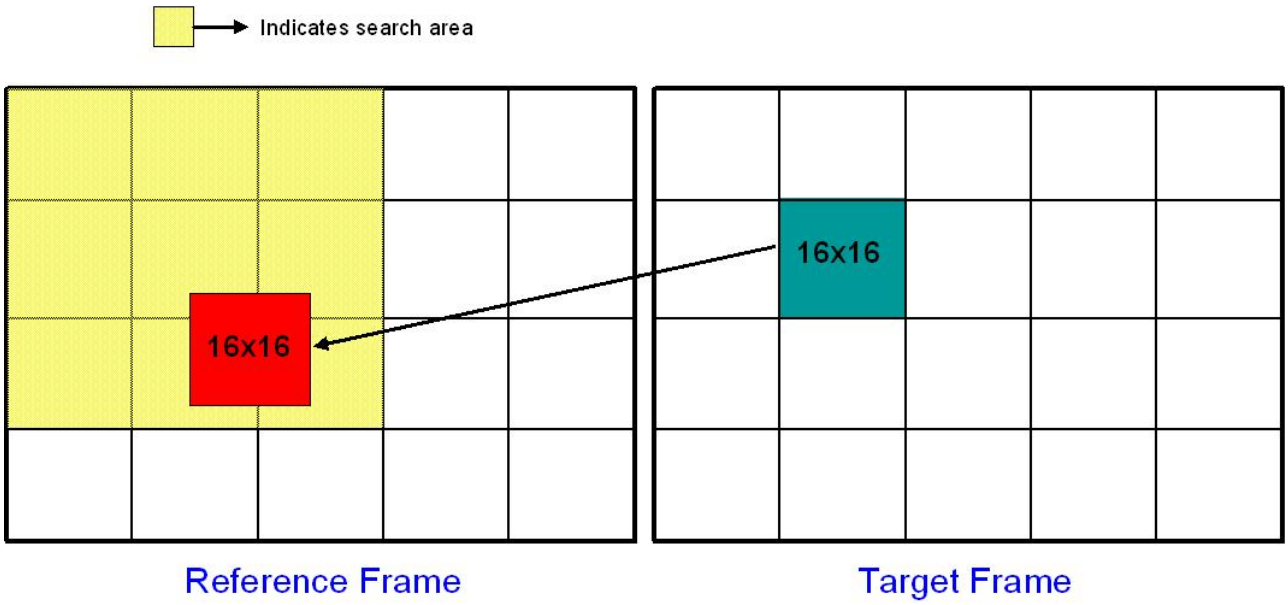

Figure 2.1 Block Matching Technique

There are a large number of different block matching search algorithms for motion estimation [2] that have been reviewed and tested.

Most of the search algorithms are based on defining a square search area (of 32, 64 or 128 pixels etc.) in the target frame. Since the movements of the objects vary from one video sequence to another, the motion vector search algorithm is primarily dependent on the type of video sequence under consideration. The search area should thus be chosen depending on the movement in the two adjacent frames. For example, if the motion of moving objects in the frames under consideration is large, a larger search area needs to be considered. Further, if a video sequence has been split at a lower rate (in terms of frames/second), it is likely that the movement between frames is higher. A larger search area should be chosen in such cases as well.

Because of the importance of a search area in motion estimation, the search algorithms are generally classified on the basis of the method chosen to define the search area. Some of these methods include: three step search [11], four step search [20], spiral search [36], 2-D logarithmic search [10], orthogonal search [21], cross search [7], one-at-time search [23], full search [4] etc. Some of these algorithms have been described in detail in the following three sub-sections. In addition to the search range, the computational complexity of these motion

estimation techniques is also dependant on other factors such as the search algorithm used, cost function used for evaluation, size of macro blocks etc. Therefore, these parameters need to be modified accordingly for different video sequences under consideration.

Exhaustive Search

Exhaustive search (also known as full search) is the method of comparing a macro-block in the reference frame with all the macro-blocks in the target frame. The parameter or a combination of parameters of the reference macro-block (e.g. pixel intensity, hue etc.) is compared with each macro-block of the target frame and the error between them is calculated using an appropriate error function (e.g. mean square error, absolute error etc.). All the errors are analyzed and the macro-block in the target frame with the minimum error is thus chosen. The displacement between the reference macro-block and the chosen target macro-block is termed as the estimated motion. The same process is repeated for each of the macro-blocks in the reference frame and the motion vectors are estimated accordingly for the full frame.

Step Search

Step search is the method of conducting the motion search in a number of chosen steps. A search area is chosen around each macro-block by specifying the number of pixels and is dependent on the relative movement between video frames. The search is then conducted in the macro-blocks belonging only to that particular search area around each macro-block in the target frame. The number of steps chosen determines the number of search iterations around each macro-block before the algorithm is stopped. The usual number of steps used for the search is three or four.

$2-D$ ***Logarithmic Search***

$2-D$ Logarithmic search is another form of step search. The main difference between the regular step search and this type of logarithmic search is the progressively reducing search area around the macro-block in the target frame. The defined search area in this type of search is usually a multiple (or sub-multiple) of the size of the macro-block. For example, for a macro-

block of 16×16 pixels, a search area of 32 pixels, 16 pixels or 8 pixels can be chosen. The search area is reduced by half in each subsequent iteration and a full search is conducted in the specified search area. The procedure of calculating errors to choose the block with the least error with respect to the reference macro-block is conducted and the best block match is eventually chosen in the target frame.

Most of the algorithms use a combination of these search methods to optimize the advantages of each of these methods. Further, one method can be used as a sub-method of the other. For example, an exhaustive search can be followed by a $2-D$ logarithmic search. The exhaustive search finds the best matching macro-block in the whole target frame and the $2-D$ logarithmic search is able to micro-search to find the best macro-block within the search area around the chosen macro-block.

2.3.2 *Optical Flow Methods*

Optical Flow is defined as the $2-D$ distribution of apparent velocities of movement of intensity patterns in an image plane [9] [26]. The Optical Flow Algorithm is based on relative motion rather than absolute motion, as in the case of motion vector search method. Using this method, the direction and speed of moving objects from one image to another is obtained in terms of velocity vectors (horizontal and vertical velocity vector components). The concept of optical vector flow is used to estimate the motion of pixels (and thus objects) in an image sequence within a visual representation. The motion represented by these optical flow vectors originate or terminate at pixels in the sequence of image frames derived from an MPEG movie. This depicts a dense vector field across all moving pixels in each frame [9]. This method is based on the assumption that the displacement between the two frames is relatively small.

Optical flow methodologies have been classified into three main groups: differential, matching and spectral (or phase correlation) methods [6]. The differential optical vector flow evaluation technique is used in this research project. This method works very well for highly textured images due to the ease of calculation of local derivatives (gradients) [5]. There are several methods that exist for determining differential optical flow. Some of the more commonly used methods are: the Horn Schunck method [9] and the Lucas Kanade method of estimating optical flow [12] [13] [14].

The mathematical implementation of the optical flow method can be derived from its definition. Specific modifications are made based on the particular method under consideration. Optical flow is defined as an apparent motion of image brightness patterns in an image sequence [9] [19].

As shown in Figure 2.2, consider a pixel of a moving object at point A (pixel coordinates (x, y) at time t) with image brightness equal to $I(x, y, t)$.

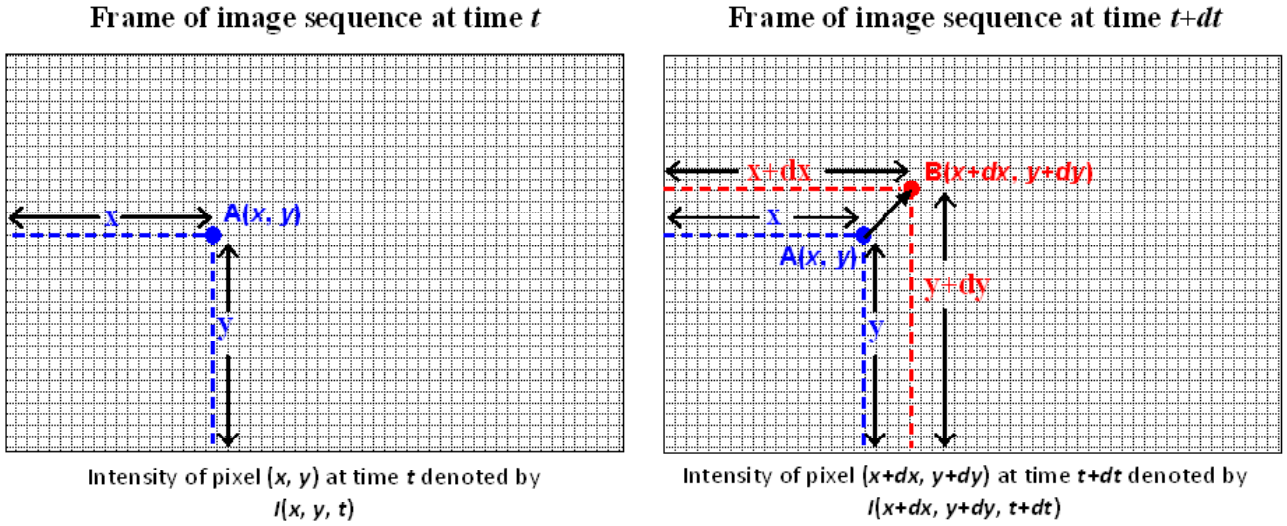

Figure 2.2 Brightness Constancy in the Optical Flow Algorithm

Because of object movement between frames, this pixel moves to point B (new pixel coordinates being $(x + dx, y + dy)$ at time $t + dt$); implying that the pixel coordinates (x, y) change in time dt to $(x + dx, y + dy)$. It is assumed that the image intensity or brightness $I(x, y, t)$ of the pixel with coordinates (x, y) is constant (brightness constancy assumption) in both these frames. Similarly, the algorithm assumes that the brightness of every point of a moving or static object does not change in time with changing frames.

The brightness constancy assumption thus results in the following equation:

$$I(x, y, t) = I(x + dx, y + dy, t + dt) \tag{2.1}$$

Using Taylor series expansion (only considering first order terms) the right hand side of the

above equation results in:

$$I(x+dx, y+dy, t+dt) = I(x,y,t) + \frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t}dt + \cdots \tag{2.2}$$

In the above equation, "$\cdots$" refers to higher order terms which can be neglected.

Thus, substituing equation 2.2 into equation 2.1 and cancelling common terms, we get:

$$\frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t}dt = 0 \tag{2.3}$$

Dividing equation 2.3 above by dt results in:

$$\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0 \tag{2.4}$$

This implies,

$$\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} = -\frac{\partial I}{\partial t} \tag{2.5}$$

The components of optical flow in the horizontal and vertical directions respectively can be denoted as

$$\frac{dx}{dt} = u \tag{2.6}$$

and

$$\frac{dy}{dt} = v \tag{2.7}$$

This results in the following equation, which is also known as the Optical Flow Constraint Equation.

$$\frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v = -\frac{\partial I}{\partial t} \tag{2.8}$$

Horn Schunck Method

The Horn Schunck method [9] can be derived from equation 2.4. Using the following notations:

$$\frac{\partial I}{\partial x} = E_x \tag{2.9}$$

$$\frac{\partial I}{\partial y} = E_y \tag{2.10}$$

$$\frac{\partial I}{\partial t} = E_t \tag{2.11}$$

Equation 2.8 can now be re-written as:

$$E_x u + E_y v = -E_t \tag{2.12}$$

In the above equation, the terms E_x, E_y and E_t are the partial derivates of the image brightness with respect to the x-direction, y-direction and time t respectively. Equation 2.12 can be further written as:

$$(E_x, E_y).(u, v) = -E_t \tag{2.13}$$

Equation 2.13 implies that the component of the movement, p in the direction of the brightness gradient (E_x, E_y) is

$$p = -\frac{E_t}{\sqrt{(E_x)^2 + (E_y)^2}} \tag{2.14}$$

It has been suggested that it is not possible to determine the component of the movement in the direction of the iso-brightness contours, at right angles to the brightness gradient. As a consequence, the flow velocity (u, v) cannot be computed locally without introducing additional constraints.

This method then introduces additional smoothness constraints SC_1 and SC_2 to minimize the square of the magnitude of the gradient of the optical flow velocity, i.e.

$$SC_1 = \left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 \tag{2.15}$$

and

$$SC_2 = \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 \tag{2.16}$$

These equations have been used to implement the algorithm by estimating the derivatives E_x, E_y and E_t from the discrete set of image brightness measurements. The values of optical flow vectors u and v is then estimated using equation 2.13.

Lucas Kanade Method

The Lucas Kanade equation [12] [13] [14] is derived from the optical flow equation given in equation 2.8. We use the following notations:

$$\frac{\partial I}{\partial x} = I_x \tag{2.17}$$

$$\frac{\partial I}{\partial y} = I_y \tag{2.18}$$

$$\frac{\partial I}{\partial t} = I_t \tag{2.19}$$

As in the case of the Horn Schunck method, the terms I_x, I_y and I_t are the partial derivates of the image brightness with respect to x-direction, y-direction and time t. Replacing these variables from 2.17, 2.18 and 2.19 in the Optical Flow Constraint Equation (Equation 2.8), thus results in:

$$I_x u + I_y v = -I_t \tag{2.20}$$

It can be seen that the Optical Flow Constraint Equation has two unknown variables (u and v) in one equation. Equation 2.20 can be expressed in matrix form as:

$$\begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -I_t \tag{2.21}$$

The Lucas Kanade method assumes the motion between two video frames to be relatively small. Therefore, the neighborhood assumed for relative motion between frames is also assumed small. Modifying the equation by assuming constant (u, v) in a small neighborhood:

$$\begin{bmatrix} I_{x1} & I_{y1} \\ I_{x2} & I_{y2} \\ \vdots & \vdots \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_{t1} \\ I_{t2} \\ \vdots \end{bmatrix} \tag{2.22}$$

where sub-scripts of I_x, I_y and I_t represent the pixels of the considered image, thus resulting in higher number of equations than the number of unknowns, eventually helping to solve for the unknowns.

Again, if the following notations are assumed,

$$A = \begin{bmatrix} I_{x1} & I_{y1} \\ I_{x2} & I_{y2} \\ \vdots & \vdots \end{bmatrix} \tag{2.23}$$

$$\overrightarrow{u} = \begin{bmatrix} u \\ v \end{bmatrix} \tag{2.24}$$

$$b = - \begin{bmatrix} I_{t1} \\ I_{t2} \\ \vdots \end{bmatrix} \tag{2.25}$$

Equation 2.21 now transforms to,

$$A\overrightarrow{u} = b \tag{2.26}$$

The goal of the Lucas Kanade method for Optical Vector Flow calculation is to minimize $\|A\overrightarrow{u} - b\|^2$. Using the method of least squares, and by multiplying with the transpose of matrix A, the term $A\overrightarrow{u} = b$ can be written as:

$$\underbrace{A^T A}_{(2\times2)} \underbrace{\overrightarrow{u}}_{(2\times1)} = \underbrace{A^T b}_{(2\times1)} \tag{2.27}$$

Or,

$$\overrightarrow{u} = (A^T A)^{-1} A^T b \tag{2.28}$$

The matrix $A^T A = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}$ should be invertible (that is, no zero Eigen values).

The term $(A^T A)^{-1} A^T$ is known as the pseudo-inverse of the matrix A and can be calculated using the pixel intensity values I_x and I_y of all the pixels of any video frame.

Due to its robustness, the Lucas Kanade method is being used in this research project. This method has been implemented in MATLAB using the Image Processing Toolbox [30].

2.3.3 *Alternative Methods*

In addition to the methods described in the previous sections, there are other alternative methods such as neural networks [8] and fuzzy logic [22] that have been used for various image processing applications. Artificial Neural Networks (ANNs) or Neural Networks are modeled on the human brain and have the ability to learn from a representative set of patterns and apply the learning to different system conditions. On the other hand, fuzzy logic forms a theoretical foundation for reasoning about imprecise propositions; such reasoning has been referred to as approximate reasoning [22].

Such unconventional approaches have the ability to overcome various challenges (such as lighting change, camera movement etc.) that usual methods (based on motion detection) face. The separation of moving objects in a MPEG video frame from the background is essentially a pattern classification problem. Some of these methods have been used and implemented successfully for solving numerous pattern classification problems. The capability of an ANN to learn and generalize has been utilized in numerous applications of image processing [28]. The method undertaken in this project has been compared with a neural network approach for image segmentation and some results have been presented in Chapter 5.

ANNs can be classified on the basis of the type of neuron connections and data that flows through the different layers of the neural network. Feed forward networks are the most commonly used neural networks for pattern classification problems. Some of these networks [8] have been described in the next few sections.

Multi-Layer Feed Forward Networks

Multi-Layer Feed Forward Networks comprise of one input layer, one output layer and one or more hidden layers. The input layer receives the input data set, which is multiplied by the input weights and mapped through a transfer function [8]. The output of the first layer becomes the input for the second layer (usually a hidden layer) and the process moves on through different layers of neurons. The different layers can be either partially or fully connected to its adjacent layer. The most commonly used learning algorithm with these kinds of neural networks is

the back-propagation algorithm, where the computed errors between the target output and the obtained output are fed back into the neural network to update its weights.

Radial Basis Networks

Radial basis networks usually use three layers, one input, one hidden and one output layer. The transfer function used in the hidden layer is usually a non-linear function whereas the output layer usually uses a linear function. Radial basis networks employ more neurons than standard multi-layer perceptrons and work very well in applications where a large number of training data sets are available [8].

Kohonen Self Organizing Maps

Kohonen Self Organizing Maps [8] do not use the principle of updating the weights according to a pre-set pattern. In these networks, the neurons of the network compete for the opportunity to learn and update the weights according to the "winning" neuron. The output of the neuron with the largest output in iteration is considered and only the connection weights of this neuron (as well as all the other neurons connected to it) are updated.

2.4 Challenges Associated with Accurate Motion Detection

There are many challenges that need to be overcome for accurate estimation of motion. Some of these challenges are:

(a) Change in pixel intensity

Motion Estimation algorithms can produce erroneous results because of pixel intensity variance of stationary objects between two scenes. This variance may exist in a video sequence due to either background lighting change or slight camera movement. In such cases, even though the background may be stationary, the motion detection algorithm may detect it as varying. Therefore, the motion detection approach needs to be robust against changes in scene illumination, but sensitive enough to detect the moving objects.

(b) Speed of movement

Another challenge faced when tracking motion between any two consecutive frames of a video is to detect motion between objects which are moving fast relative to the video frame rate. For example, the optical flow algorithm assumes the relative movement between frames to be small. Therefore, for rapidly moving objects, it is important to ensure that a high video frame rate is used.

(c) Varying background

Constantly changing backgrounds pose another major challenge in motion estimation. The variation in background can be a result of constantly moving background objects such as swaying leaves on trees, or rain. Further, radical background changes such as video surveillance in buildings for human activity recognition (where humans appear and depart intermittently in scenes) may exist in video sequences. Therefore, the motion detection algorithm used should be able to adapt and perform in such circumstances.

2.5 Image Segmentation

Image Segmentation is defined as the process of dividing an image into homogenous regions with respect to a chosen property [18]. The image segmentation methods can be categorized into three main approaches [16] [18] [32] and the details of these methods have been given in the next three sub-sections.

2.5.1 *Histogram or Threshold based Approach*

A histogram or threshold based approach assumes a certain range of the pixel intensities. This enables the separation of the image under consideration into two separate classes (i.e., the region or object of interest and the background) using a specified threshold. Most of the threshold or histogram approaches deal with gray level images [18]. This can be attributed to the fact that the gray level images have a pixel intensity range of $1 - 256$ (or $0 - 255$, depending on the program being used). Thus, by setting a threshold in this range according to the image or video frame under consideration, the moving objects can be separated from the stationary background (or vice versa). In colored images, the presence of three-dimensional pixel intensities because of R,

G and B values (each with a possible 256 values) makes the process complicated to implement.

2.5.2 *Edge Detection*

Edge or boundary detection is a method to detect the boundaries of the objects in an image or frame. An edge is characterized by a significant local change in image intensities [18]. By doing this, the objects in the frame are indirectly defined and can be separated. This method is based on the assumption that the change in pixel intensities between adjacent pixels in the region/object of interest is insignificant as compared to the background which is to be separated.

2.5.3 *Region or Pixel Based Classification*

This type of approach is based on the homogeneity of pixels belonging to the same area of an image or video frame. It is based on the assumption that the pixels belonging to the same homogenous region tend to be more alike than pixels belonging to other homogenous regions [18]. The pixels belonging to a homogeneous region are split or merged depending on the case under consideration. The method is similar to clustering techniques which place the pixels with some similarities together in groups.

2.6 The Aperture Problem

In motion estimation, the aperture problem occurs when motion is viewed through a small aperture such as a camera. It has been observed that when a moving object is observed through an aperture, only the motion orthogonal to the object can be measured. This implies that it is difficult and in some cases, impossible to measure the lateral movement of an object when the image is observed through a small aperture. Four different cases have been considered to explain this problem. Consider a uniform textured image with stripes as shown in Figure 2.3.

Figure 2.3(a) shows the case when the image is observed with the aperture in the centre of the image. The pattern observed by the aperture (inside the aperture) should be noted. The aperture has been moved in the upwards direction (tilt motion) in Figure 2.3(b). The dimensions and location of the textured image have been kept constant. In Figure 2.3(c), the aperture has been moved in the right hand direction, which signifies the left-right movement (pan motion) of an

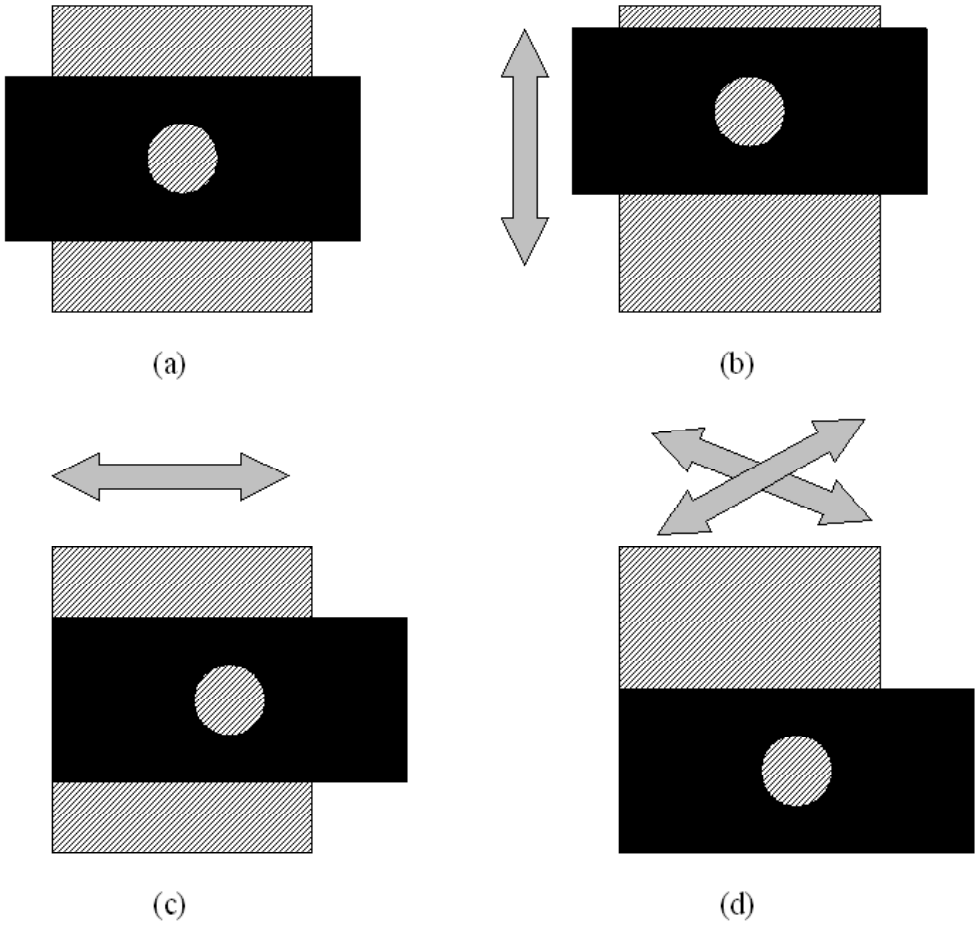

Figure 2.3 Aperture Problem in Motion Estimation

object. The pan and tilt motions have been combined in Figure 2.3(d). For better understanding of the problem, in all the cases, the aperture has been moved in such a way that the pattern seen inside the aperture is exactly the same as seen in Figure 2.3(a).

Now, if only the pattern as seen inside the aperture is observed, it appears that there has been no movement in these four different cases. Therefore, when a motion estimation algorithm is applied to recognize motion between any two such cases, the results are ambiguous. For algorithms based only on pixel intensities, the estimated motion in most of the cases will be zero. However, this is not true since the aperture has clearly moved over the image in each of the four cases. This inability of a motion estimation algorithm to record movement because of a

small aperture is known as the aperture problem. The obvious solution to this problem is using a bigger aperture. However, as most of the videos and images are captured by a camera, this solution of using a bigger aperture is not possible.

This problem can be handled to some extent if the motion between two images under consideration is relatively small. This will cause a small difference in the patterns as seen by the aperture, thus, enabling the motion detection algorithm to detect motion. Evidently, this is relatively difficult to implement with block matching techniques. Most block matching techniques use a block size of eight (8) or sixteen (16) pixels, which is too large to record small movements. Using a smaller block size increases the motion estimation time significantly, and therefore is not a plausible solution.

Optical Flow algorithms are able to counteract the effects of this aperture problem. This is attributed to the fact that these algorithms are based on the relative motion between pixels. Further, an optical flow algorithm is based on the assumption that the motion between any two images is relatively small. In this project, the optical flow algorithm has been used to detect motion between two images. All the cases considered in this project have small movements in any set of images. This has been achieved by using a very high frame rate (per second) for all videos.

2.7 Summary

This chapter presented the literature review done to study the various existing methods used for image processing. The different types of motion existing in video sequences have been discussed. The chapter also presented the block matching and optical flow methods used for estimation of motion. The challenges associated with accurate estimation of motion, such as change in pixel intensity, speed of camera movement and varying background have been addressed. Further, the different approaches used for image segmentation, such as thresholding, edge detection and pixel based classification have been presented.

3. OPTICAL VECTOR FLOW ESTIMATION ALGORITHM

3.1 Introduction

Some of the existing methods for motion detection and image segmentation have been presented in the previous chapter. The Lucas Kanade method of optical flow that has been used in this project has been described in detail in this chapter. The various factors influencing the optical flow estimation have been covered here as well.

3.2 Lucas Kanade Method of Optical Flow

This section presents the implementation of the Lucas Kanade algorithm as explained previously in section 2.3.2. Lucas Kanade method for optical flow has been designed for any MPEG video sequence. For the application of this algorithm, the considered video sequence is split into frames (or images) using an appropriate frame rate. As mentioned before, the algorithm has been designed in MATLAB using the Image Processing Toolbox. It is also important to note here that the Image Processing Toolbox uses images (rather than frames) for most of the in-built functions. Therefore, wherever applicable, the video frames have been converted into images (in .jpg or .bmp format). Figure 3.1 shows the flowchart for the step by step implementation of the Lucas Kanade method.

The process has been categorized into three main steps.

(a) Selection of the MPEG video frames and other parameters:

The first step of implementation is the selection of the MPEG video frames for which the optical flow has to be estimated. The optical flow algorithm is applicable only on gray level images or frames. The frames are selected and the colored images/frames are converted into

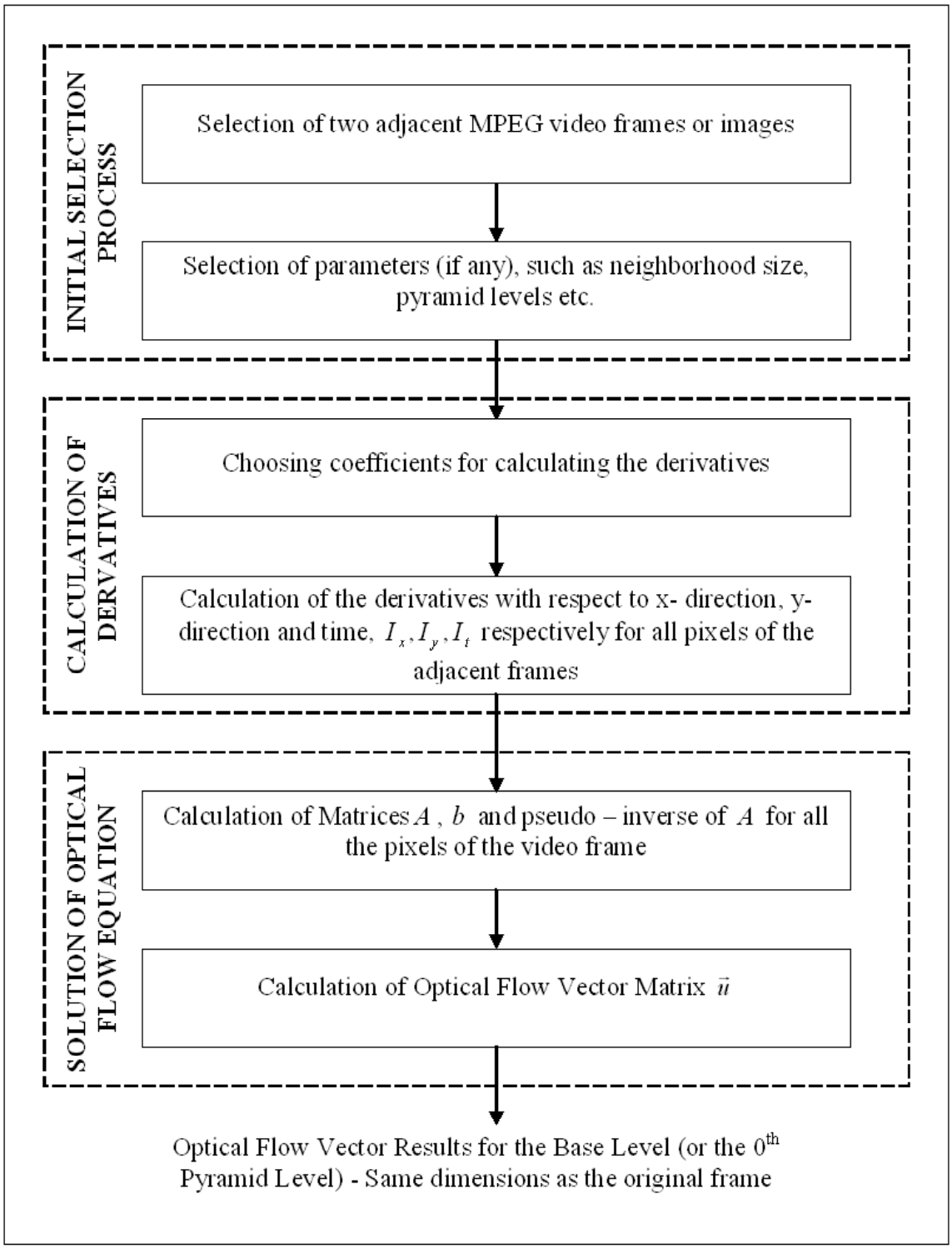

Figure 3.1 Implementation of the Lucas Kanade Method for the Base Level Case

gray level frames. The pixel intensities of the frame are saved for further calculations. The initial selection process also includes the selection of parameters such as the neighborhood size and pyramid levels, if any. These parameters have been discussed in detail in section 3.3.

(b) Calculations of Derivatives:

The next part of the optical flow method includes all the calculations of the partial derivates of the image brightness with respect to x-direction, y-direction and time t, i.e. I_x, I_y and I_t respectively. The filters' coefficients have been calculated using MALTAB and a two-dimensional convolution has been applied to the selected frames to calculate the partial derivatives. The code used for the calculation of derivatives and the values of the filter coefficients generated in MATLAB have been given in Appendix B. Once the derivatives are calculated, the two matrices A and b are calculated, which can then be used to solve the Optical Flow Equation.

(c) Solution of the Optical Flow Equation:

The Optical Flow Equation $A\overrightarrow{u} = b$ is solved by using the method of least squares. The pseudo-inverse of the matrix A is calculated to obtain the value of optical flow vector matrix $\overrightarrow{u}$. This matrix gives the value of the optical flow vectors u and v in the horizontal and vertical direction respectively. The magnitude of these vectors varies for different data sets. The threshold to segment the moving objects can be set either according to the individual optical flow vectors (in x and y direction) or the magnitude of the vector ($= \sqrt{u^2 + v^2}$).

3.3 Factors Influencing the Optical Flow

There are various factors that affect the calculation of optical flow between two images or frames. Two of the important factors have been considered in this project. These two factors are the neighborhood size (also known as window size) and the pyramid levels. These factors have been discussed in detail in sections 3.3.1 and 3.3.2 respectively.

3.3.1 *Neighborhood window size*

The optical flow method is based on the relative motion between pixels. Therefore, the movement of an object relative to its surroundings (neighborhood) plays a vital role in determin-

ing the optical flow vectors. Increasing the window size can sometimes help in acquiring more local neighborhood information, thus helping in determining the optical flow more accurately. Different window sizes have been considered in this research project.

A comparative analysis of optical flow estimation performed for the type of video sequences mentioned shows that a neighborhood window size of seven (7) pixels is enough to accurately obtain the optical flow vectors for the considered synthetic images. For real image and more complex data sets, a neighborhood window size of nine (9) pixels or eleven (11) pixels seemed to perform well for optical flow determination. The results shown in the next chapter consider these values for the neighbourhood sizes.

The variation in the window sizes helps to understand the relationship between window sizes and complexity of different types of motions for accurate optical flow estimation. The synthetic video frames considered have lesser movement of objects, therefore, a window size of seven pixels seems to perform well. However, the complexity in real images (like the movement of a book or hand across a textured surface), is much more and a low neighborhood size is not able to detect all the optical flow vectors across the moving object. As will be seen further, increasing the window size shows a significant improvement in the optical flow results.

In addition to increasing the optical flow vectors across the moving object, a bigger neighborhood size also helps to reduce the optical flow over the stationary background. This is because of the principle of relative motion on which the optical flow method is based. The analysis of different window sizes for the considered MPEG video frames shows that the neighborhood size should be large enough to capture the relative motion of the pixels in a video frame. On the other hand, using a very large window size for small relative motion may make the algorithm slower without any significant improvement in the results. Therefore, the window size must be chosen by evaluating the complexity of objects and textures in the video frame(s) under consideration.

3.3.2 *Gaussian Pyramid Levels*

Gaussian Pyramids are a hierarchy of low-pass filtered versions of the original image. The successive levels of the image correspond to lower frequencies of the original image [31]. A

Gaussian filter or a Gaussian algorithm is typically used in image processing to smooth a selected image and thus reduce image noise [35]. Thus, the application of Gaussian filtering to video frame/image results is to generate a new corresponding frame/image which has reduced image detail levels. Further, since the Fourier transform of a Gaussian filter is another Gaussian, applying a Gaussian blur has the effect of low pass filtering on the selected image. The obtained levels of Gaussian Pyramids thus do not have any sharp edges, and also does not introduce any ringing into the filtered image [3].

Figure 3.2 depicts the principle of Gaussian Pyramid filtering. With each subsequent level, the image size (area) reduces by one-fourth. In simple terms, one pixel of an $(n+1)^{th}$ level represents 4 pixels of the $(n)^{th}$ level. These Gaussian filtered images, if stacked one on top of the other as shown in Figure 3.2 form a tapering pyramid like structure, hence deriving the name Gaussian Pyramids.

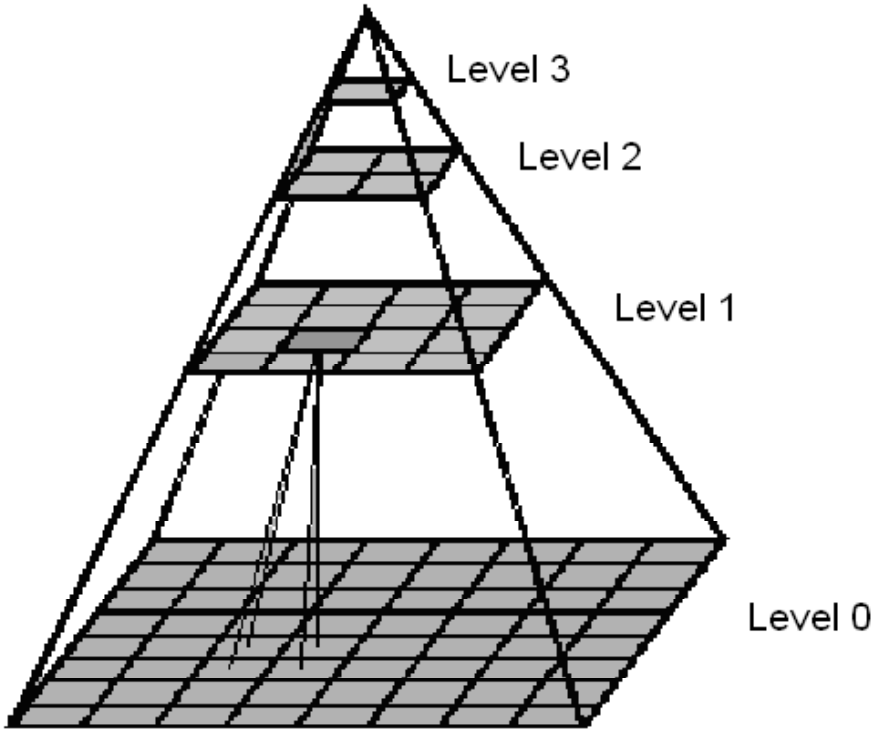

Figure 3.2 Depiction of Gaussian Levels in the form of a Pyramid

In this research, the Gaussian smoothing technique is used in the pre-processing stage to

enhance image structures at different scales for three or four pyramid levels. The image sizes at each pyramid level have been reduced to a quarter size of the previous level (each linear pixel dimension is halved in subsequent cases). The base or main level can be considered as the 0^{th} level. Most of the chosen real image sequences have pixel dimensions of 640×480. The corresponding Gaussian pyramid levels at Level 1 (L1) has the pixel dimensions of 320×240, the Level 2 (L2) has dimensions of 160×120 pixels, the Level 3 (L3) has the pixel dimensions of 80×60 pixels and the Level 4 (L4) at pixel dimensions of 40×30. The synthetic images have pixel dimensions of 200×200. The dimensions of the Gaussian levels have been modified accordingly.

Figure 3.3 shows the method employed to apply the Gaussian pyramid algorithm to the images. The first step is the calculation of the Gaussian filter (or mask) that has to be applied to the images. In this project, a 5-tap Gaussian mask has been applied to each dimension of the image. The matrix for this Gaussian mask can be given as: $[\,0.25 - \frac{a}{2} \quad 0.25 \quad a \quad 0.25 \quad 0.25 - \frac{a}{2}\,]$

The above matrix represents an odd symmetric normalized Gaussian pyramid with sum of weights equal to one. The value of a has been taken to be 0.4 in this research project. This results in the Gaussian pyramid mask of $[\,0.05 \quad 0.25 \quad 0.4 \quad 0.25 \quad 0.05\,]$. This mask is then applied to the image, the image is subsequently padded with two rows and columns of zeros and the image size is resized to half its current size to obtain the first Gaussian pyramid level. This process is continued till the number of desired pyramid levels (as specified in the beginning) is reached.

Depending on the complexity of the video sequence, as well as camera parameter variances (different types of motion as discussed in Section 2.2), a comparative analysis of optical flow estimation has been performed at different levels of Gaussian filtering. In cases where the application of the optical flow algorithm produces reasonably accurate results, only one level of Gaussian pyramids have been applied to show the comparison. The Optical Vector Flow Estimation Algorithm described in the previous section has been applied to each level of obtained Gaussian pyramid. The obtained optical flow vectors have then been superimposed on the original frame, as well as subsequent image levels after Gaussian filtering, clearly indicating motion or moving objects in the frame(s). All these results have been shown in Chapter four.

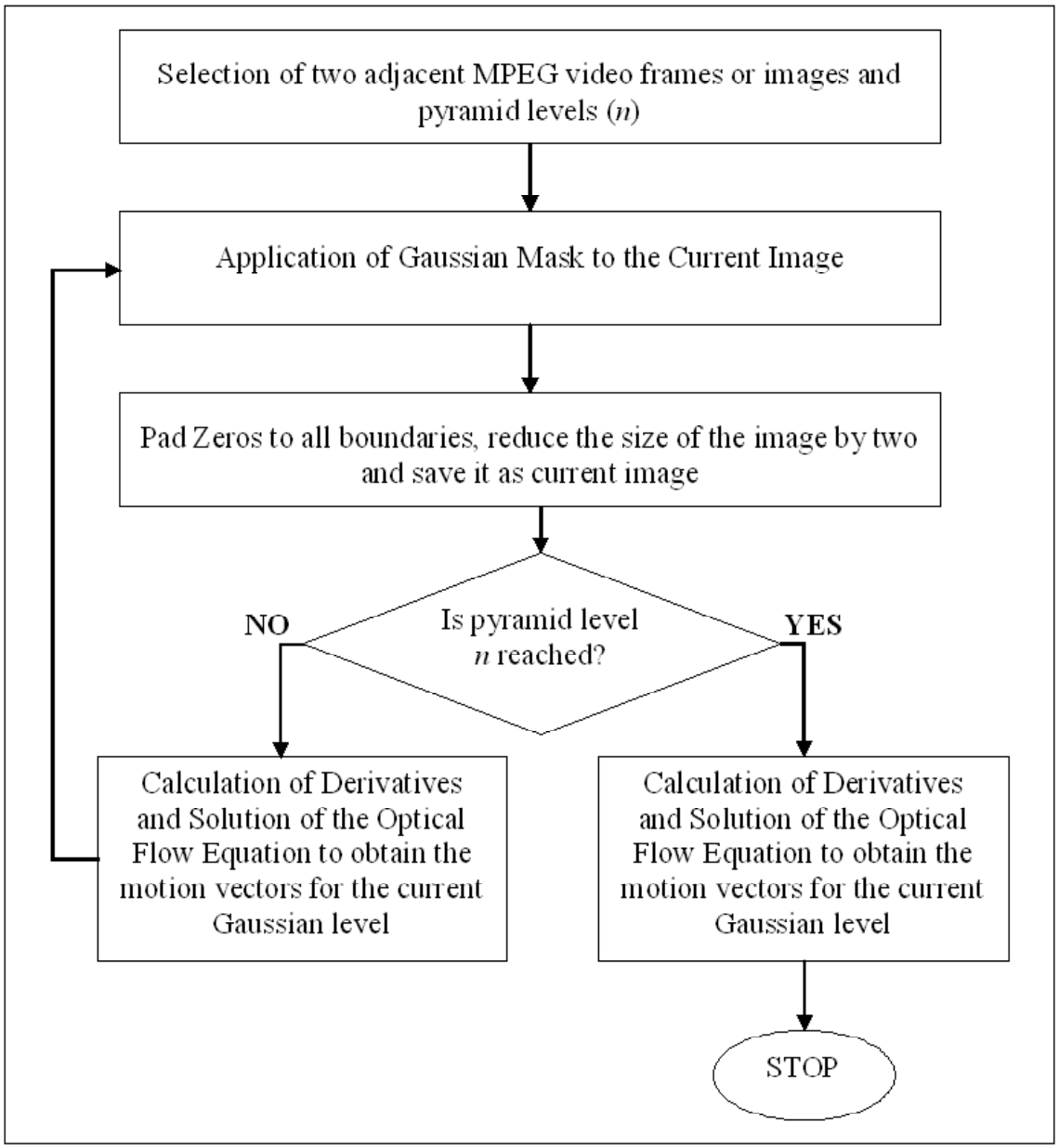

Figure 3.3 Algorithm for the generation of Gaussian Pyramid Levels

3.4 Threshold Segmentation

The threshold technique of segmentation has been discussed in section 2.5.1. As mentioned previously, this technique has proved to perform well for gray level images. Since the optical flow algorithm also uses gray level images, this technique of segmentation has been used in this project. In most motion estimation methods, a threshold for pixel intensity range of $1 - 256$ is

used. However, in the optical flow method, the motion is estimated in the form of the horizontal and vertical vectors u and v respectively. Therefore, the threshold level for each pixel (magnitude only) is mathematically calculated as:

$$Threshold_{Check} = \sqrt{u^2 + v^2} \tag{3.1}$$

The threshold should be set according to the type of video sequence and motion under consideration. In the segmentation results presented in chapter five, the threshold for each sequence has been computed by observing the minimum and maximum values of the horizontal vector u and vertical vector v. Similarly, when a segmentation technique has to be implemented in real life or on online videos, a threshold should be set by first observing a sample data set which broadly resembles the actual video which will be seen by the camera.

3.5 Summary

The optical flow algorithm used for the detection of motion has been described in this chapter. The steps involved in the implementation of the algorithm have been presented. It also addressed the concept of Gaussian pyramid filtering that has been inculcated in this project and the method for the same has been described. The method of using a threshold for segmentation of images has also been discussed.

4. IMPLEMENTATION OF OPTICAL FLOW METHOD FOR MOTION DETECTION

4.1 Introduction

The optical flow algorithm used in this project and the application of Gaussian filtering was discussed in the previous chapter. This chapter presents the results obtained by the application of the algorithm for different types of motions. Various synthetic and real image sequences have been considered and the optical flow vector results have been shown for each set.

4.2 Application of Optical Flow to Video Sequences

The optical flow algorithm has been tested for different types of data sets, representing different types of motions. The results obtained by application of the Gaussian pyramid filtering have also been presented in this chapter. The number of Gaussian pyramid levels considered depend on the complexity of the selected image sequences. Complexity is defined as the number of simulatenously moving objects and the textured/patterned surfaces present therein. This implies that the number of Gaussian pyramid levels chosen is higher for a more complex image. This is described in further detail in the subsequent sections.

Each of these MPEG video sequences considered in this project have been converted into video frames first, which are further converted into images for the application of the Optical Flow Algorithm using Image Processing Toolbox of MATLAB. Once Optical Flow vectors for each set are obtained, these images can be converted back to a video sequence using a reverse method and can be played back as a video sequence. The video sequence can be obtained in the form of either optical flow vectors alone or optical flow vectors superimposed on the original video sequence.

4.3 Results of Video Sequences using Optical Flow Algorithm

Different sets of synthetic and real images have been considered in this section. The Optical Flow Algorithm presented in Chapter 3 has been applied to each of these sets and the results are presented in this section. The results for each of these sets show a standalone optical flow vector graph and optical flow vectors superimposed on the original image for different Gaussian pyramid levels.

A total of nine (9) sets of images have been considered. This includes two (2) synthetic image sequences and seven (7) real image sequences. Synthethic images are image sequences that are visually simulated to achieve or represent a certain type of motion or complexity. Real images are image sequences captured to depict motion in real life including effects of lighting changes.

The two sets of synthetic images have different levels of complexities. A brief description of these image sequences has been given below.

1. Sphere

   This image sequence consists of a textured sphere rotating around its own axis. The complexity level for this image sequence is relatively low because of only one rotating object (with no occlusion) against a non-changing background. Therefore, only one set of result has been shown for this image sequence. Further, the shape and direction of optical flow vectors of the rotating sphere have been shown in a magnified format.

2. Sphere and Cube Combination

   The complexity level of this image sequence is higher as compared to the first image set. This image set includes a combination of a textured sphere (rotating around its own axis) and a plain blue colored cube (rotating independently, also around its own axis) against a non-changing background. In this sequence the sphere and the cube are independently rotating and moving towards each other, thus resulting in an occlusion in the later frames of the video sequence. It is expected that the results of the optical flow algorithm should not be affected by this occlusion. The two sets of results show optical flow vectors for movement with and without occlusion.

The description of the seven sets of real image sequences has been given below. The optical flow algorithm has been tested for different types of motions in these image sequences.

3. Human Hand Movement in front of a Textured Background

   This real image sequence captured by a video camera shows the downward motion of a human hand against a textured background. The complexity of this image sequence arises from the shape of the hand and the fingers. The optical flow algorithm is expected to identify the shape of the moving human hand. Three sets of results have been considered in this case. These three sets show different positions of the hand as the hand moves downwards.

4. Human Hand and Book Movement in front of a Textured Background

   The complexity of this image sequence becomes higher by the addition of a book being held by a human hand. In this case, the optical flow algorithm should be able to detect the shape and diagonal downward motion of the hand as well as the textured book. Three sets of images showing different positions of the hand and book have been considered.

5. Mug Rotation in front of a Textured Background

   This image sequence is a combination of hand (as described in 3) as well as a white labeled mug movement. Instead of a diagonal downward movement, the mug is being rotated in the hand. Since the mug is relatively non-textured, the response of the optical flow algorithm is observed and tested in this case. Three different rotating mug positions have been considered in this case.

6. A Baby Raising his Hand

   This image sequence captures a baby moving his arm in the upward direction. This video sequence has some camera movement (resulting in a slight blurring effect), thus representing a combination of pronounced arm motion and slight background motion (due to camera movement). The optical flow algorithm should depict smaller magnitude vectors for the background as compared to the moving arm vectors. One set of arm movement has been shown in this case.

7. Rotation of a Flower Basket Arrangement

   This image sequence shows a non-textured flower basket containing flowers rotating around a perpendicular axis (passing through the point of suspension of the basket). This set has been tested for movement of simultaneously rotating objects. Rotary motion of the non-textured flower basket has been combined with pan-like motion of the flower arrangement. The background in the image sequences remains stationary. One set of images has been considered. The optical flow algorithm should be able to detect the motion of the flowers, non-textured basket and the strings holding the basket.

8. Moving Traffic representing Zoom-in Motion

   This image sequence has been captured from within a moving vehicle. As the traffic is coming towards the vehicle from the opposite direction, the objects move closer and appear bigger in size. This represents zoom-in motion equivalent to a camera zooming in on a stationary object. Two different sets of images have been considered. A close-up of the obtained optical flow vectors has been included to study the direction of these vectors for a typical zoom-in motion.

9. Camera Movement in a Scene

   This last set shows an image sequence when a camera moves instead of an object in an image. In this type of motion, the whole scene changes with each frame. Therefore, the optical flow vectors should be spread in the entire image frame. In this particular frame, the camera captures flower basket arrangements on a table. The camera moves in a diagonal motion and the optical flow vectors should be able to detect this direction. Two different sets of images have been considered in this case.

4.3.1 *Results using Synthetic Video Sequences*

This section presents the results for two (2) video sequences of synthetic images. It should be noted that in all the presented results, the two chosen frames in the first row are the images taken from the video sequence. The results in rows second (or below) are the results generated by the Lucas Kanade algorithm used in this thesis.

Sphere

This set of images [33] [15] shows a textured sphere rotating on an imaginary axis passing through the centre. The texture on the sphere thus changes with each subsequent image. The first set used to test this data set of images comprises of frames one (1) and two (2) and the results are shown in Figure 4.1. All the other sets of images considered in this image sequence produced similar results.

It can be seen that the Optical Flow vectors obtained are concentrated only inside the circumference and over the surface of the whole sphere. The Optical Flow vectors shown in all the figures have been superimposed on the original image. Even though the circumference of the sphere remains constant, the optical vector flow results obtained are as expected. This is due to the fact that the textured surface inside the sphere changes with each image and the optical flow algorithm is able to detect that change.

The Optical Flow algorithm, on the other hand, is based on relative motion between pixels of an image; therefore, even a small movement between pixels is enough to detect motion. This data set is a relatively simple video sequence, therefore, only the base level of Gaussian pyramid has been considered. Furthermore, the results obtained at the base level are quite accurate, therefore, further Gaussian pyramids have not been examined.

Figure 4.2 shows a magnified view of the optical flow vectors obtained for this image sequence. It can be observed that the direction of these vectors indicate the movement of the pixels of the sphere. The vectors close to the circumference or the edge of the sphere have a horizontal as well as vertical component, whereas the vectors in the centre of the sphere are relatively horizontal. The rotation of the sphere translates into a pan-like motion, especially in the centre of the

sphere. Consequently, the optical flow vectors indicate this pan-like motion. This image dataset also proves that the optical flow algorithm works very well for textured surfaces.

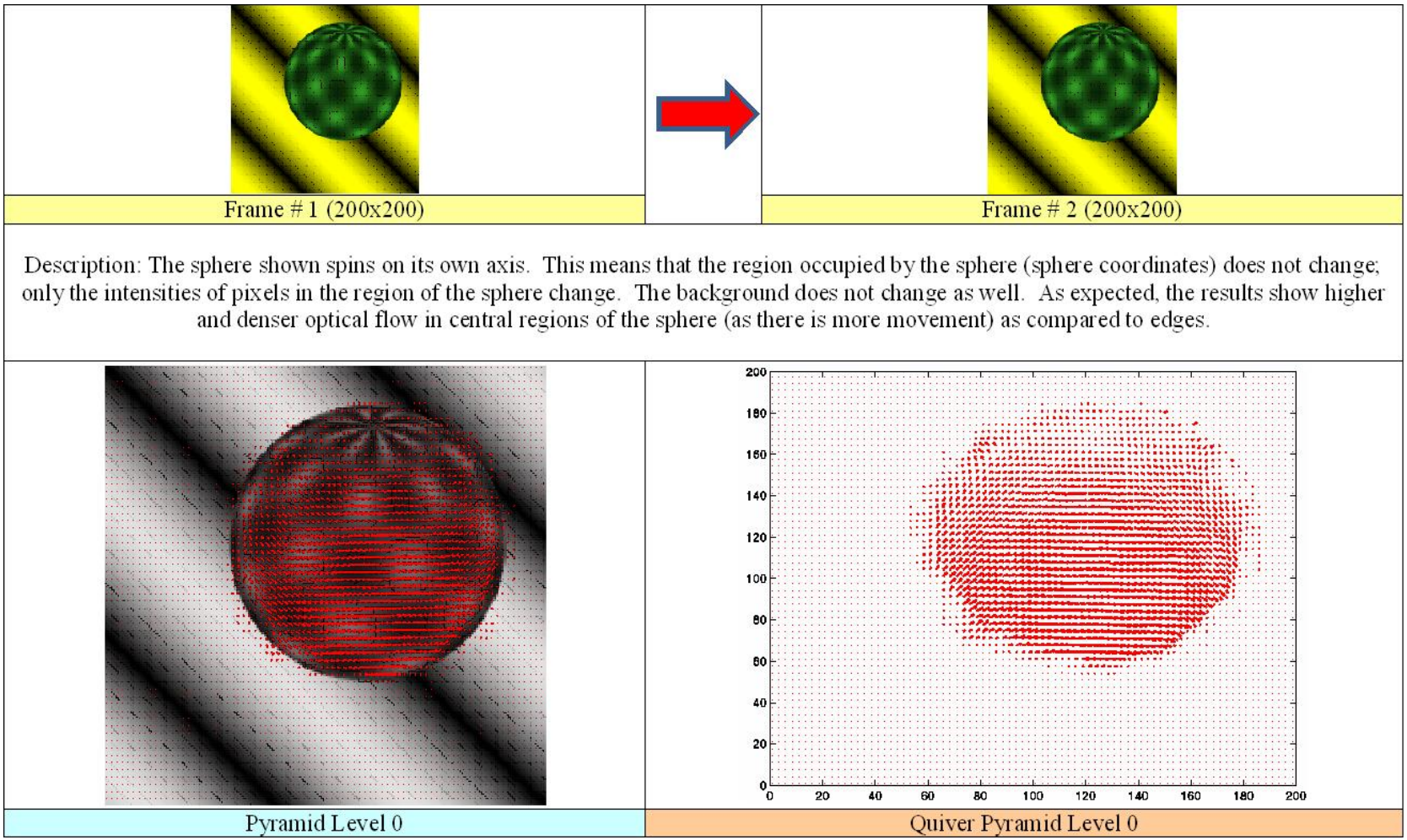

Figure 4.1 Optical Flow Results for Synthetic Image of a Sphere: Frames 1 and 2

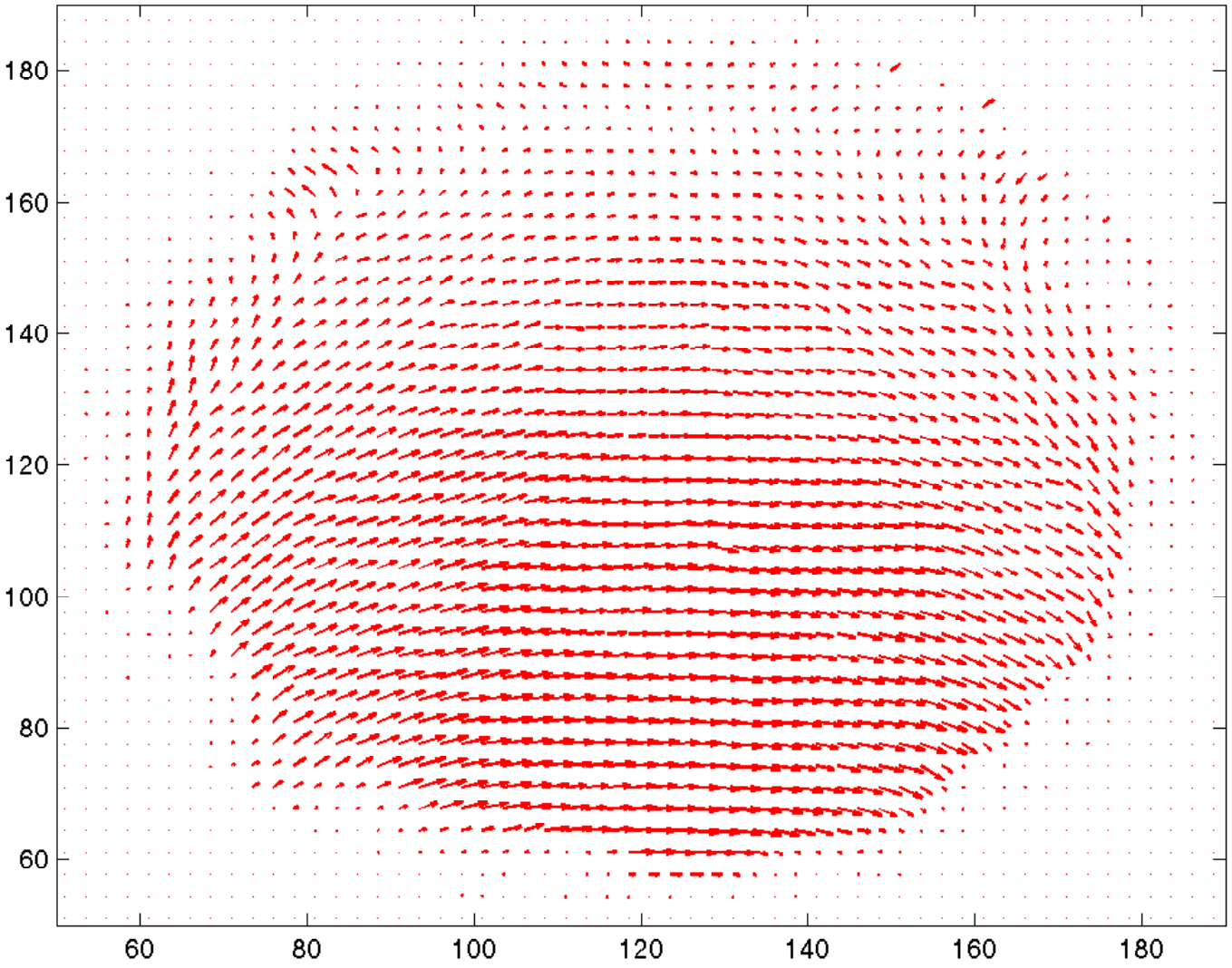

Figure 4.2 Magnified Optical Flow Results for Synthetic Image of a Sphere: Frames 1 and 2

Sphere and Cube Combination

This image sequence [33] is slightly more complicated compared to the previous image sequence. An independently rotating cube has also been added to the rotating sphere shown in the previous image sequence. The results obtained by applying the Optical Flow Algorithm have been shown in Figures 4.3 and 4.4. It may be seen that the sphere and cube are rotating independently along an axis perpendicular to the surface of this page. The movement of the sphere and cube towards each other results in a small occlusion in the second set of images, as seen in Figure 4.4. The obtained results show that the optical flow algorithm is able to accurately detect the motion along the circumference of both the objects, clearly indicating motion.

The base Gaussian pyramid level does not show significant motion inside the cube. Further, the occlusion of the sphere and cube adds to the complexity of the images and prevents the optical flow algorithm to accurately detect the motion. Therefore, two Gaussian pyramid levels have been applied and the results obtained are shown in the figures.

There are two main aspects that come forward by observing these results. Firstly, as can be observed, the rotating cube in this case is a non-textured surface. In the base level, the optical flow algorithm does not detect any motion inside the cube. This implies that the optical flow algorithm may not be able to recognize relative motion when applied to smooth surfaces. In this scenario, the Gaussian pyramid filtering improves the results with each progressing level. This is because of decrease in resolution of the smooth surface inside the cube with the generation of each gaussian pyramid level.

The second observation is that the optical flow algorithm does not get affected by the occlusion between the sphere and the cube. When the occlusion occurs as shown in Figure 4.4, the optical flow algorithm is able to detect the object which is in front (sphere). Another factor that may be contributing to the accuracy of optical flow algorithm in this case might be the occlusion of a non-textured cube by a textured sphere.

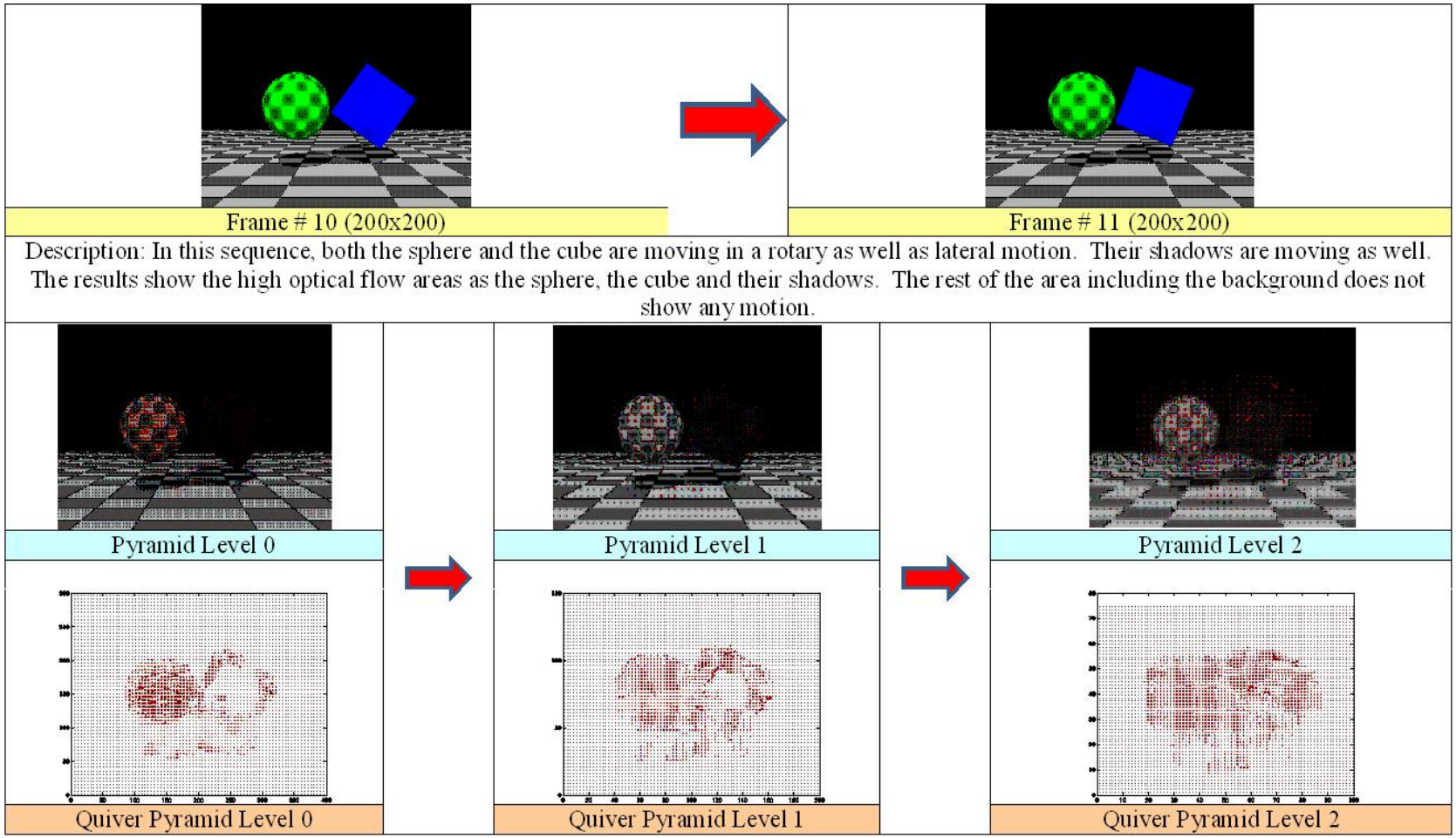

Figure 4.3 Optical Flow Results for Synthetic Image of a Sphere and Cube rotating independently: Frames 10 and 11

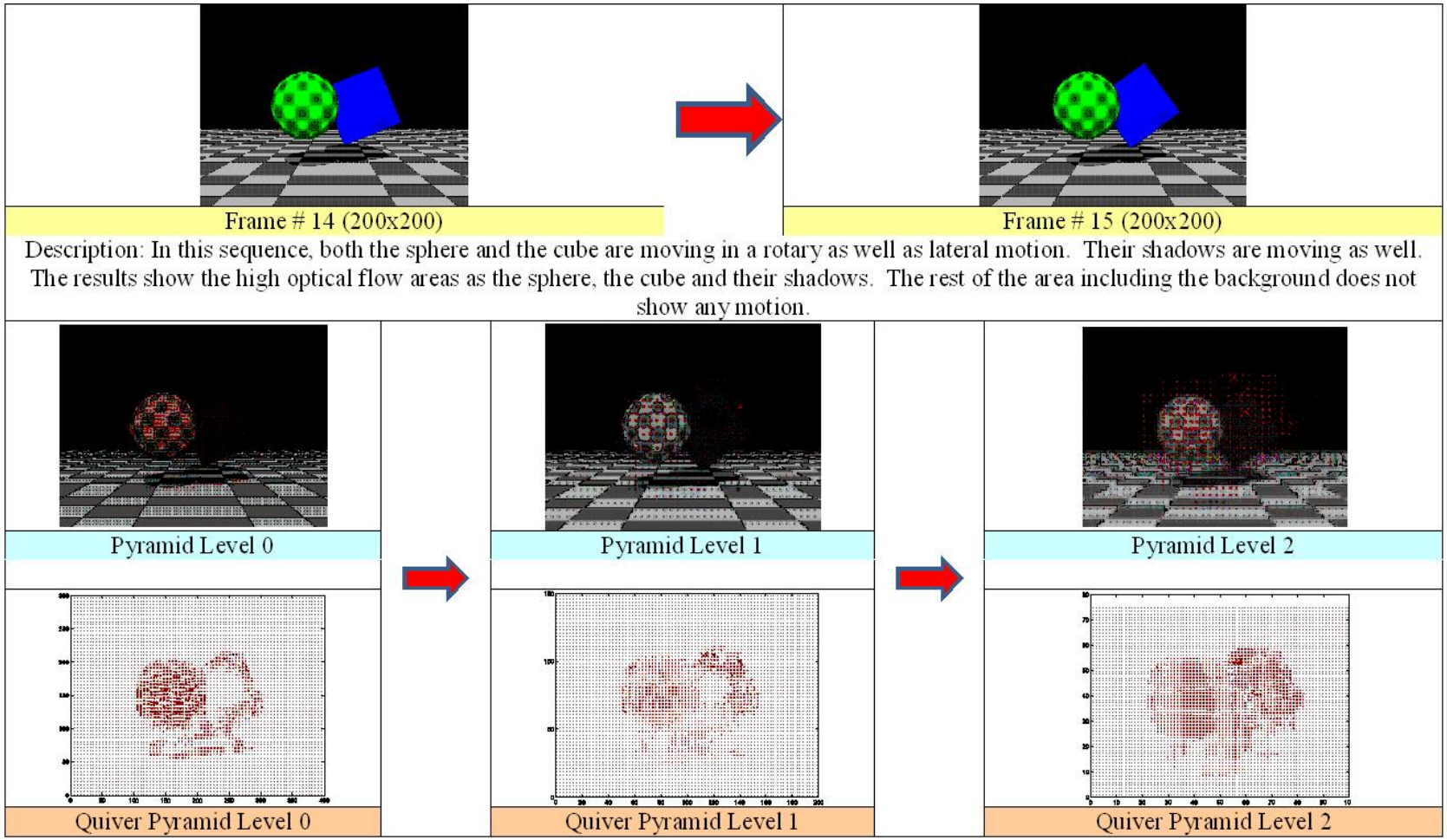

Figure 4.4 Optical Flow Results for Synthetic Image of a Sphere and Cube rotating independently: Frames 14 and 15

4.3.2 *Results using Real Video Sequences*

The optical flow algorithm produced good results for the two synthetic image sequences. The optical flow algorithm has now been tested for real-life movement. The algorithm should be able to respond equally well to real image sequences. The purpose of testing the optical flow algorithm for different type of images and motions is to ensure that the algorithm is reliable. Further, if the algorithm is able to correctly detect motion in these cases, it can be deemed as robust and accurate.

This section presents the results for seven (7) different types of video sequences of real images. The description of each video sequence has been given in the following individual sections. Similar to the previous section, results in rows two (or below) are the results generated by the Lucas Kanade algorithm.

Human Hand Movement in front of a Textured Background

In this video sequence [29], a human hand moves across a textured background of books. The hand moves into the scene from the top right hand corner moving downwards across the frames. This results in a combination of pan and tilt motion. Three (3) different sets of results clearly showing the movement of the hand have been presented for this set. These are presented in Figures 4.5 to 4.7. The results clearly show the concentration of the optical flow vectors over the moving hand. The results also show two levels of Gaussian Pyramids that have been used for this data set. It can be observed that with each pyramid level, the profile of the moving hand becomes clearer and concentrated in comparison to the stationary background.

As was mentioned previously, the optical flow algorithm has been tested for this image sequence to detect the motion of the hand and the fingers. The shape of the hand and the fingers can be clearly seen in all three cases at different positions in the frame. It can be observed that as the Gaussian pyramid filtering decreases the resolution of the image, the shape of the hand (and fingers) gets slightly deformed. However, the optical flow vectors increase in magnitude and become denser in comparison to the rest of the stationary background.

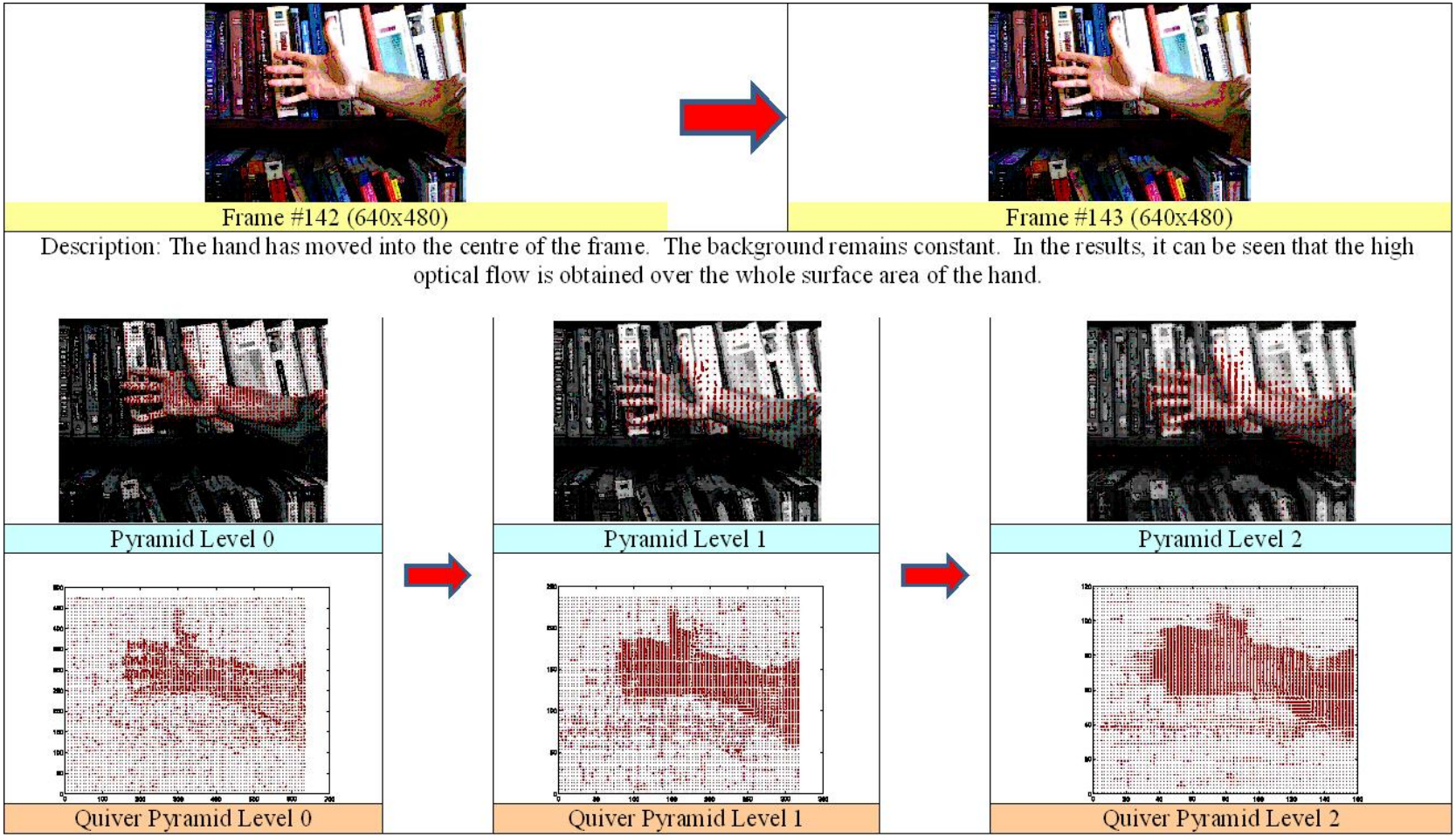

Figure 4.5 Optical Flow Results for Real Image of a Hand moving over a highly textured background: Frames 142 and 143

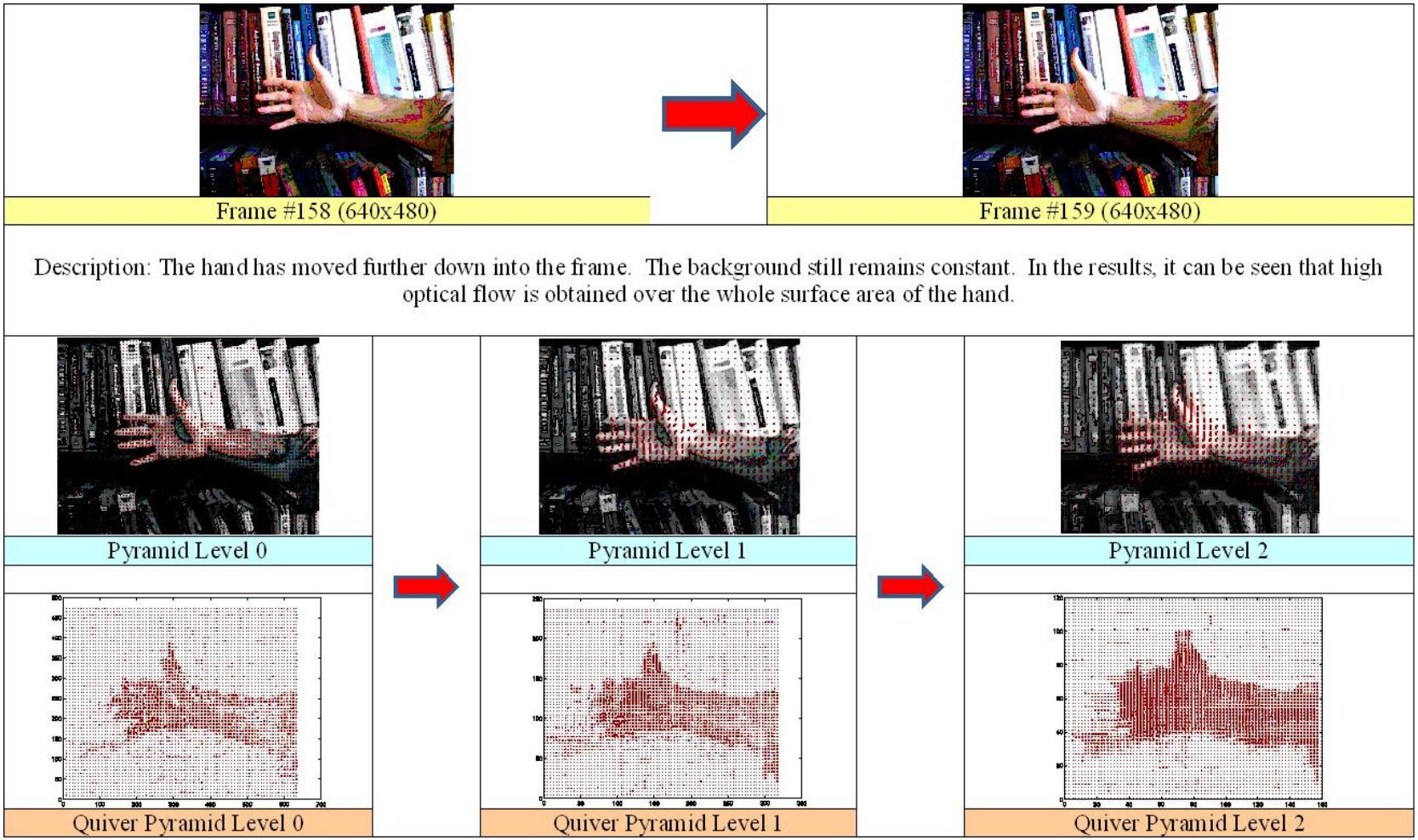

Figure 4.6 Optical Flow Results for Real Image of a Hand moving over a highly textured background: Frames 158 and 159

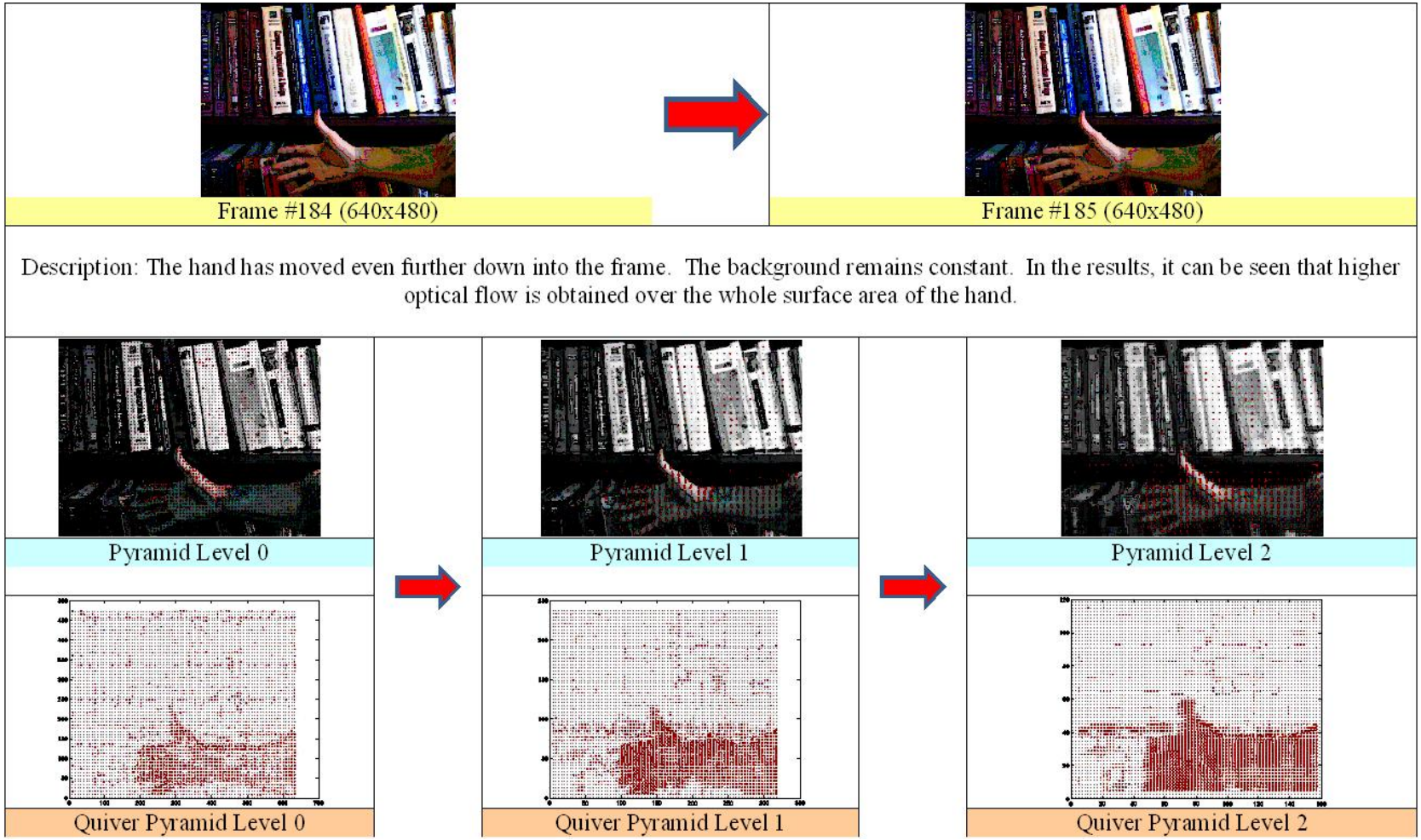

Figure 4.7 Optical Flow Results for Real Image of a Hand moving over a highly textured background: Frames 184 and 185

Human Hand and Book Movement in front of a Textured Background

In this image sequence [29], a highly textured book can be seen in addition to a moving hand, making it more complex than the previous image sequence. The book is being held in the human hand and is moving in front of the textured book shelf. Three levels of Gaussian pyramids have been considered for this particular image sequence. Figures 4.8 to 4.10 show three (3) sets of results with three Gaussian pyramid levels each.

The hand in this video sequence moves the book from the top right hand corner to the bottom left hand corner, resulting in a diagonal motion. The optical flow results show significant movement of the hand and the book. The moving hand and the book have been clearly distinguished in the results. The shape of the hand holding the book can be clearly seen and the optical flow vectors become more concentrated with each progressive Gaussian pyramid level. The results show a significant improvement by the application of Gaussian filtering as the contours of the moving objects become more distinguishable with increase in the pyramid level.

Inspite of this image being complex due to the presence of the highly textured background, textured book and relatively lesser textured hand, the movement of the hand and book has been correctly estimated. The results obtained from this image sequence clearly suggest that the optical flow algorithm works well for textured surfaces even in the presence of multiple moving objects.

Another scenario worth considering is using a non-textured book or object instead of a textured book. For example, using a uniformly colored book, instead of a textured one as used in this case will present a more challenging case. A similar case has been undertaken in the next image sequence. The response of the optical flow algorithm when subjected to the movement of two similarly textured objects (i.e., non-textured book and non-textured hand) will determine its adaptability.

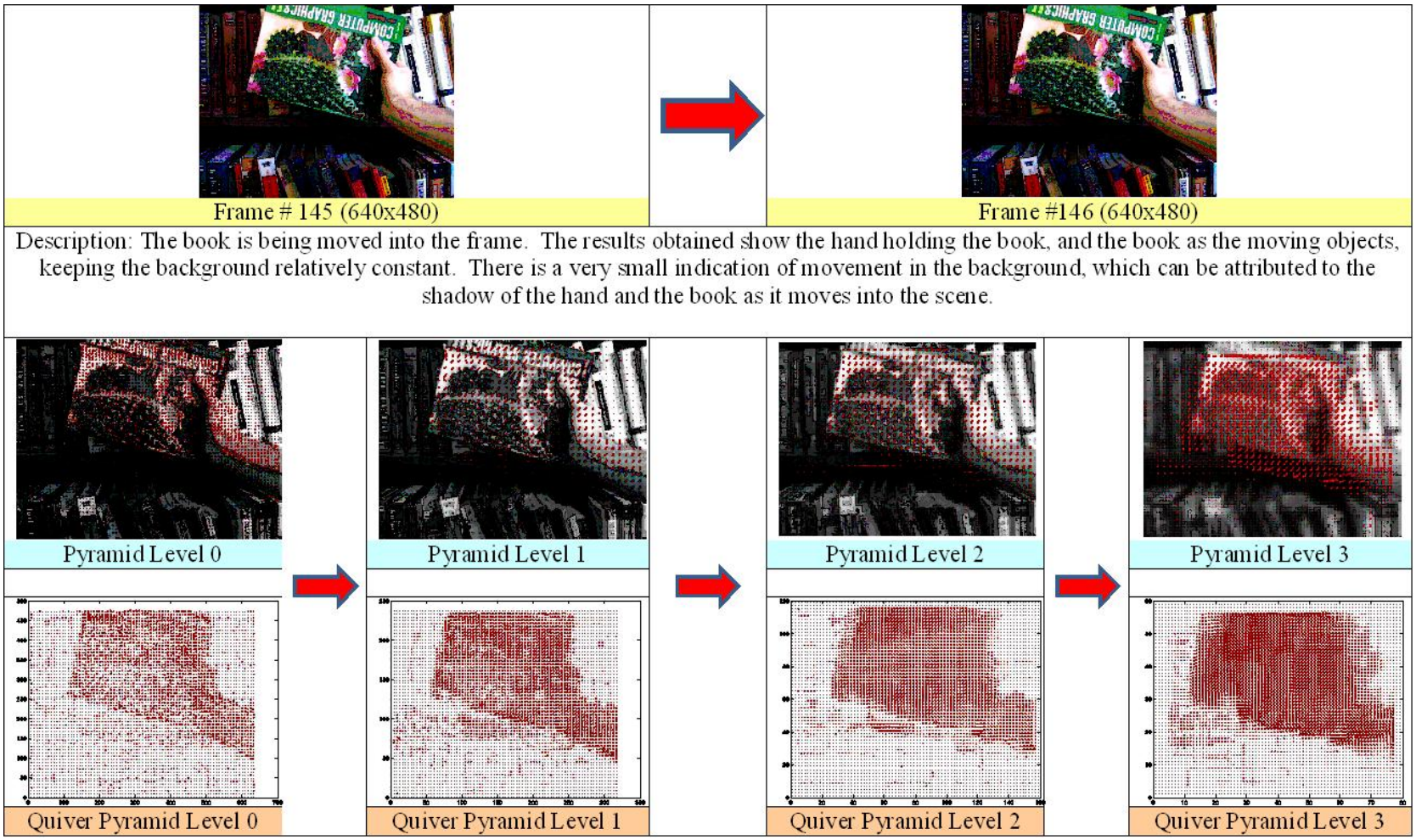

Figure 4.8 Optical Flow Results for Real Image of a Hand carrying a textured book and moving over a highly textured background: Frames 145 and 146

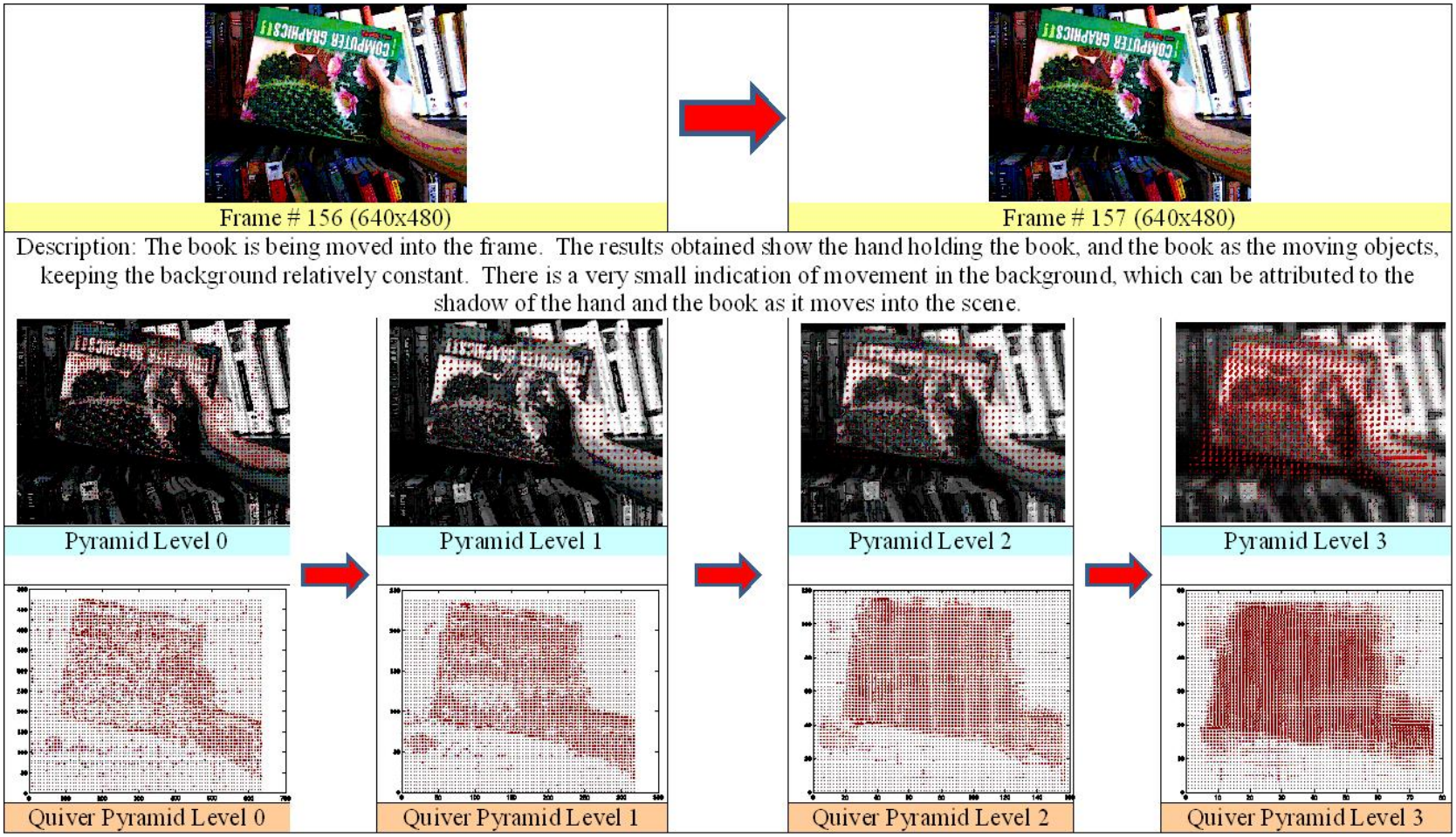

Figure 4.9 Optical Flow Results for Real Image of a Hand carrying a textured book and moving over a highly textured background: Frames 156 and 157

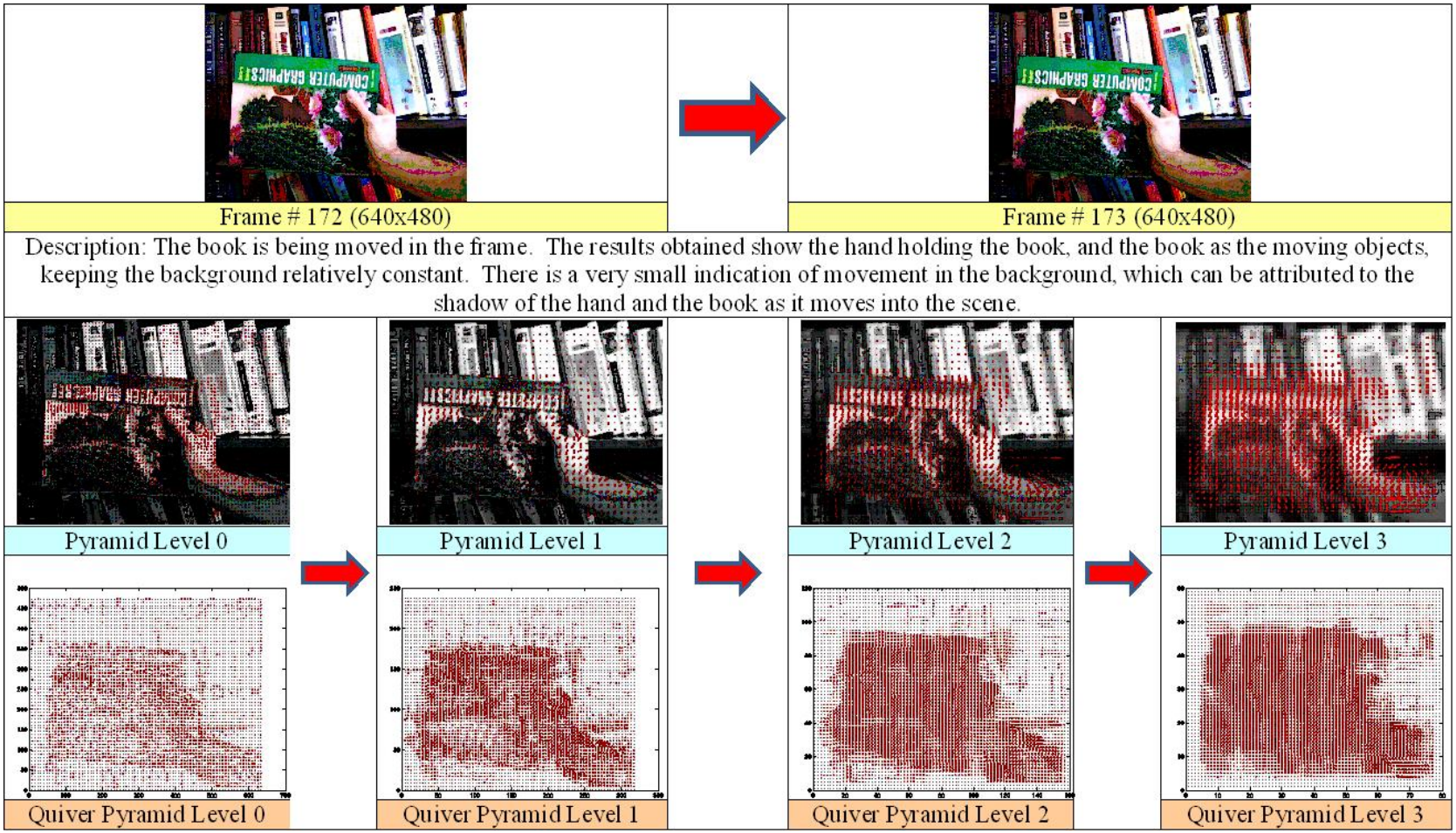

Figure 4.10 Optical Flow Results for Real Image of a Hand carrying a textured book and moving over a highly textured background: Frames 172 and 173

Mug Rotation in front of a Textured Background

The application of the optical flow algorithm to a combination of two non-textured surfaces has been considered in this image sequence. This set of images [29] comprises of a non-textured mug which is being rotated by a human hand. The hand in these images remains almost constant with the mug being rotated back and forth. The mug has a uniform white color which acts as a non-textured surface, barring the slight change in pixel color intesity levels due to gradual lighting and shading changes. It has been observed that with each Gaussian pyramid level, the detection of motion for this non-textured surface becomes challenging. As Gaussian pyramid filtering is applied, the image size becomes smaller, thus decreasing the resolution of the image. The texture of the mug becomes further smoothened with each level of Gaussian filtering. This makes the estimation of optical flow vectors difficult due to a decrease in relative motion between pixels.

This image sequence represents rotary motion and the results are shown in Figures 4.11 to 4.13. The base and the first Gaussian pyramid level are able to detect motion correctly. The results are able to clearly show the structure of the hand and the mug in the images. The results seem to deteriorate after the first level. These results show that Gaussian filtering is a good way to smoothen out highly textured surfaces before the application of optical flow algorithm. However, for non-textured or smooth surfaces Gaussian pyramid filtering can be eliminated as the optical flow algorithm performs better at the base level.

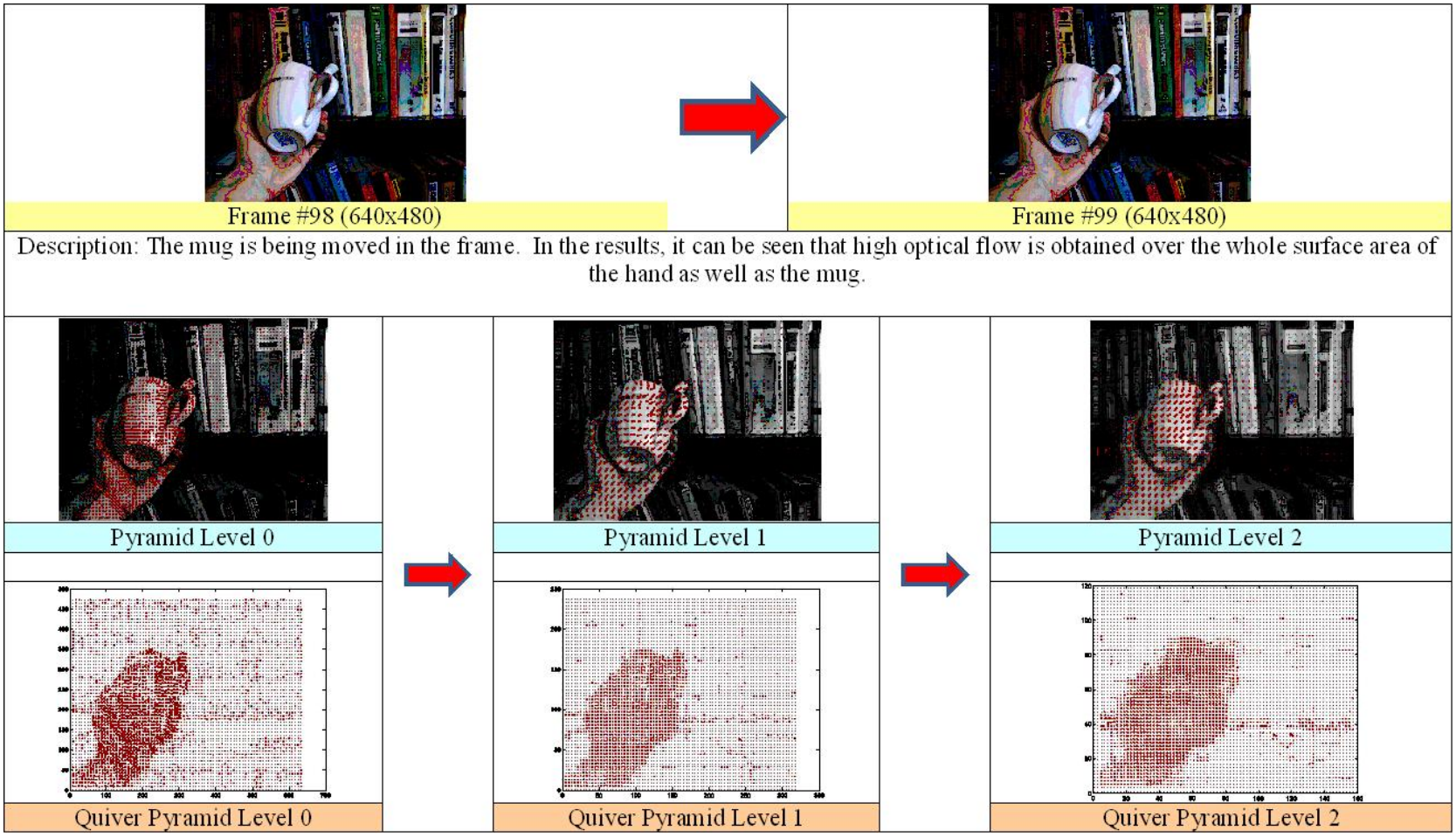

Figure 4.11 Optical Flow Results for Real Image of a Mug and Hand and moving over a highly textured background: Frames 98 and 99

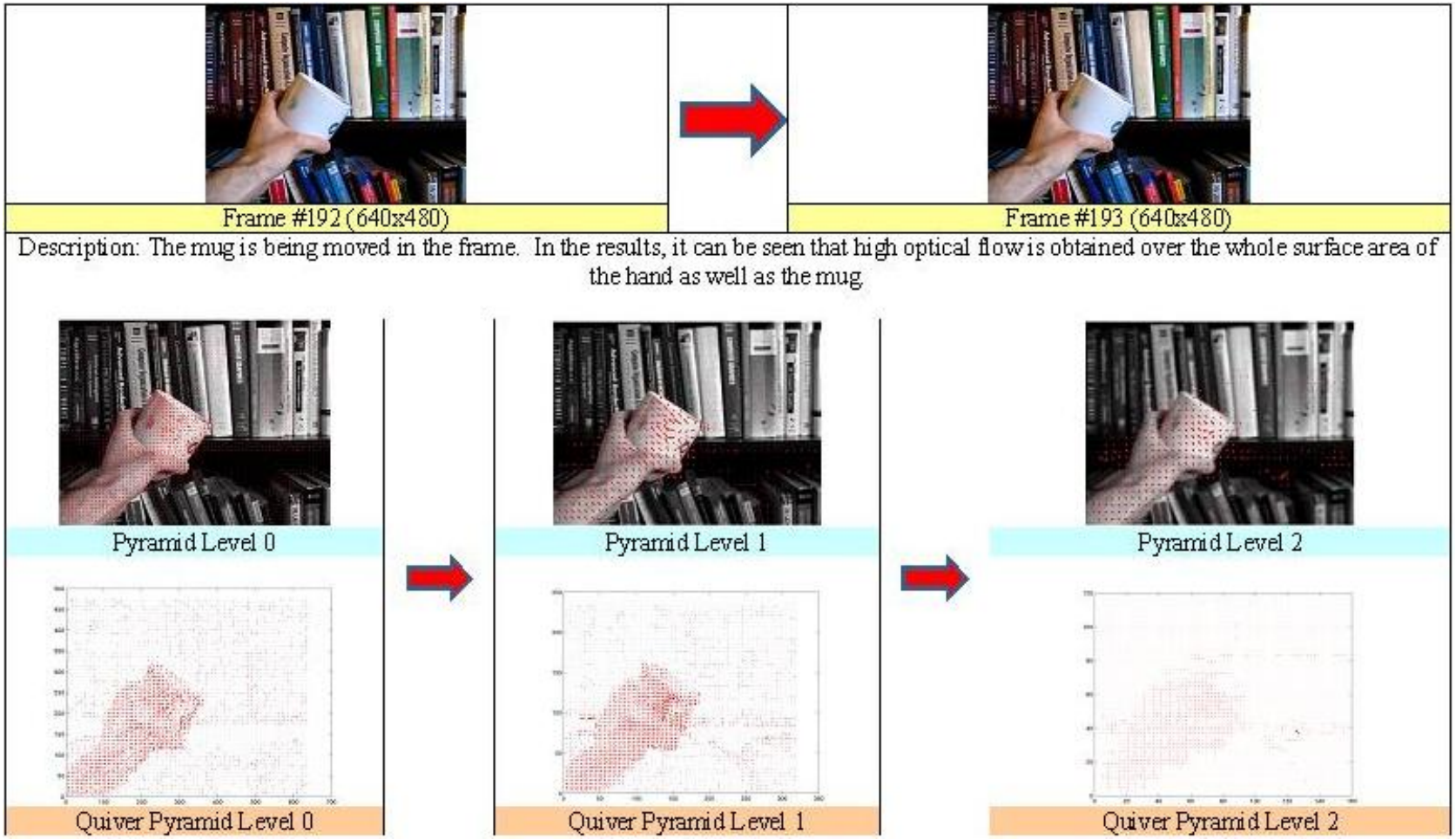

Figure 4.12 Optical Flow Results for Real Image of a Mug and Hand and moving over a highly textured background: Frames 192 and 193

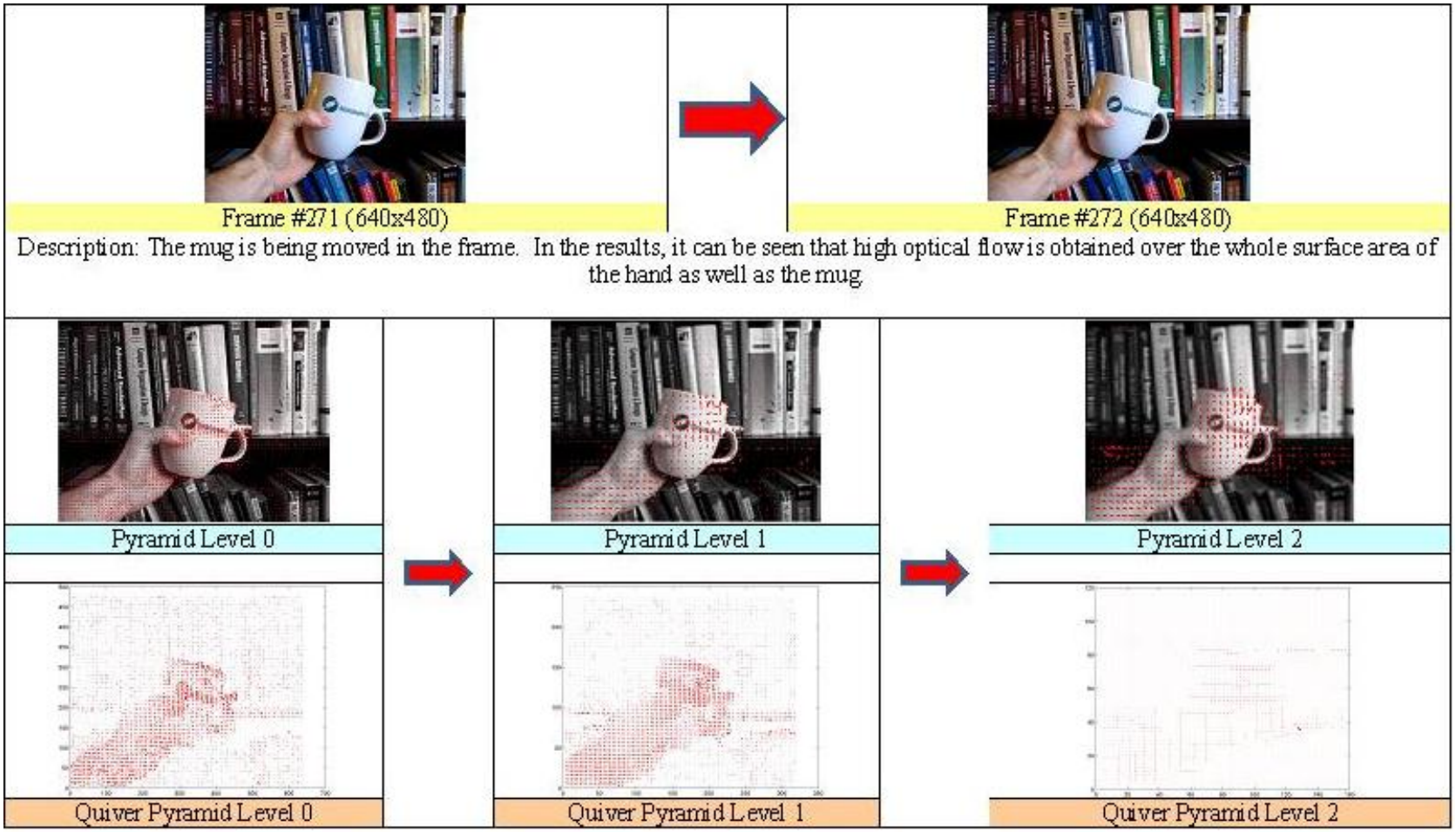

Figure 4.13 Optical Flow Results for Real Image of a Mug and Hand and moving over a highly textured background: Frames 271 and 272

A Baby Raising his Hand

This video sequence [17] shows a baby moving his hand in the upwards direction (similar to a pivoting motion about the elbow) to put food in the mouth. Frames 4 and 5 of the video sequence are shown in Figure 4.14. As was briefly mentioned before, this video sequence has some amount of camera motion, which causes a slight blurring effect. This blurring appears as background motion. Therefore, the motive for testing this video sequence is to be able to detect the baby's arm movement in the presence of external parameters such as slight camera/ background motion.

The application of the optical flow algorithm shows the concentration of the resultant motion vectors over the baby's arm. However, a few large motion vectors can be seen in the stationary background as well. This is as a result of lighting change in the background (arising due to slight camera movement). As the Gaussian pyramid filtering is applied, it can be seen that the results improve significantly. The results obtained for the second Gaussian pyramid level clearly separate the moving arm from the stationary background. This implies that Gaussian pyramid filtering is able to smoothen out the texture present in the image which helps to differentiate the moving objects even in the presence of camera movement. As the image resolution becomes lower and the image becomes smaller in size, the slight camera movement (and hence lighting changes) does not significantly affect the detection of moving objects (as in the case of base level Gaussian pyramid).

This result suggests that Gaussian pyramid filtering can be effective in video surveillance applications where lighting changes or other similar external parameters may be an issue to get accurate motion estimation results.

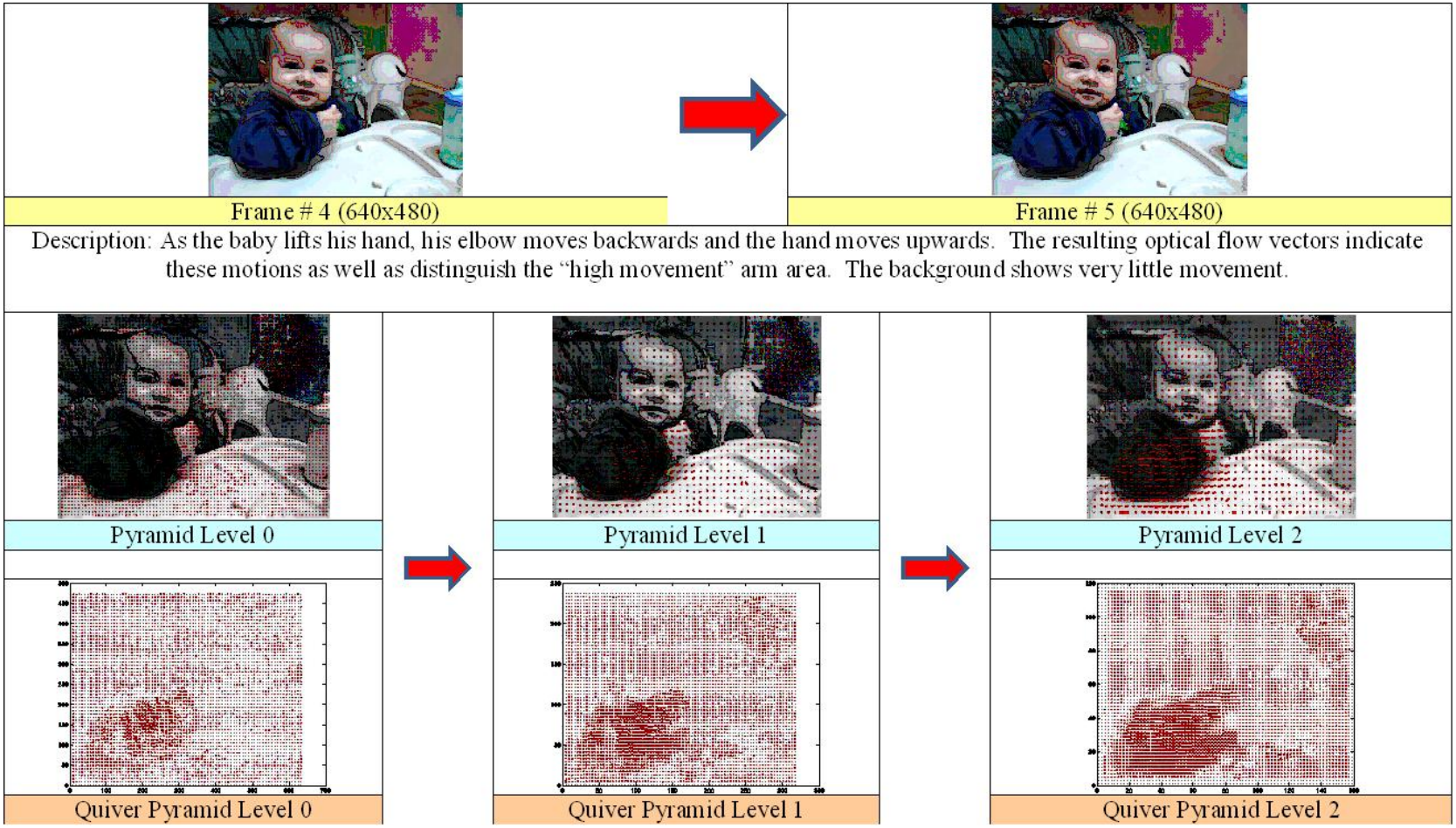

Figure 4.14 Optical Flow Results for Real Image of a Baby raising his hand (in the upward direction) to put food in the mouth: Frames 4 and 5

Rotation of a Flower Basket Arrangement

This image sequence is an example of movement of a non-textured object. The optical flow algorithm has been tested for its sensitivity towards a non-textured object with enough relative movement between frames.

The flower basket in this set of images is rotating around a vertical axis [17]. Figure 4.15 shows the results obtained for this type of movement between frames 8 and 12. In this case, the base level results for optical flow are not very accurate and do not clearly distinguish the moving objects from the background. The concentration of the vectors improves with each Gaussian pyramid level. The optical flow results for first, second and third pyramid levels clearly show the concentration of vectors over the flowers, the basket in which they are placed as well as the string holding the basket.

An important observation that can be made from these results is that even though the flower basket is a smooth non-textured surface, the optical flow algorithm detects the motion correctly. This implies that the optical flow algorithm has the ability to estimate the motion of smooth monotonous surfaces if there are shading or lighting changes. Therefore it is important that all such factors be considered before choosing a motion estimation algorithm for any particular application.

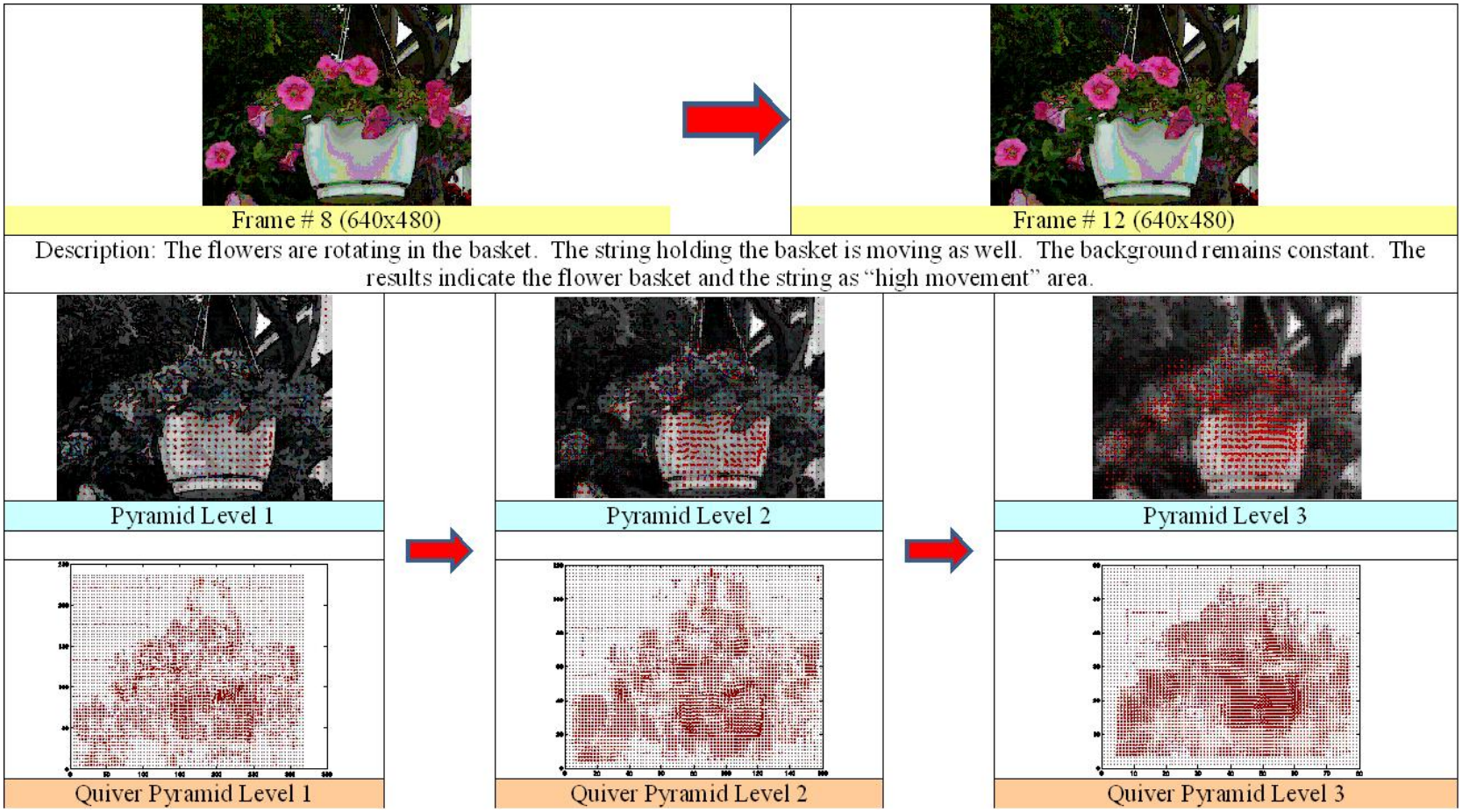

Figure 4.15 Optical Flow Results for Real Image of a flower basket rotating around a vertical axis: Frames 8 and 12

Moving Traffic representing Zoom-in Motion

This video sequence [29] captures moving traffic and has been taken from within a vehicle. The moving traffic is coming closer to the principle vehicle from the opposite side of the road and eventually passes by from the opposite direction. This motion appears like a zoom-in motion because objects become bigger with each passing image. The optical flow algorithm has been tested to detect this type of motion. The expected results are optical flow vectors moving outwards from the centre of the image towards all the four edges and four corners of the image.

The results accurately show the optical flow vectors all over the image, with concentration along the edges of the image. Figures 4.16 and 4.17 show two different image sets from this video sequence. The results from first, second and third levels of Gaussian filtering are shown. Another interesting feature to notice in these results is the estimation of a "V" like structure which is present on the road. This "V" can be seen in the obtained optical flow results.

Figure 4.18 shows a magnified view of the obtained optical flow vectors. It can be seen that the vectors start from the centre of the image and move towards the edges and corners, correctly representing the movement. The results prove that the optical flow algorithm also works well for zoom-type camera movement. The Gaussian pyramid levels show denser optical flow vectors, implying that Guassian pyramid filtering improves the estimation of motion in a complex video sequence such as a zoom-in or zoom-out.

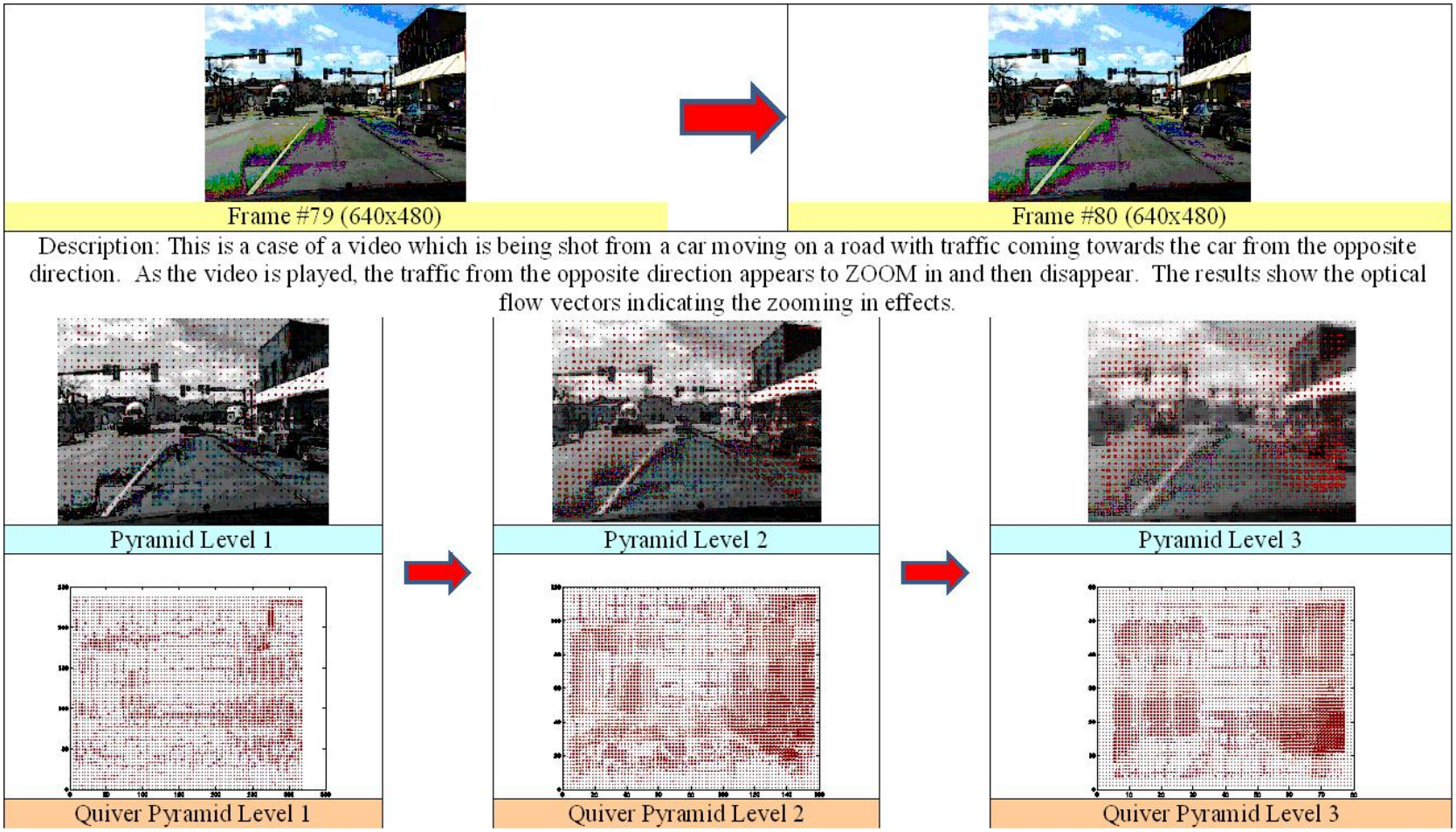

Figure 4.16 Optical Flow Results for Real Image capturing moving traffic as taken from within a vehicle: Frames 79 and 80

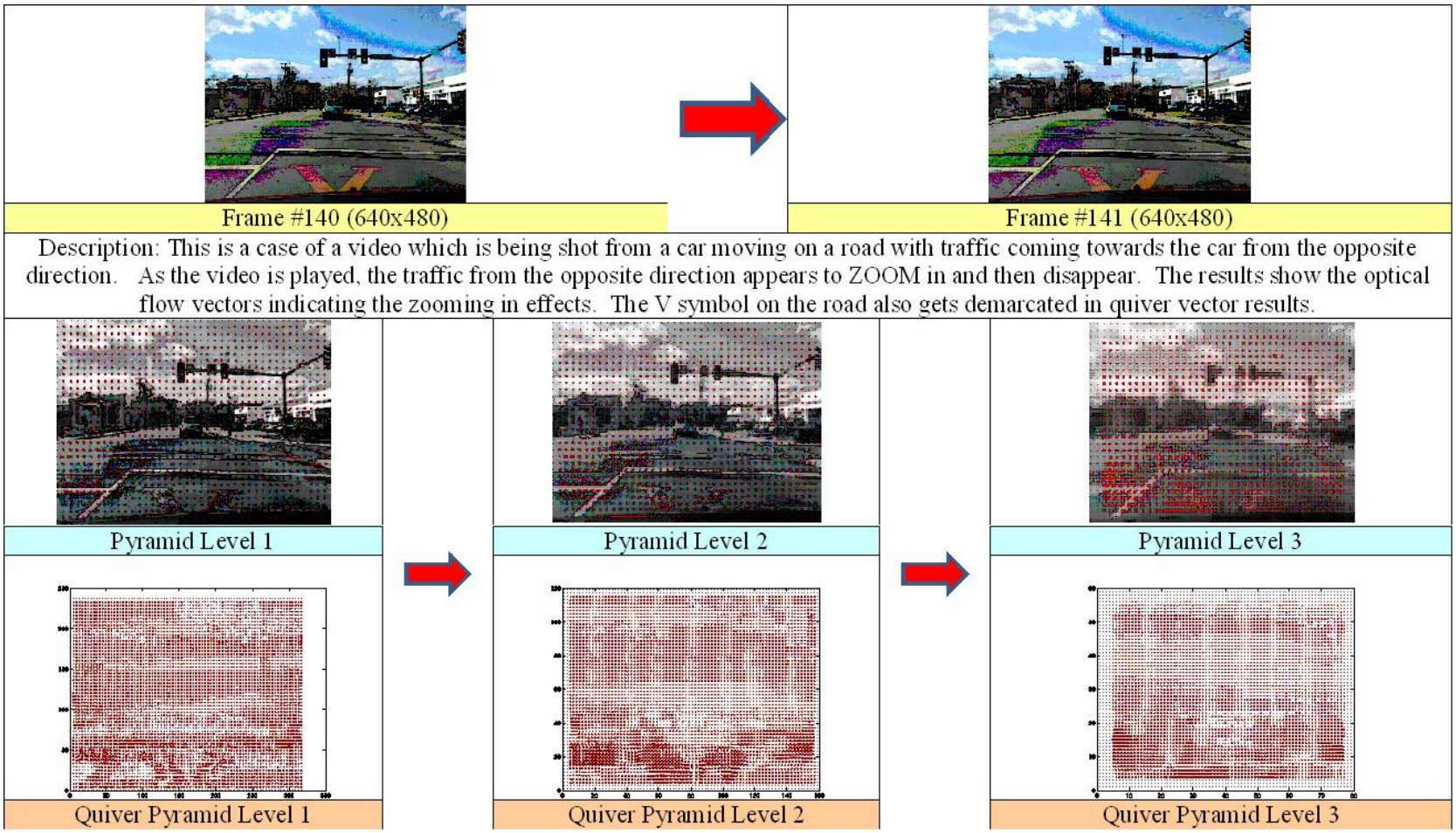

Figure 4.17 Optical Flow Results for Real Image capturing moving traffic as taken from within a vehicle: Frames 140 and 141

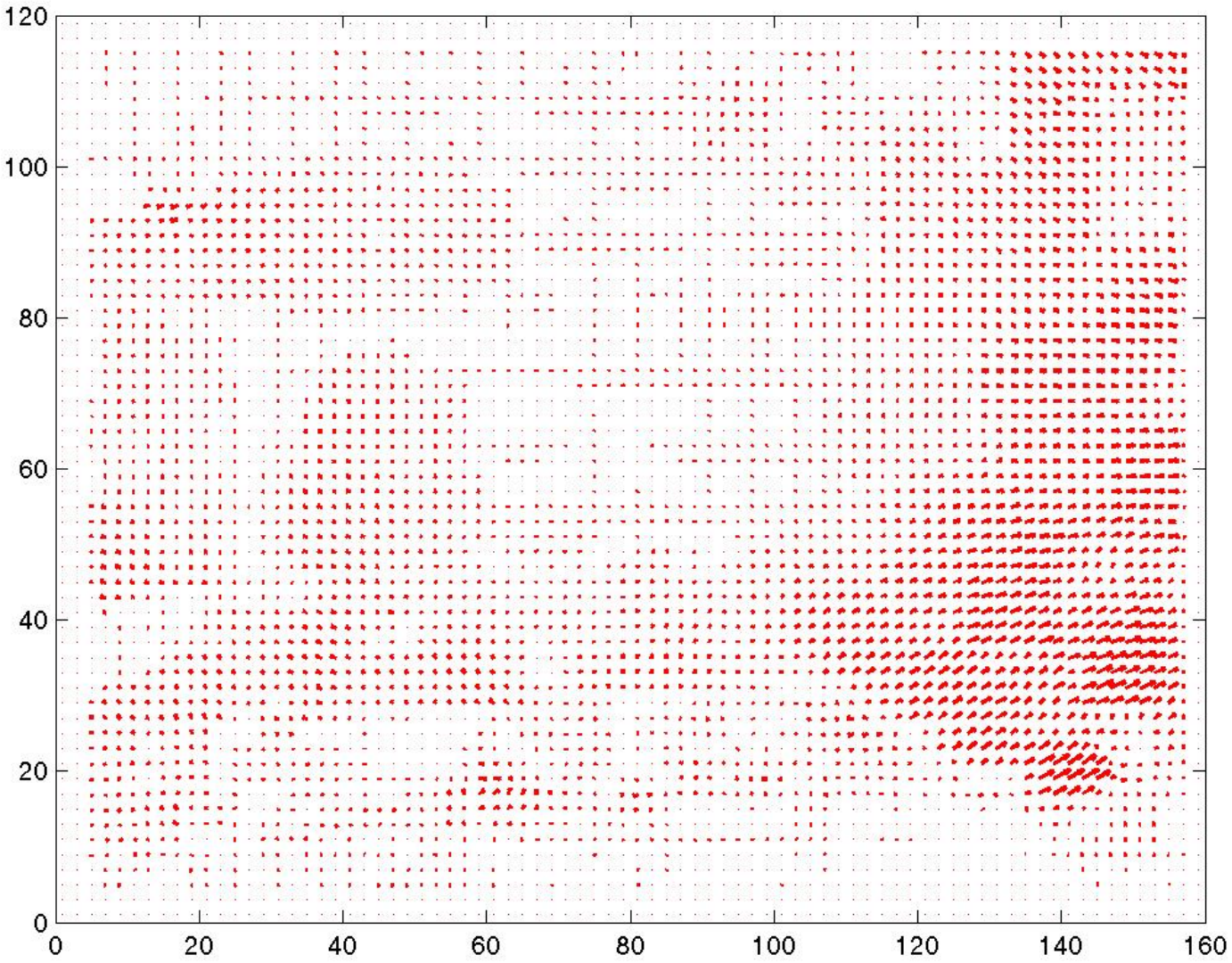

Figure 4.18 Magnified Optical Flow Results for Real Image capturing moving traffic as taken from within a vehicle: Frames 79 and 80

Camera Movement in a Scene

This set of images has been taken from a video sequence [29] where the camera moves instead of the table of flowers. Because of the pan and tilt movement of the camera, the whole scene changes with each progressing frame (or image). The results shown in Figures 4.19 and 4.20 show two different cases of camera movement in this video sequence. The description of the movement is included in each of these figures. Since the camera is moving in this case, each pixel of the image appears to be moving to the optical flow algorithm. Therefore, the optical flow algorithm should be able to detect this relative motion of pixels between objects.

The results show the optical flow vectors all over the image, correctly detecting the motion in this video sequence. Closer observation shows that the optical flow vectors are more or less parallel to each other indicating equal camera motion for the entire scene. Three levels of Gaussian pyramid levels have been considered and the results become more concentrated with each level.

This image sequence is complex because of the presence of different colored flowers, background, table etc. However, the optical flow algorithm does not get affected by the presence of multiple moving objects. The Gaussian pyramid filtering improves these results, further attesting to the fact that as the image becomes slightly less complex, the optical flow results improve even further.

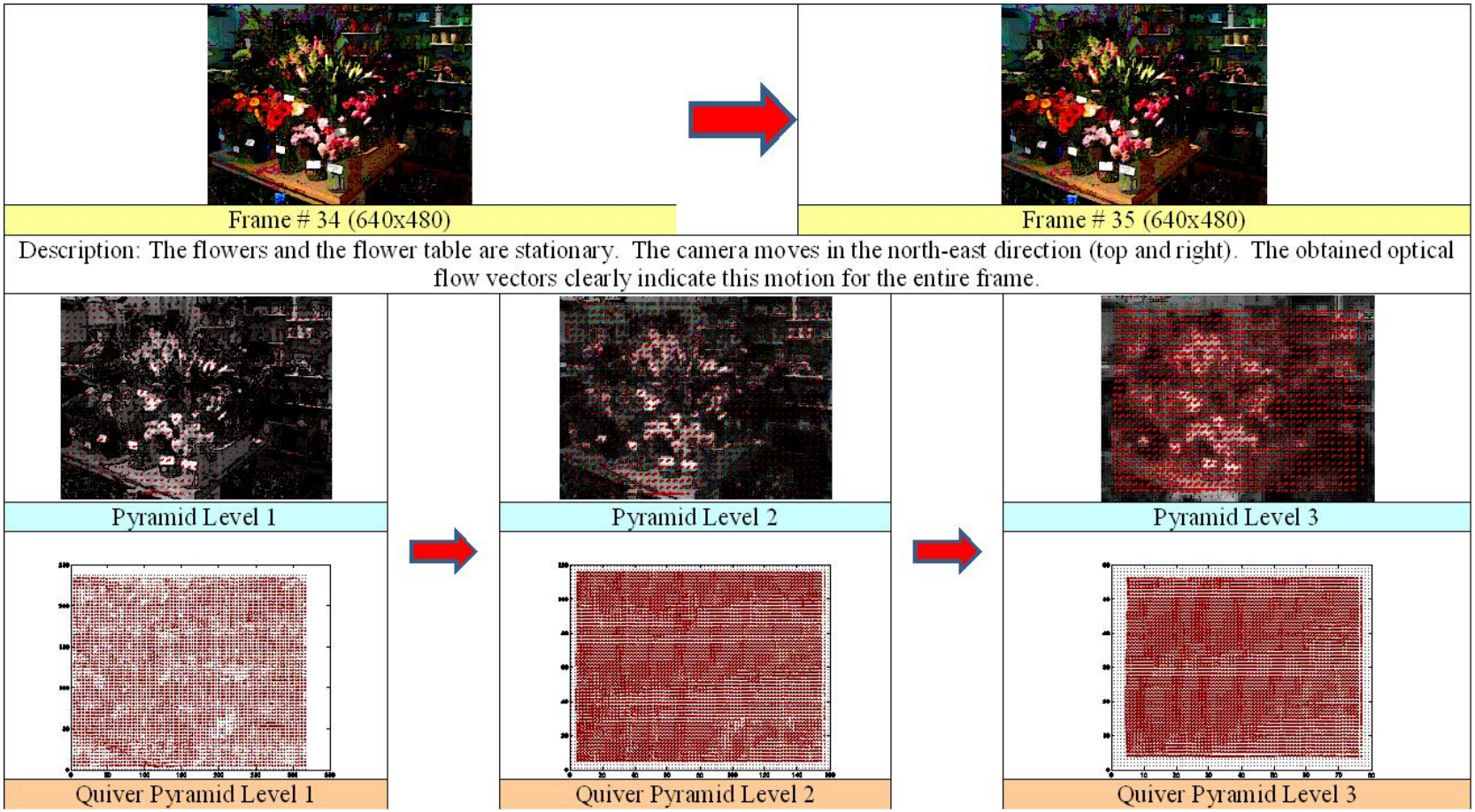

Figure 4.19 Optical Flow Results for Real Image with camera having pan and tilt motion over a table of flowers: Frames 34 and 35

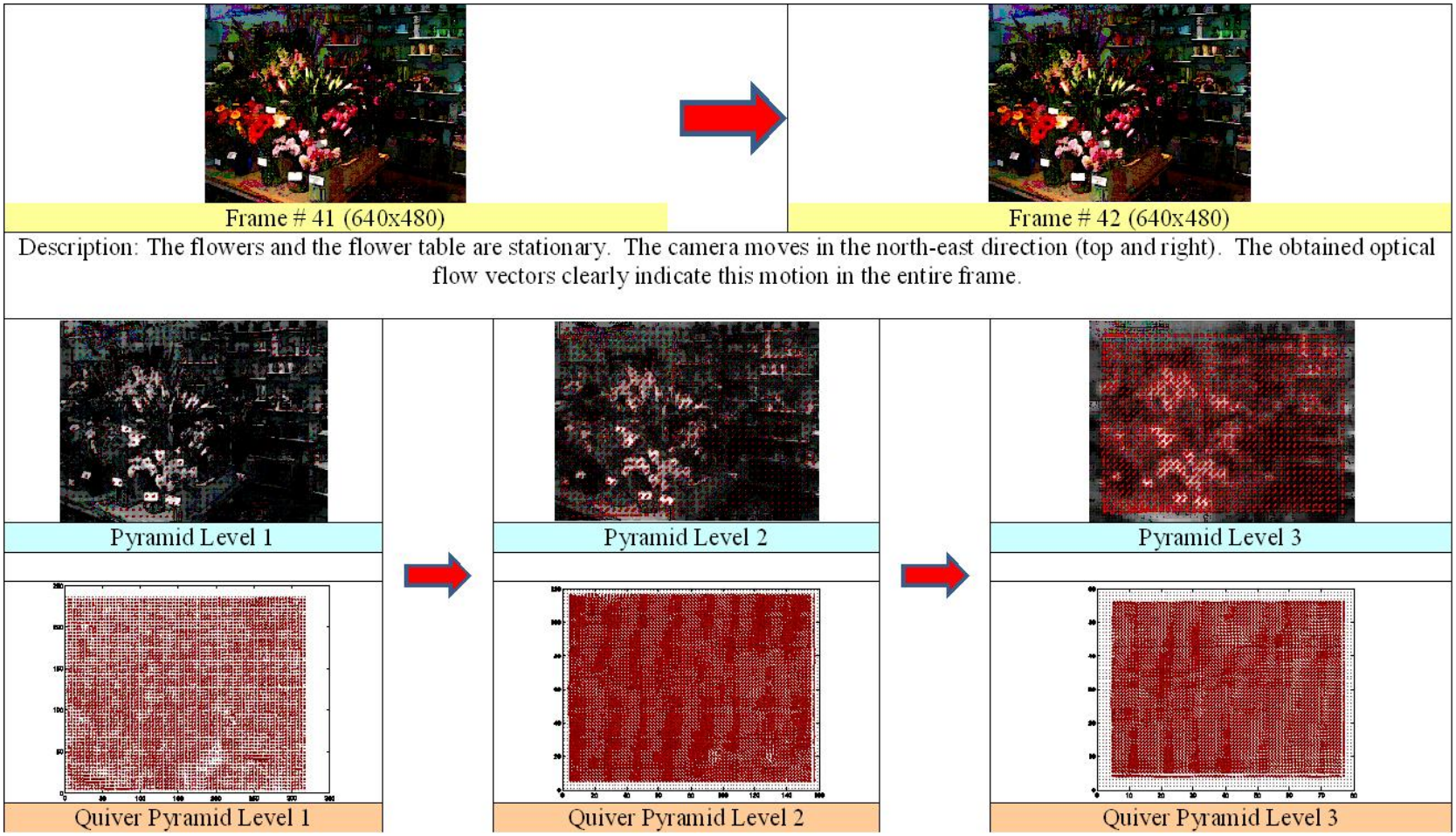

Figure 4.20 Optical Flow Results for Real Image with camera having pan and tilt motion over a table of flowers: Frames 41 and 42

4.4 Discussion

Optical flow results for different types of image sequences were presented in the previous section. The optical flow algorithm responded differently in different image sequences. Gaussian pyramid filtering has been applied in all these video sequences and the number of pyramid levels considered depends on the complexity of the image sequence. A few observations have been made after analyzing these results.

The first image sequence involving a rotating sphere showed that the optical flow algorithm was able to detect individual moving objects (i.e. sphere) in a video sequence. The synthetic image sequence of the sphere and cube combination show that the considered optical flow algorithm did not get affected by occlusion of the non-textured cube by a textured spehere. Another case worth considering would be if a textured surface is occluded by a non-textured one.

The cases considered here also showed that the optical flow algorithm responded better to textured surfaces. This can be seen in the image sequences involving the human hand movement in front of a textured background as well as the human hand and book movement in front of a textured background. The optical flow algorithm was able to perform well even in the case of multiple textured objects. The results further improve with the application of Gaussian pyramids.

The optical flow algorithm has then been tested for a combination of non-textured surfaces. The results from this image sequence involving a relatively non-textured human hand and mug combination showed that the optical flow algorithm is more effective at the base level when non-textured objects are involved. The Gaussian pyramid filtering affects the resolution of non-textured objects which can deteriorate the results with progressive Gaussian pyramid levels.

The results from another case of non-textured objects (rotation of a flower basket arrangement) suggests that if there is enough movement, or considerable lighting and shading changes, the optical flow algorithm performs better. The image sequence of a baby raising his arm had some amount of camera motion (giving rise to a blurring effect due to background motion). The application of Gaussian pyramids in this case improved the performance of the optical flow algorithm. This suggests that Gaussian pyramid filtering can be really useful in complex image

sequences where a motion estimation algorithm may not be able to perform well due to challenges in the image sequence.

The optical flow algorithm has been further tested for complex motions such as zooming effects and camera motion on the entire scene. The results show that the optical flow algorithm performed reliably in both these situations. The Gaussian pyramid filtering further improved the obtained results.

It has been previously discussed that correct estimation of motion is important for accurate segmentation of the moving object(s) from the image sequences. The optical flow results shown in this chapter have been used for segmentation of the moving objects that have been identified. The segmentation techniques discussed previously have been applied to one set of synthetic image sequence and three sets of real image sequences and the results have been presented in the next chapter.

4.5 Summary

This chapter presented the results obtained by the application of the Lucas Kanade Optical Flow algorithm on nine different video sequences. These video sequences represented different types of motions in real and synthetic videos. The obtained results prove the reliability and accuracy of the considered algorithm for different types of video sequences. The results obtained by the usage of Gaussian pyramids have also been presented and analyzed.

5. SEGMENTATION OF MOVING OBJECTS IN A SEQUENCE OF VIDEO IMAGES

5.1 Introduction

The previous chapter presented the motion estimation results obtained using the Lucas Kanade Optical Flow algorithm. These optical flow vector results have been used for the segmentation of some of the image sequences seen previously. These segmentation results are presented in this Chapter. The segmentation results are enhanced by using erosion and dilation techniques and the results have been analyzed. Finally, a comparative analysis between the obtained results with other segmentation techniques has been conducted.

5.2 Segmentation Results from the Optical Flow Method

This section presents segmentation results for four different video sequences. Figure 5.1 shows the segmentation result obtained for Frame 2 of the synthetic image of Sphere, previously shown in Section 4.3.1. The value of threshold used to obtain this segmentation result is 0.02. The threshold technique used is able to correctly separate the rotating sphere from its background. The accuracy of the results can be attributed to the correct estimation of optical flow vectors.

It should be noted that the original pixels of the images have been filled with black color for all the values lying below the threshold value. Figure 5.2 shows the segmentation result obtained for the video sequence of the hand and book combination moving across the textured book shelves. A threshold value of 1.10 gives accurate segmentation results in this case. It can be seen that there are some errors in the background in this figure. These errors can be reduced by using some image enhancement techniques discussed further in Section 5.3.

Figure 5.3 shows the segmentation result for the hand of the baby moving towards his mouth.

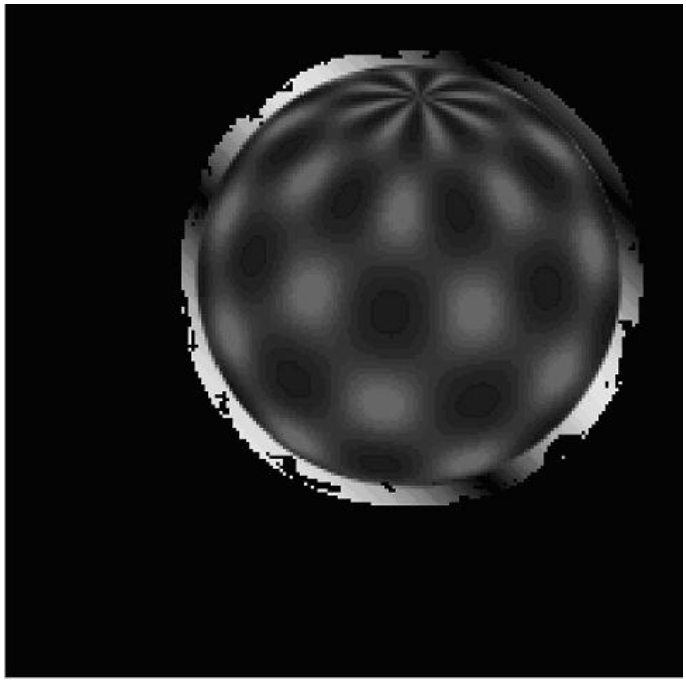

Figure 5.1 Segmentation of the sphere

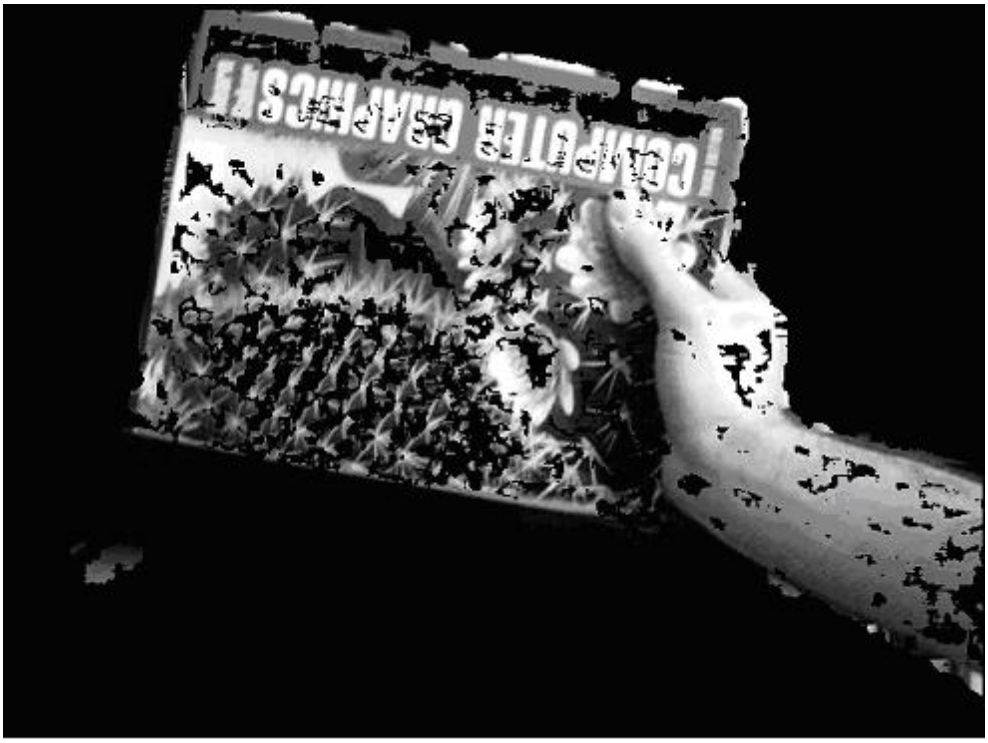

Figure 5.2 Segmentation of the hand-book combination

The optical flow vectors have previously been discussed in Section 4.2.2. The result presented uses a threshold value of 2.0. Other values of threshold were tested as well and the comparison

Figure 5.3 Segmentation for the movement of the baby's hand

of different results has been presented in the next section. Figure 5.4 shows the segmentation of the video sequence with a moving hand. The result shows the clearly segmented shape of the hand over a black background.

The threshold value plays an important role in accurate segmentation. This value is largely dependent on the type of video under consideration. The optical flow vector results have been tested for different threshold values and the segmentation results have been analyzed. To illustrate the importance of this value, two different video sets have been considered. This includes one synthetic and one real image data set.

(a) Threshold Variation for the Synthetic Image Segmentation

The segmentation of video sequence of the sphere rotation has been considered for three different values of thresholds. Figures 5.5, 5.6 and 5.7 show the variation in the segmentation results for threshold values of 0.005, 0.02 and 0.05 respectively. It can be seen that the value of 0.005 is able to segment the sphere with a constant band around the sphere which belongs to the

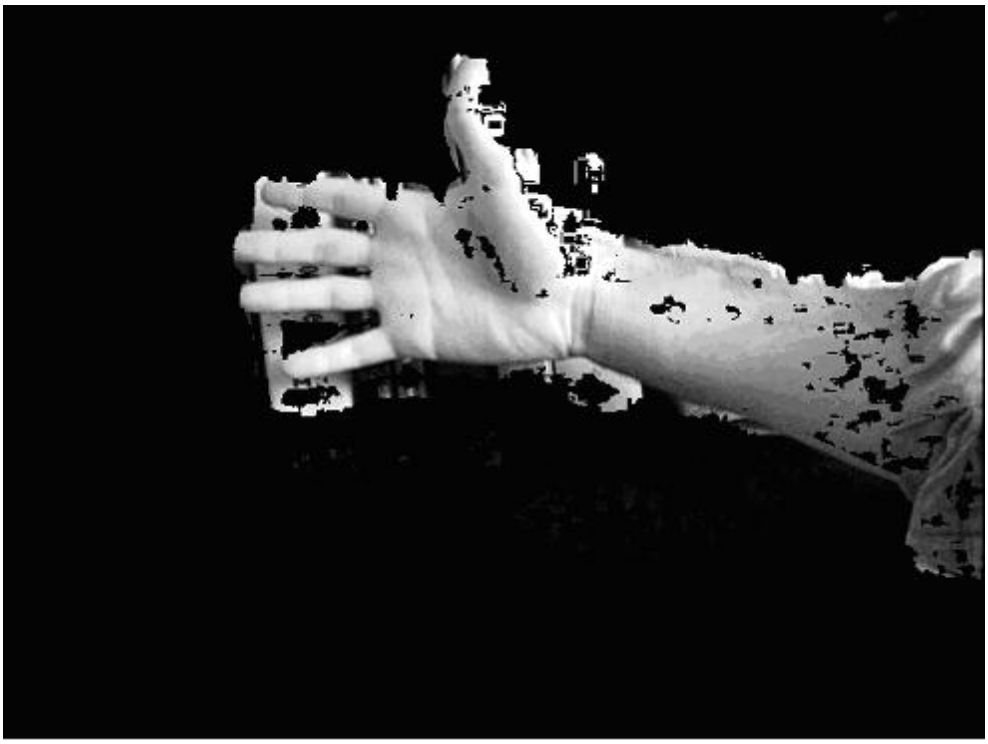

Figure 5.4 Segmentation of the moving hand

background, but is not completely segmented out. This implies that the threshold value is not enough. When the threshold value is increased to 0.02, a large portion of the band present in the first case is segmented out. The threshold value is increased to 0.05 to determine if a further increase is able to segment the background further. It can be seen that an increase of threshold value to 0.05 eliminates more stationary background; however, it also segments small portions of the moving sphere. Therefore, this result is not acceptable. By analyzing these results, it can be concluded that a threshold value slightly higher than 0.02 is a good value for this video sequence to segment images with an acceptable level of accuracy.

(b) Threshold Variation for the Real Image Segmentation

As in the previous case, three values of thresholds have been considered for the video sequence of the moving arm of the baby. The results obtained by using these three values of 1.5, 2.0 and 2.5 are shown in Figures 5.8, 5.9 and 5.10 respectively.

The first result with a threshold value of 1.5 is not able to segment the background from the image correctly. Therefore, it is not acceptable. When the threshold value is increased to

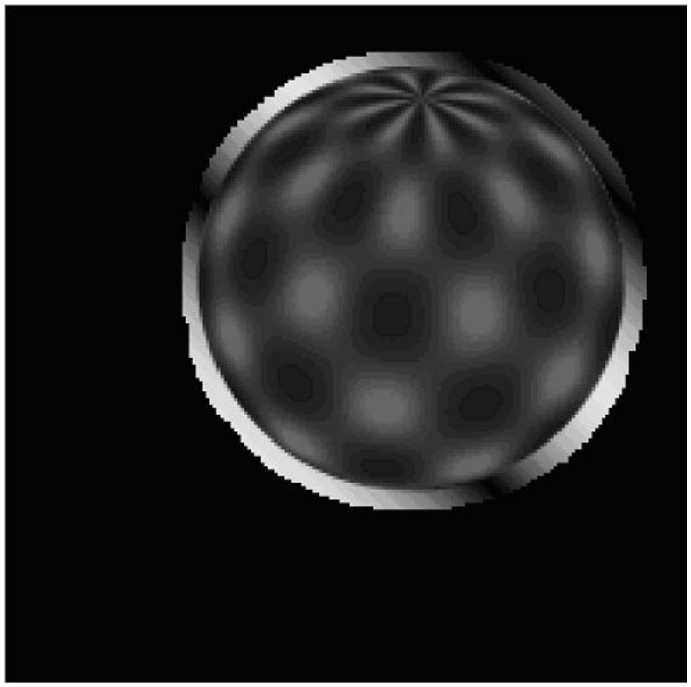

Figure 5.5 Segmentation of the moving sphere using threshold value of 0.005

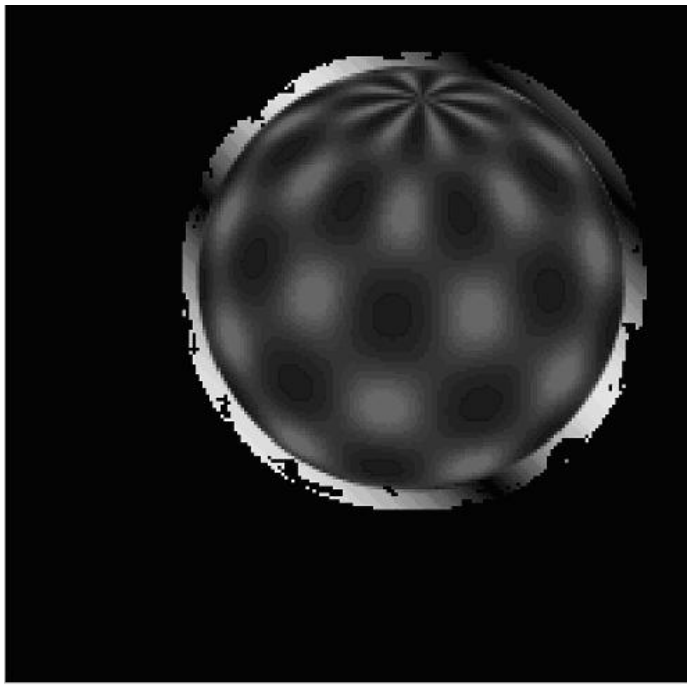

Figure 5.6 Segmentation of the moving sphere using threshold value of 0.02

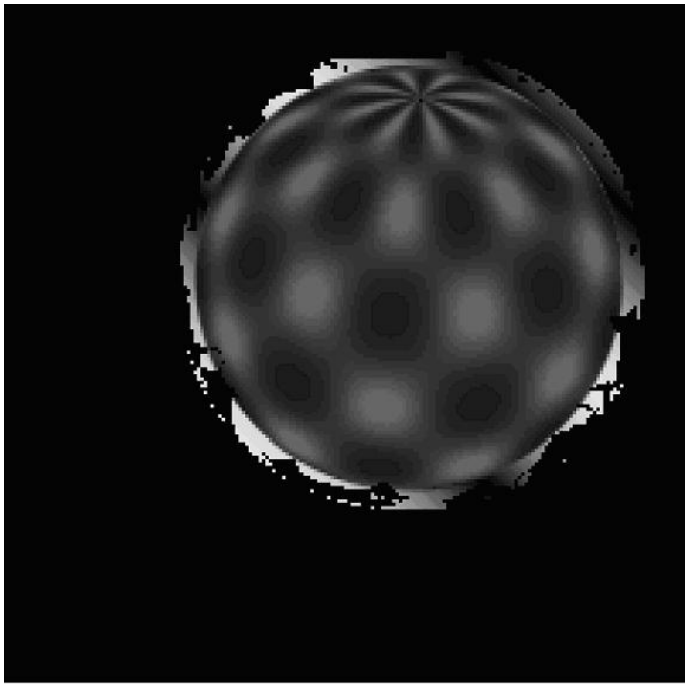

Figure 5.7 Segmentation of the moving sphere using threshold value of 0.05

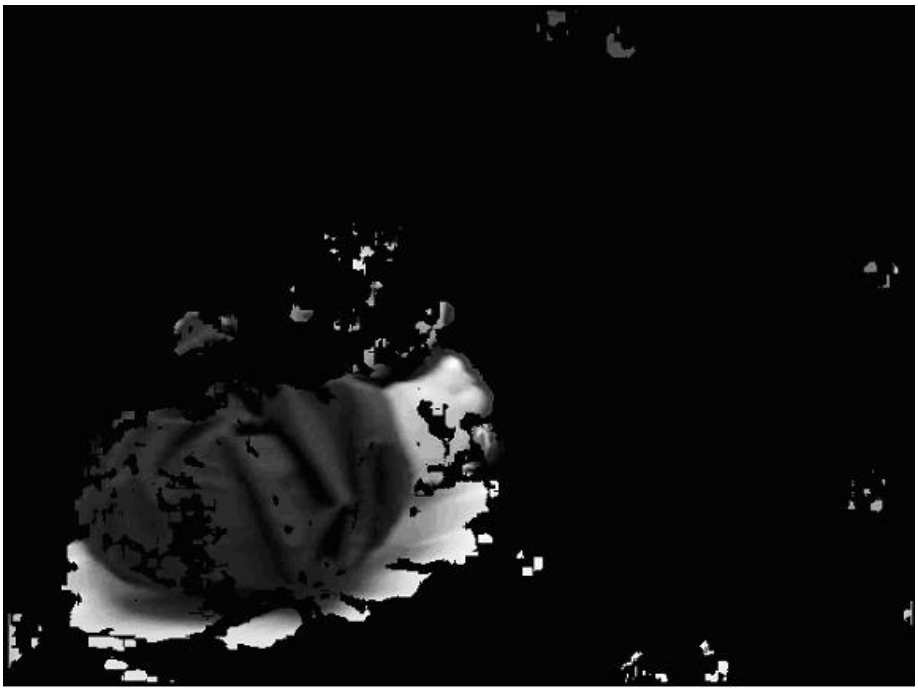

Figure 5.8 Segmentation of the moving arm using threshold value of 1.5

Figure 5.9 Segmentation of the moving arm using threshold value of 2.0

Figure 5.10 Segmentation of the moving arm using threshold value of 2.5

2.0, the stationary background is accurately segmented out with minor spots. If the threshold value is increased further to 2.5, a significant portion of the moving arm is eliminated in the segmentation process. Even though there are minor errors in the second result (threshold value of 2.0), it can be accepted as it is able to balance between the other two results with threshold values of 1.5 and 2.5.

The segmentation process of any video sequence depends on various factors. The results prove that it is imperative to estimate the moving objects accurately before any segmentation operation can be carried. Depending on the video sequence, there may be other factors involved, which could be inter-dependent or completely independent. This makes accurate segmentation through an automated process very challenging. The analysis of results obtained by different threshold shows that it is very important to reach equilibrium where the moving object is segmented within some pre-determined and acceptable limits. As mentioned previously, the implementation of segmentation techniques for online videos should be done after testing a sample video data set to first determine an acceptable threshold level.

5.3 Enhancement of the segmented image content

The segmentation results presented in the previous section are correct within reasonable limits. However, there are certain enhancements that have been made to improve these results further. This section presents the results obtained by applying a combination of these methods. Two such techniques, known as erode and dilate have been used to enhance the segmented image by eliminating noise and retaining the original image morphology. Dilate (or dilation) adds extra pixels to the boundary of objects in the image and causes objects to dilate or grow in size while Erode (or erosion) removes pixels from the boundary of objects in the image and causes objects to shrink in size.

Different stages and combination of erosion and dilation techniques has been used to enhance the segmented image content. The analysis of the results shows that the usage of either of erosion or dilation technique alone was able to enhance only the already segmented portions of the image. However, this method further pronounced some of the existing errors in the segmented image.

The application of a combination of these two techniques improved the already segmented content of the image as well as removed most of the existing errors of the segmented image. It was observed that four consecutive stages of enhancement of the segmented image (using a combination of erosion and dilation techniques) produced the most accurate results. This also reduces the dependency of the segmentation method on the value of threshold. Four different combinations have been tested by combining the erosion and dilation techniques. These combinations are:

(a) Erode → Dilate → Erode → Dilate

(b) Erode → Dilate → Dilate → Erode

(c) Dilate → Erode → Erode → Dilate

(d) Dilate → Erode → Dilate → Erode

The observation of results obtained by the application of these four combinations showed that the third combination of Dilate → Erode → Erode → Dilate performs well for all types of video sequences. The operations of erode and dilate have been implemented using the MATLAB Image Processing Toolbox. The description of these functions and the MATLAB code has been given in Appendix A and Appendix B respectively.

Figures 5.11, 5.12, 5.13 and 5.14 show the four stages of results for the segmented images of sphere, hand and book, baby arm and moving hand which have previously been shown in Figures 5.1, 5.2, 5.3 and 5.4 respectively. All the four sets of results clearly show a significant improvement in the quality of segmentation. The final results after the four-step enhancement process show a significant reduction in background error. Further, it also reduces the errors in the segmented moving objects.

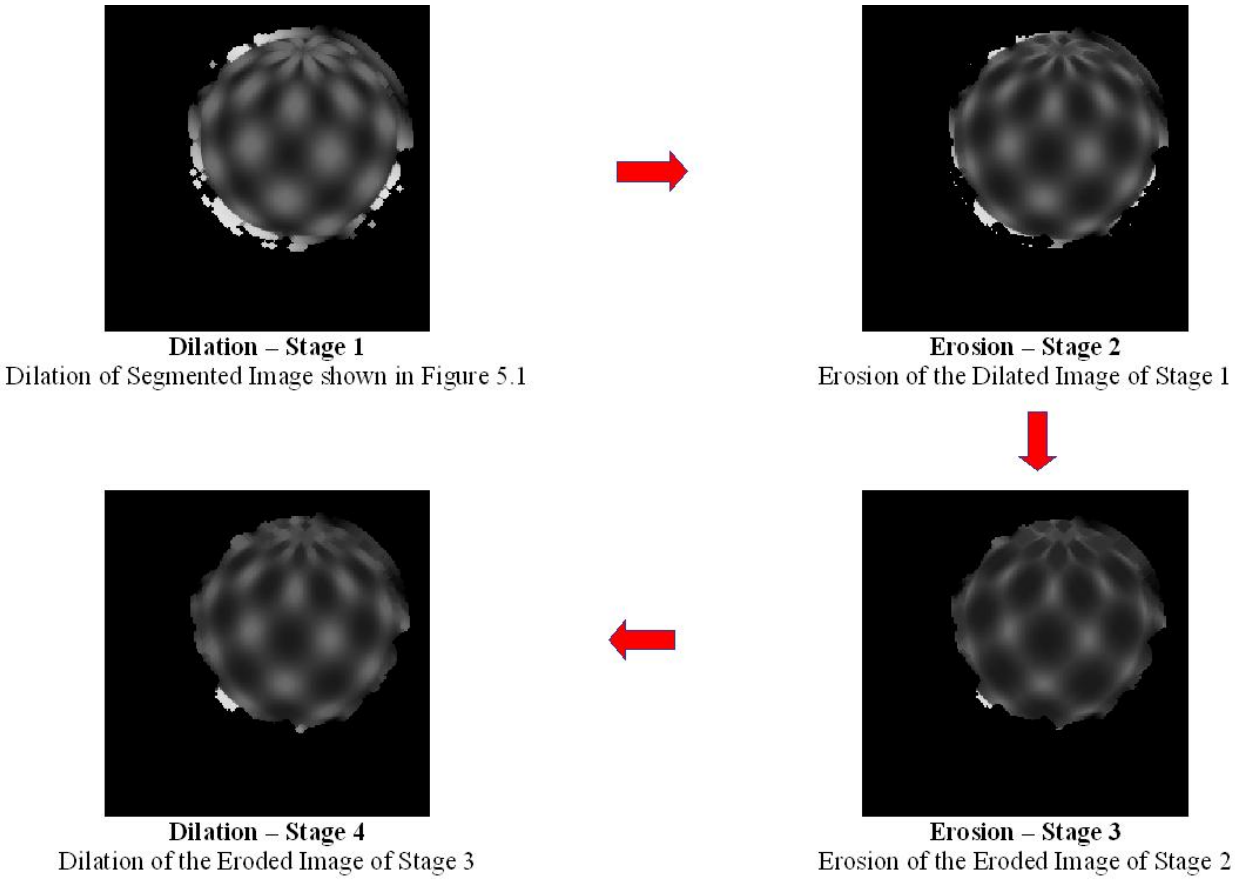

Figure 5.11 Enhancement of the segmented image of the rotating sphere

5.4 Discussion and Comparison with other Image Segmentation Methods

The segmentation results presented in the previous section are considerably accurate. It is essential to re-iterate that correct estimation of motion vectors plays an important and critical role in accurate segmentation of any video sequence. The segmentation methods and the enhancement techniques discussed here are able to perform well only under the circumstances where the moving objects in a video have been identified correctly by the motion estimation algorithm.

The results covered in the previous sections prove the reliability and robustness of the considered optical flow algorithm under varying conditions. The algorithm and the segmentation

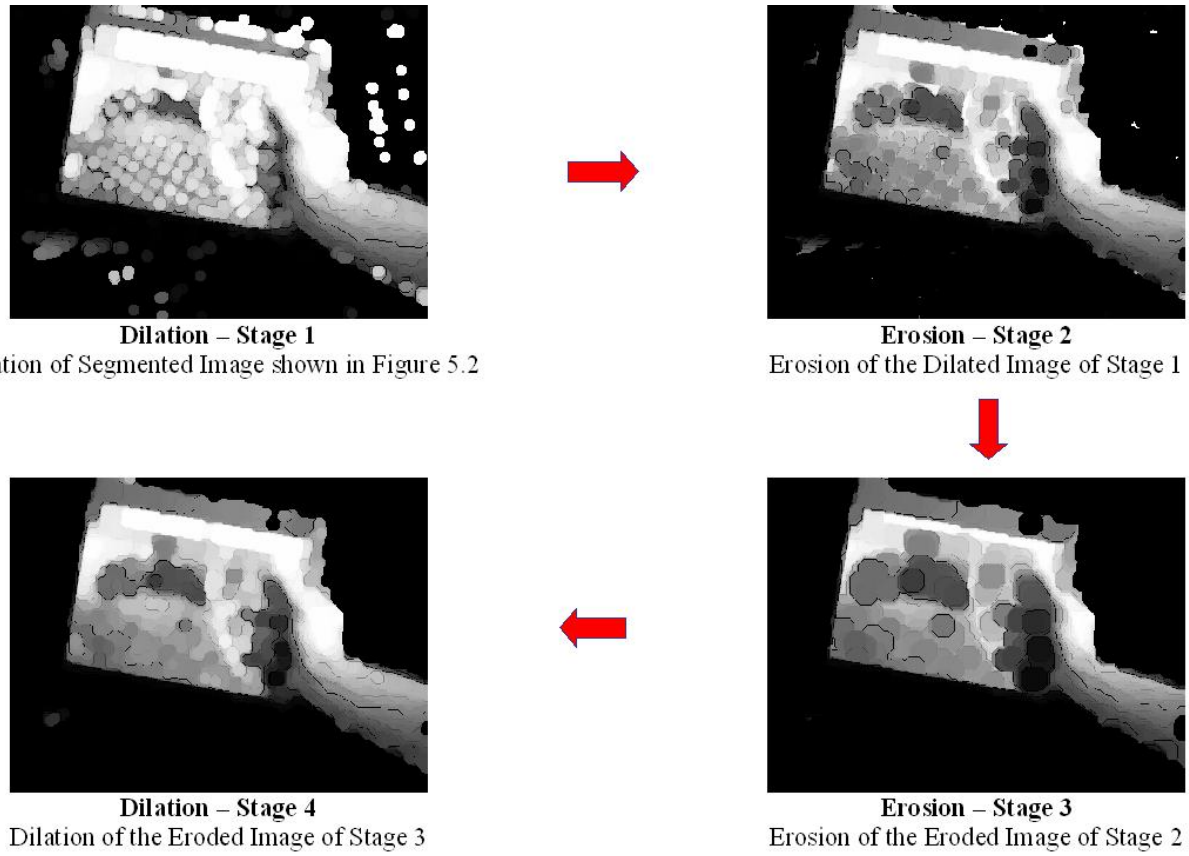

Figure 5.12 Enhancement of the segmented image of the moving hand and book combination

techniques perform well for different types of real and synthetic video sequences. A comparative analysis has been conducted to test the consistency of the considered optical flow algorithm with the other existing methods. A block matching method and a neural network approach has been considered and the findings have been reported in sections 5.4.1 and 5.4.2 respectively.

5.4.1 *Results using a Block Matching Technique*

The different block matching methods have been previously discussed in section 2.3.1. Two of these methods, a full or exhaustive search and $2-D$ logarithmic search have been combined to implement a block matching technique. Macro-blocks of 16×16 have been considered and each macro-block of the reference frame is compared with each macro-block of the target frame. The minimum mean square errors (MSE) calculated by comparing the pixel intensities (R, G and B

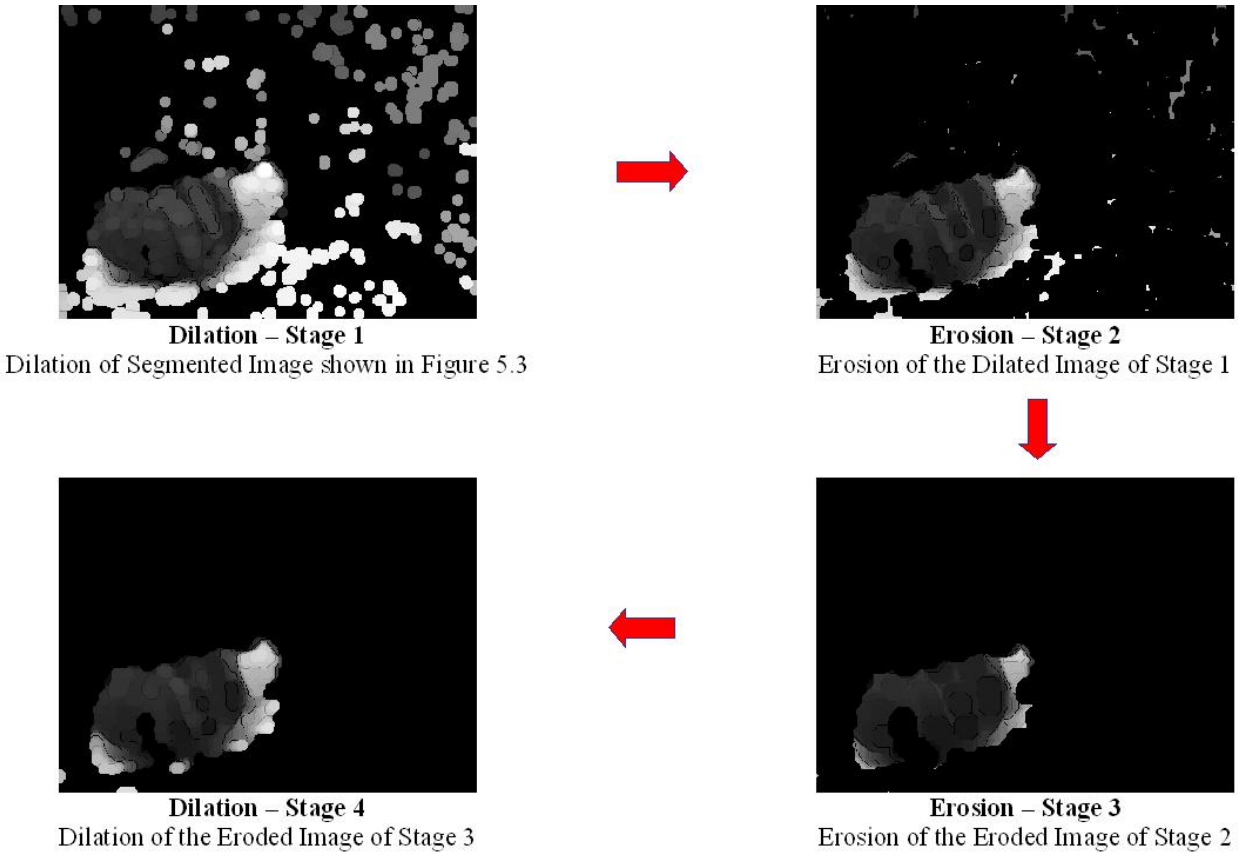

Figure 5.13 Enhancement of the segmented image of the baby's arm

values) for each macro-block of the reference and target frame are recorded. This concludes the full search portion of the algorithm.

The $2-D$ logarithmic search has been used to "micro-search" the best matching block around the vicinity of the macro-block which produced the least error through the full search. To understand this further, consider the first 16×16 macro-block in the reference frame. The macro-block in the target frame which resulted in the minimum mean square error is considered and a search area of 64 pixels is constructed around it. All the macro-blocks in this search area are compared with the reference macro-block using the same MSE criteria to find the best matching block. This process is carried for each of the macro-blocks of the reference frame.

The results obtained for motion vectors through this process are reasonably good as shown in Figures 5.15 and 5.16.

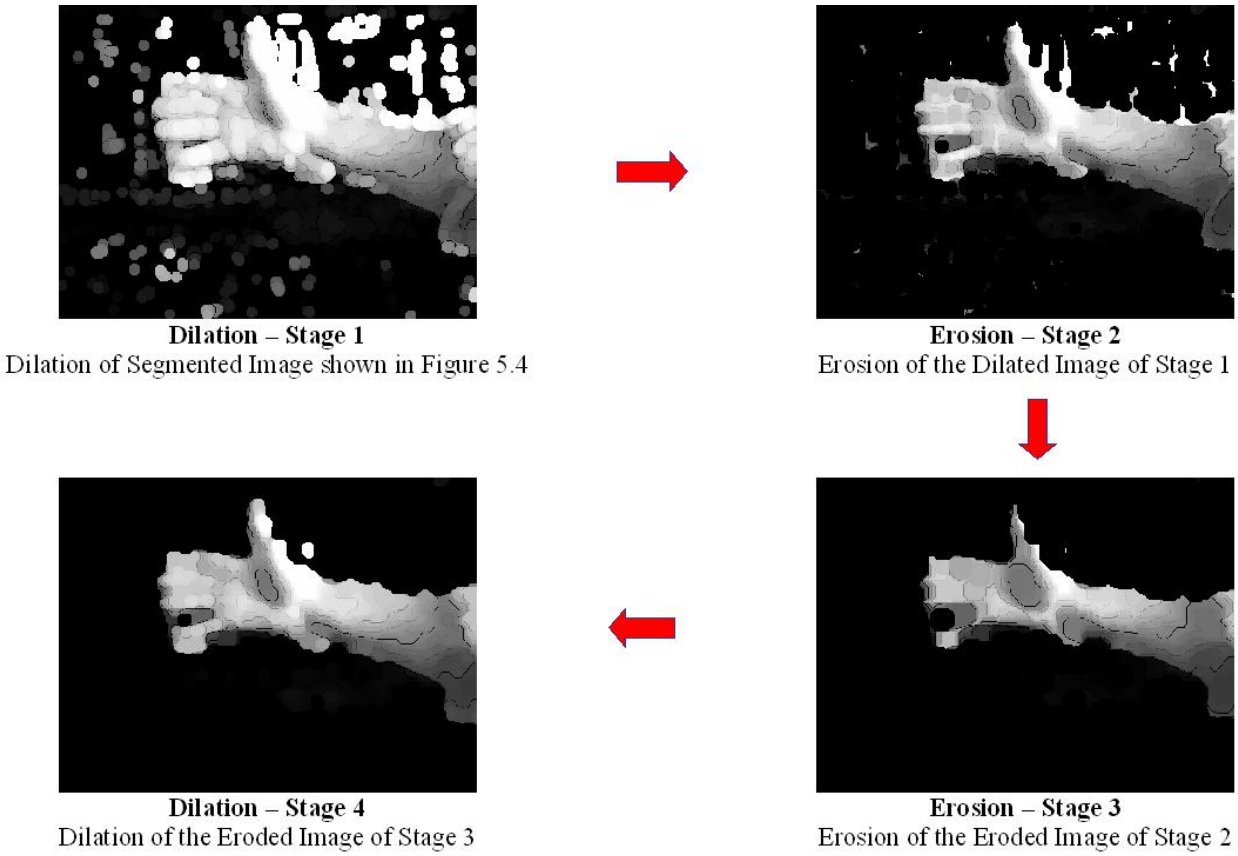

Figure 5.14 Enhancement of the segmented image of the moving hand

It has been observed that the results are more sensitive to lighting changes as compared to the optical flow algorithms. In the obtained segmentation results shown in Figures 5.17 and 5.18, the block matching algorithm is not able to accurately differentiate the moving object from the stationary background. As a result, the obtained segmentation results are not very accurate. This reinforces the importance of accurate motion estimation for segmentation of moving objects.

These results can be improved by employing a more rigorous block matching algorithm. However, this might lead to higher computation time, which needs to be considered as well for real-time implementation. It has been observed that application of the block matching algorithm to the same two image frames is more time consuming as compared to the optical flow algorithm. This aspect must be taken into consideration, especially for implementation on online videos, where computational speed is of critical importance. Therefore, it is important to consider all

Figure 5.15 Motion Vectors between Frames 11 and 12 of a Rotating Flower Basket using Block Matching Method

parameters such as reliability, accuracy and computational time before choosing a particular algorithm and approach.

5.4.2 *Results using a Neural Network Technique*

Neural Networks can be used for motion estimation to separate the moving object from the stationary background on basis of pattern classification. A neural network approach has been used on the previously discussed video sequence of the moving arm of the baby. Since the background remains constant, the neural network is trained to differentiate between pixel intensities of the background and of the moving arm. Once trained, the neural network is used to segment the moving arm from the stationary background in all other frames of the video.

A multi-layer feed forward network (MFNN) with a network configuration of $6 - 12 - 1$

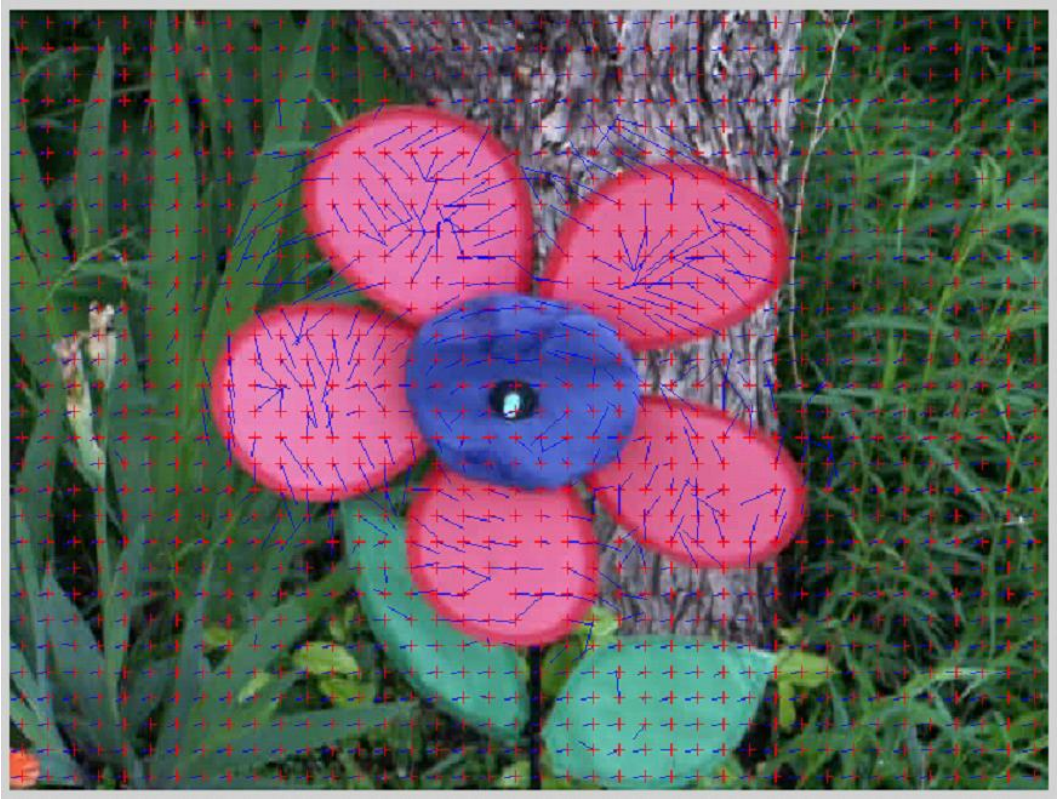

Figure 5.16 Motion Vectors between frames 23 and 24 of a Rotating Pinwheel using Block Matching Method

performs well for this video sequence. The back propagation algorithm has been used to train the MFNN in MATLAB. The MFNN uses log-sigmoid neurons in the input and hidden layer and tangent sigmoid neurons in the output layer. The inputs given to the neural network are in the form of R, G and B pixel intensities and the neural network is trained to give a +1 output for all pixels belonging to the moving arm. It is trained to give a zero (0) output for all pixels belonging to the stationary background. Some of the results obtained by the neural network have been shown in Figure 5.19.

The neural network has been trained by using one frame of the video sequence (i.e., Frame 2). To test the generalization capability of the neural network, it has been tested for segmentation of other frames of the video sequence. Figure 5.19 shows results for frames 2, 4, 6, 9, 11 and 13. The results show that the neural network is able to generalize and segment the moving arm from the rest of the background in all the cases. Therefore, it can be concluded that if a

Figure 5.17 Segmentation results using Block Matching Method for Frame 12 of a Rotating Flower Basket

neural network has been trained properly, it is able to successfully segment the moving objects. It should also be noted that even though only R, G and B pixel intensities have been used to train the neural network in this case, other parameters such as hue, brightness etc. may also be included to improve the neural network recognizing capability.

In circumstances where the background of a video remains constant with time, this approach can be particularly beneficial. Once the neural network is trained properly, it is able to generalize and can be used to produce any number of segmentation results for the same video sequence. Therefore, if the background does not change with passing of time, the neural network is able to detect other moving objects that might also appear in the video. This is one of the main advantages of using this approach. However, the time taken to train the neural network poses a major hindrance for its successful online implementation. Further, the process of training the neural network specific to each video sequence may defeat the purpose of using a neural network

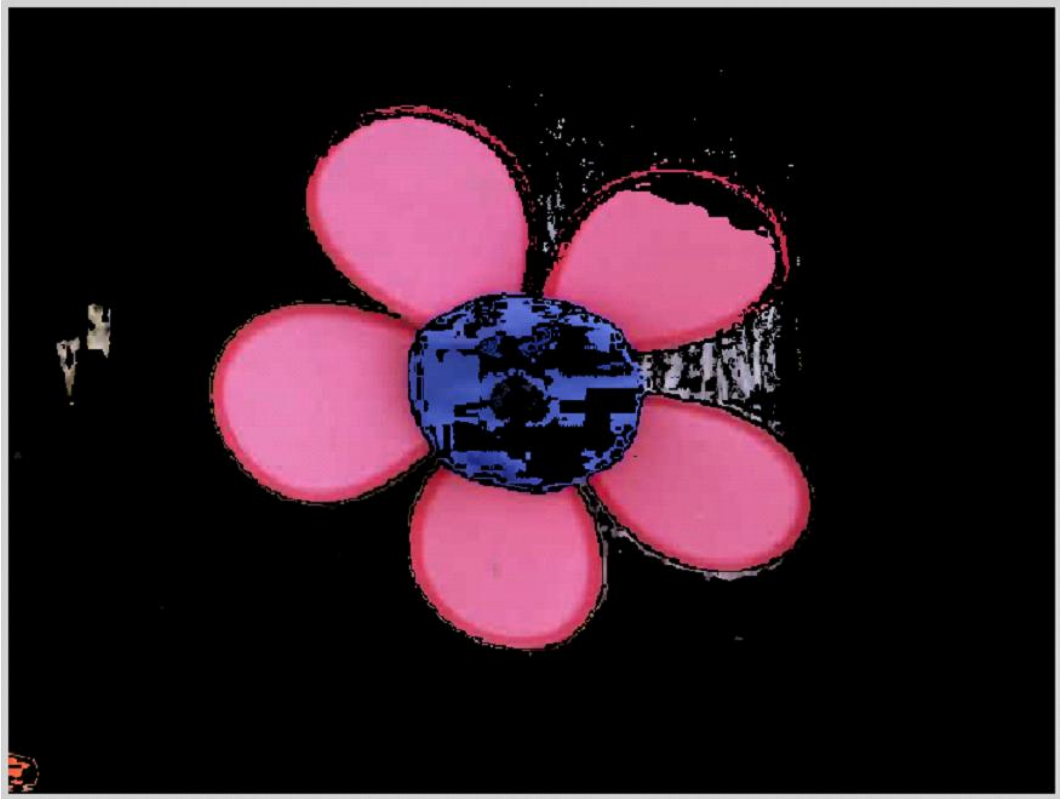

Figure 5.18 Segmentation results using Block Matching Method for Frame 24 of a Rotating Pinwheel

for segmentation i.e., generalization.

5.5 Summary

This Chapter presented the segmentation results obtained for different video sequences. The concept of varying the threshold values and enhancement of segmented content of image has been discussed and the results have been presented. The Chapter also discussed the conclusions that can be drawn by observing the presented results. A comparative analysis of the algorithm with other motion estimation methods has also been presented.

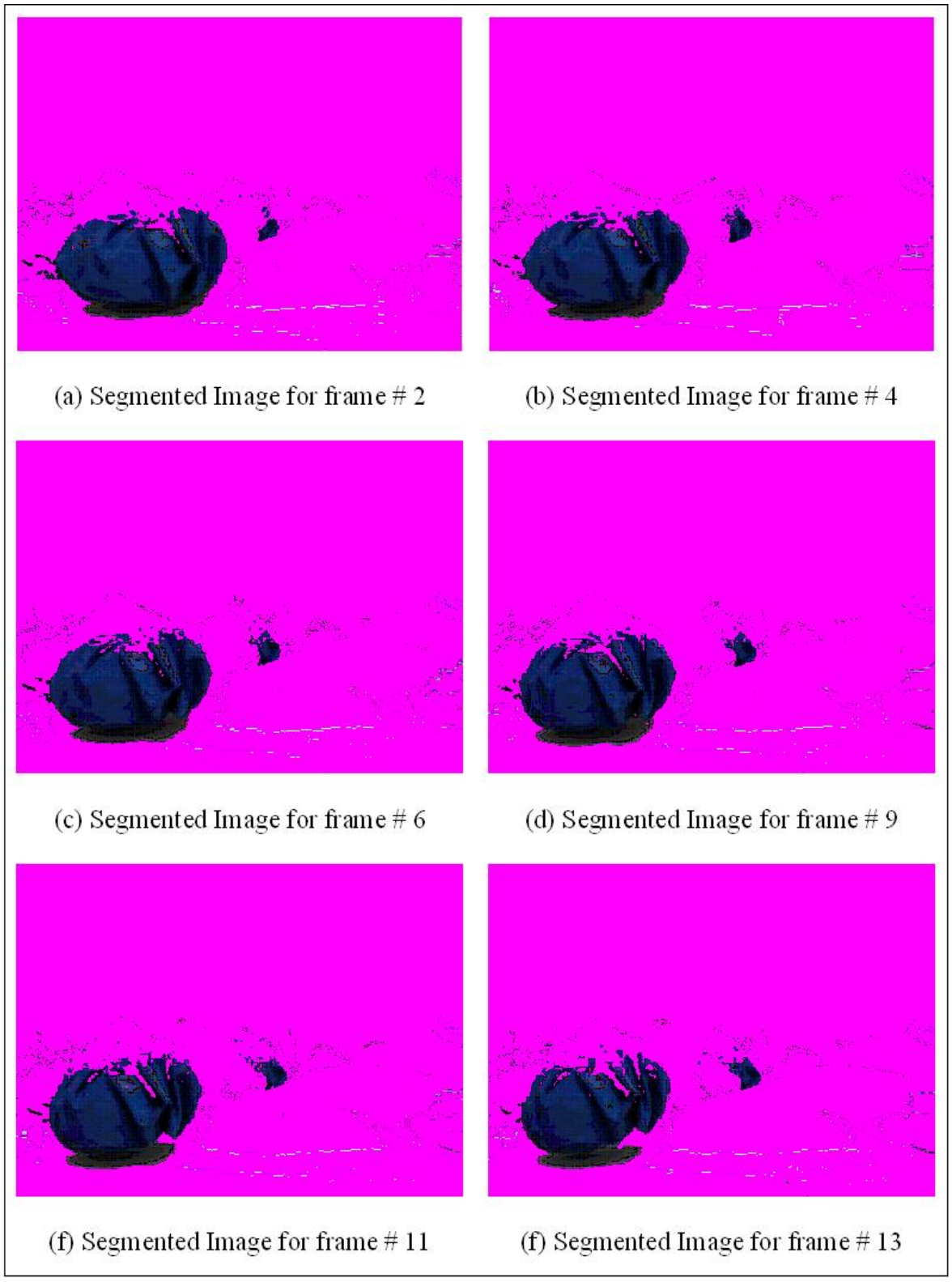

Figure 5.19 Segmentation results using a neural network technique

6. SUMMARY AND FUTURE WORK

6.1 Summary and Conclusions

There has been a significant development in the computer vision technologies in recent times. The digital video technology is emerging rapidly, especially in the area of video surveillance and video processing. There has been a vast improvement in the technology of digital cameras, digital camcorders and other video technologies, which until the 1990s were used only by specialist users. These developments have led to lowering the costs of video equipment, thus leading to mainstream usage of digital video technologies. Home videos are becoming more popular than ever before and users can edit the videos by means of inexpensive (and sometimes freely available open-source code) software using their personal computers.

Detection of moving objects in a video sequence is finding its application in the areas computer vision, video surveillance, video processing and 2.5 dimensional structure estimation from motion (SfM). The estimated motion is then used to segregate the moving objects from the stationary background in the video sequence. The accurate estimation of motion plays a vital role in segmentation of moving objects; therefore, it needs careful analysis and assessment.

Review of the different motion estimation techniques has been conducted. The three main methods currently used are block matching techniques, optical flow techniques and artificial intelligence methods. All the methods have been described in Chapter 2. The Lucas Kanade Optical Flow method has been chosen for the detection of motion in this research project. The Lucas Kanade method, like all optical flow methods, is based on relative motion between objects. It calculates horizontal and vertical optical flow vectors. The optical flow vectors depend on the apparent motion of image brightness patterns of image frames/video.

The challenges such as different types of motion, lighting changes etc. faced during correct determination of motion have been considered and the algorithm has been tested for robustness

in such conditions. The parameters that affect the results, such as neighborhood size have been included in the algorithm. Another parameter that has been included in the algorithm design is Gaussian Pyramid Filtering. Gaussian pyramids are helpful in motion estimation in complex scenes and videos. Each pyramid level reduces the size of the frame/image by one quarter, thus, reducing the resolution and complexity of the image. This helps in accurate estimation and concentration of optical flow vectors over the moving object.

Various real and synthetic video sequences have been considered. These video sequences represent a wide range of motions. The results obtained by the application of the Lucas Kanade optical flow algorithm and Gaussian pyramids for all the different data sets are very encouraging. The results prove that the algorithm is able to perform well, irrespective of the type of video under consideration.

There are various existing segmentation techniques that are used for different types of video sequences. After analyzing all the techniques, the threshold technique has been selected. The horizontal and vertical optical flow vectors obtained from the Lucas Kanade method have been used to calculate the threshold to segment the moving objects in a video. Further, the impact of different threshold values on the segmentation results has been analyzed. Both, the optical flow algorithm and the thresholding have been implemented using MATLAB and its Image Processing Toolbox.

The obtained segmentation results clearly define the moving object in the video, which can be attributed to accurate optical flow vectors. Two techniques, erode and dilate have been used for enhancing the segmented image content. Different combination of these two methods have been tested and a four step method of dilate $\rightarrow$ erode $\rightarrow$ erode $\rightarrow$ dilate has been chosen. The results achieved by this process show a significant improvement from the original segmentation results. This enhancement has been tested for four different data sets and as observed, it performs well for all of them.

The optical flow algorithm considered in this research project has been tested for robustness and reliability under different conditions. The algorithm has provisions to include various parameters which affect the accuracy of motion estimation algorithms. The algorithm has been

compared with two other motion estimation techniques: the block matching method and the neural network technique. The results obtained from both these techniques have been discussed. The block matching technique is not able to match the accuracy of the optical flow algorithm for the same computational time. The neural network technique is able to produce good segmentation results. However, the training process of the neural network is time consuming and may not be a feasible solution for implementation in real time videos.

6.2 Future Work

This section presents some directions for future work that can be undertaken.

Online Implementation:

The optical flow algorithm provides good motion estimation and segmentation results for offline videos. The implementation of this algorithm for real time videos can be undertaken as future work.

Computational Time (speed) vs Accuracy:

The application of motion detection is mainly in video surveillance. For online videos, speed of computation is of critical importance. On the other hand, it is imperative that the motion estimation algorithm is accurate and reliable to correctly spot the moving objects in the video. Therefore, it is very important to achieve a balance between speed and accuracy for any method under consideration. The motion estimation methods can be explored further to determine the method best suited for video surveillance applications.

References

[1] L. A. Alexandre and A. C. Campilho, "A 2D Image Motion Detection Method Using a Stationary Camera," *10th Portuguese Conference on Pattern Recognition*, pp. 103-107, 1998.

[2] M. Alkanhal, D. Turaga and T. Chen, "Correlation Based Search Algorithms for Motion Estimation," *Picture Coding Symposium*, Portland (USA), April 21-23 1999.

[3] T. Bakir and S. J. Reeves, "A Filter Design Method for Minimizing Ringing in a Region of Interest in MR Spectroscopic Images," *IEEE Transactions on Medical Imaging*, Vol. 19, No. 6, pp. 585-600, June 2000.

[4] M. Chen, L. Chen and T. Chiueh, "One-Dimensional Full Search Motion Estimation Algorithm For Video Coding," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 4, No. 5, pp. 504-509, October 1994.

[5] D. H. Cooper and J. Graham, "Estimating Motion in Noisy, Textured Images: Optical Flow in Medical Ultrasound," *Proceedings of British Machine Vision Conference, University of Manchester, 1996*, Poster Session 2, 1996.

[6] E. De Castro and C. Morandi, "Registration of Translated and Rotated Images Using Finite Fourier Transforms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, September 1987.

[7] M. Ghanbari, "The Cross-Search Algorithm for Motion Estimation," *IEEE Transactions on Communications*, Vol. COM-38, No. 7, pp. 950-953, July 1990.

[8] S. Haykin, *Neural Networks: A Comprehensive Foundation*, Second Edition, Prentice Hall, ISBN: 0132733501, 1999.

[9] B. K. P. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intelligence*, Vol. 17, pp. 185-203, 1981.

[10] J. Jain and A. Jain, "Displacement Measurement and its application in interframe image coding", *IEEE Transactions on Communications*, Vol. COM-29, pp. 1799-1808, December 1991.

[11] R. Li, B. Zeng, and M. L. Liou, "A New Three-step Search Algorithm for Block Motion Estimation," *IEEE Transactions on Circuits and Systems for Video Technology, Speech and Signal Processing*, Vol. 4, No. 4, pp. 438-442, August, 1994.

[12] B. D. Lucas, *Generalized Image Matching by the Method of Differences*, Ph.D. Dissertation, Carnegie Mellon University, 1981.

[13] B. D. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision," *International Joint Conference on Artificial Intelligence*, pp. 674-679, 1981.

[14] B. D. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision," *Proceedings of Imaging Understanding Workshop*, pp. 121-130, 1981.

[15] B. McCane, K. Novins, D. Crannitch, and B. Galvin, "On Benchmarking Optical Flow," *Proceedings of Computer Vision and Image Understanding*, pp. 126-143, 2001.

[16] J. Morel and S. Solimini, *Variational Methods in Image Segmentation: with seven image processing experiments*, Birkhuser, ISBN: 0817637206, 1995.

[17] MPEG Video Sequences, K. Takaya, October 2003.

[18] E. Navon, O. Miller and A. Averbuch, "Color Image Segmentation based on Automatic Derivation of Local Thresholds," *Proceedings of the 7th Australian Pattern Recognition Society Conference*, Vol. 2, pp. 571-580, 2003.

[19] S. Negahdaripoiur, "Revised interpretation of optical flow for dynamic scene analysis," *IEEE transactions on Pattern Analysis and Machine Intelligence*, Vol. 20, No. 9, pp. 961-979, 1998.

[20] L. Po and W. Ma, "A Novel Four-Step Search Algorithm for Fast Block Motion Estimation," *IEEE transactions on Circuits and Systems for Video Technology*, Vol. 6, No. 3, pp. 313-317, June 1996.

[21] A. Puri, H. Hang and D. Schilling, "An Efficient Block-matching Algorithm for Motion Compensated Coding," *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 25.4.1-25.4.4, 1987.

[22] T. J. Ross, *Fuzzy Logic with Engineering Applications*, International Edition, Mc-Graw Hill, Inc. , ISBN: 0470860758, 1995.

[23] R. Sinivasan, K. Rao, "Predictive coding based on efficient motion estimation," *Proceedings of International Conference on Communications*, Amsterdam, Part 1, pp. 521-526, 1988.

[24] L. G. Shapiro and G. C. Stockman, *Computer Vision*, Prentice Hall, ISBN: 0130307963, pp. 279-325, 2001.

[25] C. Sun, H. Talbot, S. Ourselin and T. Adriaansen, "Digital Image Computing: Techniques and Applications," *Proceedings of the VIIth Australian Pattern Recognition Society Conference, Australian Pattern Recognition Society*, Vol. 2, 2003.

[26] Y. Q. Shi, H. Sun, *Image and Video Compression for Multimedia Engineering: Fundamentals, Algorithms and Standards*, CRC Press, ISBN: 0849334918, 2000.

[27] K. Takaya, "Detection of Moving Objects in Video Scene - MPEG like Motion Vector vs. Optical Flow," *The First International Workshop on Video Processing for Security*, Canada, 2006.

[28] K. Takaya and R. Malhotra, "Tracking Moving Objects in a Video Sequence by the Neural Network trained for motion vectors," *Proceedings of IEEE Pacific Rim Conference on Communications*, pp. 153-156, 2005.

[29] The Computer Graphics Group at the University of Virginia *http://www.cs.virginia.edu/ gfx/Courses/2007/ ComputerVision*, May 2007.

[30] *The MathWorks, MATLAB Image Processing Toolbox, User's Guide*, Version 5.0.

[31] A. Tran, K. Liu, K. Tzou and E. Vogel, "An Efficient Pyramid Image Coding System," *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Vol. 12, pp. 744-747, 1987.

[32] S. E. Umbagh, *Computer Imaging: Digital Image Analysis and Processing*, CRC Press, ISBN: 0849329191, 2005.

[33] University of Otago, Computer Vision Homepage *http://www.cs.otago.ac.nz/research/vision/index.html*, May 2007.

[34] J. Weng, T. S. Huang, N. Ahuja, *Motion and Structure from Image Sequences*, CRC Press, ISBN: 3540556729, 1992.

[35] G. Yun, P. Pan and Y. Kang, "Research on a New Gaussian Self-Adaptive Smoothing Algorithm in Image Processing," *Proceedings of IEEE International Workshop on VLSI Design and Video Technology*, pp. 348-352, 2005.

[36] T. Zahariadis and D. Kalivas, "A Spiral Search Algorithm for Fast Estimation of Block Motion Vectors," *Proceedings of the European Signal Processing Conference 1996*, Vol. 2, pp. 1079-1082, 1996.

A. APPENDIX A
MAIN FUNCTIONS USED IN MATLAB

A.1 Main MATLAB Commands

The main commands used in the MATLAB code have been listed below. The description for each of these commands has been taken from MATLAB product help family.

Table A.1 Description of MATLAB Commands Used

'	Transpose of a matrix
,	Next Column
.*	Array Multiply
conv2	Two dimensional convolution of two specified matrices
disp	Displays the array, without printing the array name
double	Convert to double precision
error	Display message and abort function
flipud	Flip matrix in up/down direction
figure	Create new figure window
ginput	Graphical input from mouse
hold on	Holds the current plot
input	Prompt for user input
load	Load MATLAB workspace variables (.mat format)
num2str	Converts number to string
pinv	Pseudoinverse
quiver	Plots velocity vectors as arrows with vector components at the given points
uint8	Convert to unsigned 8-bit integer

A.2 Image Processing Toolbox Commands

Various Image Processing Toolbox in-built functions have been used for different operations. The table below shows the commands used and the operations performed by these in-built functions. The description has been taken from MATLAB Product Help Family.

Table A.2 Description of Image Processing Commands Used

RGB2GRAY	This command converts RGB images or colormap to grayscale by eliminating the hue and saturation information while retaining the luminance.
IMREAD	Reads the image from a graphics file. If the file contains a grayscale intensity image, output is a two-dimensional array. If the file contains a truecolor (RGB) image, output is a three-dimensional (M-by-N-by-3) array.
IMWRITE	Writes the image to graphics file. The graphics file can be either a grayscale image (M-by-N) or a truecolor image (M-by-N-by-3).
IMDILATE	Dilates the grayscale, binary, or packed binary image and returns a dilated image. It uses a structuring element object (SE), or array of structuring element objects. If the image is logical and the structuring element is flat, IMDILATE performs binary dilation; otherwise, it performs grayscale dilation. If SE is an array of structuring element objects, IMDILATE performs multiple dilations, using each structuring element in SE in succession.
IMERODE	Erodes the grayscale, binary, or packed binary image and returns an eroded image. It uses a structuring element object (SE), or array of structuring element objects. If the image is logical and the structuring element is flat, IMERODE performs binary erosion; otherwise it performs grayscale erosion. If SE is an array of structuring element objects, IMERODE performs multiple erosions of the input image, using each SE in succession.

B. APPENDIX B MATLAB CODE FOR IMPLEMENTATION OF THE LUCAS KANADE ALGORITHM

This Appendix presents the details of the MATLAB code that has been used for the implementation of the Lucas Kanade Optical Flow Algorithm.

B.1 Main Program of Lucas Kanade Algorithm (`mainprogram.m`)

```
%==================== CODE FOR OPTICAL FLOW TO DETECT MOTION IN IMAGES ====================%

%========== THIS IS THE MAIN PROGRAM FOR LUCAS KANADE OPTICAL FLOW METHOD ==================%

clear; % clear all files in the current workspace
clc; % clear command window
% The Lucas Kanade Algorithm works only on gray images, so there is a provision to convert
% the color images into gray, if needed.

% Input from user to test if the images are colored or gray

user_ip1 = input('Are the images colored or gray? Please enter 1 for colored and 0 for gray:  ');
if user_ip1 ==1 % Converting the images into double format

    image1 = double(rgb2gray(imread('Book\00000001.jpg')));
% Reading Image 1 after converting into gray - Reference Image

    image2 = double(rgb2gray(imread('Book\00000002.jpg')));
% Reading Image 2 after converting into gray - Target Image

elseif user_ip1== 0
    image1= double(imread('Book\00000001.jpg'));
% Reading Image 1 - Reference Image
```

```
    image2= double(imread('Book\00000001.jpg'));
% Reading Image 2 - Target Image
end

if or(size(image1,1)~= size(image2,1), size(image1,2)~=size(image2,2))
% Test to check if the images are of the same size
    error('The images chosen are not of the same size');
% Error message if images are of not the same size
end

disp(' The original image will be taken as the base or 0th Pyramid Level');
% Display message for the user

pyrlevels = input('Please enter the number of levels (>=1) of pyramids that you want to use:  ');
% User Input for Number of levels of pyramids

winsize = input('Please enter the size of the smoothing window (neighbourhood size):  ');
% User Input for Neighbourhood size for smoothing

[height,width] = size(image1); % Reading the size of the Image
image1_P0 = image1;
% 0 or Base Level Pyramid, Image 1 - Same size as input image

image2_P0 = image2;
% 0 or Base Level Pyramid, Image 2 - Same size as input image

[u0,v0] = LucasKanade(image1_P0,image2_P0,winsize);
% u0 and v0 are the base level horizontal and vertical optical flow vectors respectively
% LucasKanade.m is the function to solve the optical flow equation

u0_quiver=flipud(u0);v0_quiver=flipud(v0);
% Flipping the calculated vectors in up/down direction to plot optical flow vectors

figure(1);
% New figure # 1 to plot quiver vectors at the base level

quiver(1:(width/80):width, 1:(height/60):height,u0_quiver(1:(height/60):height,1:(width/80):width),
v0_quiver(1:(height/60):height,1:(width/80):width),'r','LineWidth',1.5);
% Plot quiver u0 and v0 vectors on the current plot in red color with a line width of 1.50

print( 1, '-dbmp', 'quiver_P0');
% printing the base level quiver vectors in .bmp format

print( 1, '-djpeg', 'quiver_P0');
% printing the base level quiver vectors in .jpeg format
```

```
figure(2);
% New figure # 2 to superimpose quiver vectors at the base level on the image

imshow(uint8(image1));
% Show the reference image in the new figure window, converted into uint8 format for display

hold on
% figure 2 is held to plot quiver vectors on the same figure

quiver(1:(width/80):width, 1:(height/60):height, u0(1:(height/60):height,1:(width/80):width),
v0(1:(height/60):height,1:(width/80):width),'r','LineWidth',1.5);
% Plot quiver u0 and v0 vectors  on the current plot in red color with a line width of 1.50

print( 2, '-dbmp', 'quiver_superimp_P0.bmp');
% Printing the base pyramid level quiver vectors superimposed on the reference image in .bmp format

print( 2, '-djpeg', 'quiver_superimp_P0.jpg');
% Printing the base pyramid level quiver vectors superimposed on the reference image in .jpeg format

fig_num=3;
% Count to start next figure starting at # 3

for x = 1:pyrlevels
% FOR loop for the specified pyramid levels

label1 = [num2str('quiver_P') num2str(pyrlevels)];
% Creating a label for quiver vector images for different pyramid levels

label2 = [num2str('quiver_superimp_P') num2str(pyrlevels)];
% Creating a label for quiver vector images for different pyramid levels

image1 = pyramid(image1);
% Current Pyramid Level for Reference Image is calculated by using function pyramid.m

image2 = pyramid(image2);
% Current Pyramid Level for Target Image is calculated by using function pyramid.m

[height,width] = size(image1);
% Reading the size of the new image reduced due to Gaussian filtering

[u,v] = LucasKanade(image1,image2,winsize);
% New horizontal and vertical flow vectors (u and v) calculated using function LucasKanade.m

u_quiver=flipud(u);v_quiver=flipud(v);
```

```
% Flipping the calculated vectors in up/down direction to plot optical flow vectors

figure(fig_num);
% New figure to superimpose quiver vectors at the base level on the image

quiver(1:(width/80):width,1:(height/60):height,u_quiver(1:(height/60):height,1:(width/80):width),
v_quiver(1:(height/60):height,1:(width/80):width),'r','LineWidth',1.5);
% Plot quiver u and v vectors  on the current plot in red color with a line width of 1.50

print(fig_num, '-dbmp', label1);
% Printing the current pyramid level quiver vectors image in .bmp format using the generated label

print(fig_num, '-djpeg', label1);
% Printing the current pyramid level quiver vectors image in .jpeg format using the generated label

figure(fig_num+1);
% Increasing figure number for next figure

imshow(uint8(image1));
% Show the Reference Image

hold on
% figure is held to plot quiver vectors on the same figure

quiver(1:(width/40):width,1:(height/30):height, (1:(height/30):height,1:(width/40):width),
(1:(height/30):height,1:(width/40):width),'r','LineWidth',1.5);
% Plot quiver u and v vectors  on the current plot in red color with a line width of 1.50

print(fig_num+1, '-dbmp', label2 );
% Printing the current pyramid level quiver vectors superimposed on the reference image
% in .bmp format using the generated label

print(fig_num+1, '-djpeg', label2 );
% Printing the current pyramid level quiver vectors superimposed on the reference image
% in .jpeg format using the generated label

fig_num = fig_num+2;

% Increasing figure number for next figure
end
```

B.2 Lucas Kanade Program for Solution of the Optical Flow Equation (LucasKanade.m)

```
%==== Function Lucas Kanade to calculate the Optical Flow by Solving the Optical Flow Equation ====%

function [u,v] = LucasKanade(image1,image2,winsize);
% Image1, Image2 and Neighborhood Size from the mainprogram.m

% Calculating derivatives for images in x, y and t
xderivative_filt = 0.25 *[-1 1; -1 1];
% Filter Coefficients for x-direction

yderivative_filt = 0.25 *[-1 -1; 1 1];
% Filter Coefficients for y-direction

tderivative_filt = 0.25 *[ 1 1; 1 1];
% Filter Coefficients for time t



% Calculation of Derivative of Images - Average of Image1 and Image2
Ix = conv2(image1,xderivative_filt) + conv2(image2,xderivative_filt);
% Applying Convolution Theorem (Filtering) to Image1 and Image2 with respect to the x-direction

Iy = conv2(image1,yderivative_filt) + conv2(image2,yderivative_filt);
% Applying Convolution Theorem (Filtering) to Image1 and Image2 with respect to the y-direction

It = conv2(image1,tderivative_filt) + conv2(image2,-tderivative_filt);
% Applying Convolution Theorem (Filtering) to Image1 and Image2 with respect to time

u = zeros(size(image1));
% Initialize Horizontal Optical Flow vector matrix

v = zeros(size(image2));
% Initialize Vertical Optical Flow vector matrix

halfsize = floor(winsize/2);
% Calculating half of neighborhood size

% Solution of the Optical Flow Equation
for i = (halfsize+1):(size(Ix,1)- halfsize)
% Starting halfwinsize later and ending halfwinsize before
   for j = (halfsize+1):(size(Ix,2)-halfsize)
% Starting halfwinsize later and ending halfwinsize before
```

```
      delIx = Ix(i-halfsize:i+halfsize, j-halfsize:j+halfsize);
% Adjusting the x-derivative to account for the neighborhood size

      delIy = Iy(i-halfsize:i+halfsize, j-halfsize:j+halfsize);
% Adjusting the y-derivative to account for the neighborhood size

      delIt = It(i-halfsize:i+halfsize, j-halfsize:j+halfsize);
% Adjusting the t-derivative to account for the neighborhood size

      delIx = delIx';
% Transpose of the matrix for calculation
      delIy = delIy';
% Transpose of the matrix for calculation
      delIt = delIt';
% Transpose of the matrix for calculation

      delIx = delIx(:);

      delIy = delIy(:);

      delIt = -delIt(:);
% Adding the negative sign to t-derivative to solve the equation

      A = [delIx delIy];
% Matrix A of the Optical Flow Equation

      U = pinv(A'*A)*A'*delIt;
% Right Hand Side of the Optical Flow Equation

      u(i,j)=U(1);
% Horizontal Component - First Row of the Matrix

      v(i,j)=U(2);
% Vertical Component - Second Row of the Matrix

   end
end
```

B.3 Gaussian Pyramids (`pyramid.m`)

```
%============ Function Pyramid to reduce the size of the images and apply Gaussian Filter ============%

% This function applies a gaussian mask (LP filter) to reduce high frequency noise
```

```
% i.e., smoothing the image

function reduceimage = pyramid(im);
% A 5-tap Gaussian Mask  which will be applied to each dimesion of the images
% [0.25-a/2 0.25 a 0.25 0.25-a/2] with a = 0.4
% An odd symmetric normalized gaussian pyramid with sum of weights Wi = 1

gauss_mask = [0.0500  0.2500  0.4000  0.2500  0.0500];
%  Assigning the value of Gaussian Mask

gauss_filter = gauss_mask' * gauss_mask ;
% Calculating the Gaussian filter = square of the Gaussian Mask

size_im = size(im);
% Reading the size of the original image

new_size = ceil(size_im/2);
% New reduced size for the output size - divide by 2

reduceimage = zeros(new_size,class(im));
% Initializing the new reduced size image and filling it with zeros array

% FOR loop for adding two rows and two columns of padding to all the boundaries
m = [-2:2];
% Count for rows and columns

im = [ im(1,:) ; im(1,:); im ; im(size_im(1) ,:) ; im(size_im(1) , :) ];
% Adding two rows each to top and bottom of image

im = [ im(:,1)  im(:,1) im  im(:,size_im(2))  im(:,size_im(2)) ];
% Adding two columns each to left and right of image

% Extracting values for the reduced size image
for i = 0:new_size(1) - 1
    for j = 0:new_size(2) - 1
        im1 = im(2*i+m+3, 2*j+m+3).* gauss_filter;

% Application of Gaussian filter to the image
        reduceimage(i+1, j+1) = sum(im1(:));
% New reduceimage is calculated

    end
end
```

B.4 Image Segmentation (segmentation.m)

```
%================ Segmentation of the image obtained from Optical Flow Results ================%

% FOR loop to test the value of threshold for each pixel of the Image
for i = 1:480
% Height of the image
    for j = 1:640
% Width of the image
        threshold = sqrt((u0(i,j)^2) + (v0(i,j)^2));
% Calculation of the threshold for each pixel

        if threshold >=  1.1
% Test to check if the threshold of the pixel is within the specified value of Threshold

            new_image(i,j) = image1_P0(i,j);
% New Image pixel = Original Base Level Pixel
        else
            new_image(i,j) = [0];
% New Image pixel is segmented by making it Black

        end
    end
end

figure;
% New figure

imshow(uint8(new_image));
% Display the new image

imwrite(uint8(new_image),'segmentation1pt1.bmp')
% Saving the produced the image in .bmp format
```

B.5 Dilation (dilate.m)

```
%================ Function Dilate to Enhance Segmented Image Content ================%

function dilated_im = dilate(im, disksize,count)
% Function takes user defined disksize (similar to neighborhood size),
% count keeps a track of the stage level

dilated_im = imdilate(im,strel('disk',disksize));
```

```
% Dilating the Image according to specified disksize

label = [num2str('dilate') num2str(count) num2str('.bmp')];
% Creating a label for the new dilated image

imwrite(dilated_im,label,'bmp')
% Saving the new Image in .bmp format
```

B.6 Erosion (`erode.m`)

```
%================ Function Erode to Enhance Segmented Image Content ================%

function  eroded_im = erode(im,disksize,count)
% Function takes user defined disksize (similar to neighborhood size),
% count keeps a track of the stage level

eroded_im = imerode(im,strel('disk',disksize));
% Eroding the Image according to specified disksize

label = [num2str('erode') num2str(count) num2str('.bmp')];
% Creating a label for the new eroded image

imwrite(eroded_im,label,'bmp');
% Saving the new Image in .bmp format
```

B.7 Filter Values

This section gives the values of different filter coefficients that have been used in this project. The filter coefficients used by the Lucas Kanade Optical Flow Algorithm to calculate derivatives with respect to the x-direction, y-direction and time t, are given in B.7.1, B.7.2 and B.7.3 respectively. B.7.4 and B.7.5 give the value of the Gaussian mask and Gaussian filter coefficients respectively.

B.7.1 Filter Coefficients to Calculate Derivatives with respect to the x-direction

$$\text{Filter Coefficients for the x-direction} = \begin{bmatrix} -0.2500 & 0.2500 \\ -0.2500 & 0.2500 \end{bmatrix}$$

B.7.2 Filter Coefficients to Calculate Derivatives with respect to the y-direction

$$\text{Filter Coefficients for y-direction} = \begin{bmatrix} -0.2500 & -0.2500 \\ 0.2500 & 0.2500 \end{bmatrix}$$

B.7.3 Filter Coefficients to Calculate Derivatives with respect to time, t

$$\text{Filter Coefficients for time} = \begin{bmatrix} 0.2500 & 0.2500 \\ 0.2500 & 0.2500 \end{bmatrix}$$

B.7.4 Gaussian Mask used for filtering

$$\text{Gaussian Mask} = \begin{bmatrix} 0.05 & 0.25 & 0.40 & 0.25 & 0.05 \end{bmatrix}$$

B.7.5 Filter Coefficients used for Gaussian pyramids

$$\text{Filter Coefficients for Gaussian Pyramid} = \begin{bmatrix} 0.0025 & 0.0125 & 0.0200 & 0.0125 & 0.0025 \\ 0.0125 & 0.0625 & 0.1000 & 0.0625 & 0.0125 \\ 0.0200 & 0.1000 & 0.1600 & 0.1000 & 0.0200 \\ 0.0125 & 0.0625 & 0.1000 & 0.0625 & 0.0125 \\ 0.0025 & 0.0125 & 0.0200 & 0.0125 & 0.0025 \end{bmatrix}$$