

MASTER THESIS



Optical Flow Features for Event Detection

Mohammad Afrooz Mehr, Maziar Haghpanah

Embedded and Intelligent Systems, 120 credits

Halmstad, 2013-12

Optical Flow Features for Event Detection

Master Thesis Report

IDE 1324

2013, December

Authors: Mohammad Afrooz Mehr and Maziar Haghpanah

Supervisor: Dr. Stefan Karlsson

Examiner: Prof. Antanas Verikas

Preface

This master thesis has been done at the School of Information Science, Computer and Electrical Engineering as a result of our studies at Halmstad University, Sweden. The report aims investigating the current works and approaches in this domain, optimizing the existing algorithms, making a new way of estimating optical flow by use of the Gaussian pyramids for event detections and proposing the achieved results and a solution to help future studies in this area.

Deep and special thanks go to Stefan Karlsson, our supervisor at Halmstad University, for his helpful and unlimited comments, guidance, and supports.

To Prof. Antanas Verikas, our examiner at Halmstad University for his patience and attentions.

To our family for their emotional supports along the way of this degree work.

Mohammad Afrooz Mehr

Maziar Haghpanah

Halmstad University, December 2013

Abstract

In this thesis, we employ optical flow features for the detection of the rigid or non-rigid single object on an input video.

For optical flow estimation, we use the Point Line [PL] method [2] (as a local method) to estimate the motion of the image sequence which is generated from the input video stream.

Although the Lukas and Kanade [LK] is a popular local method for estimation of the optical flow, it is weak in dealing with the linear symmetric images even by use of regularization [e.g. Tikhonov].

The PL method is more powerful than the LK method and can properly separate both line flow and point flow. For dealing with rapidly changing data in some part of an image (high motion problem), a gaussian pyramid with five levels (different image resolutions) is employed. In this way, the pyramid height (Level) must be chosen properly according to the maximum optical flow that we expect in each section of the image without iteration.

After determining the best-estimated optical flow vector for every pixel, the algorithm should detect an object on video with its direction to the right or left. By using techniques such as segmentation and averaging the magnitude of flow vectors the program can detect and distinguish rigid objects (e.g. a car) and non-rigid objects (e.g. a human).

Finally the algorithm makes a new video output that includes detected object with flow vectors, the pyramid levels map which has been used for optical flow estimation and a respective binary image.

Contents

1 Introduction	- 1 -
1.1 Problem Statement (Motivation)	- 1 -
1.2 Approach Chosen to Solve the Problem.....	- 3 -
1.3 Limitations	- 4 -
1.4 Thesis Goals and Contribution	- 6 -
2 Background, Definitions and Related works	- 8 -
2.1 Optical flow and Motion	- 8 -
2.2 Optical flow calculation and Aperture problem	- 9 -
2.3 Optical flow methods	- 11 -
2.4 Gaussian Pyramid of Gradients.....	- 21 -
3 Estimation of optical flow vectors	- 24 -
3.1 Estimated Optical flow vectors for each pyramids' level.....	- 24 -
3.2 MSE (Mean Squared Error)	- 28 -
3.3 Comparing of Error Vectors in Different Pyramid Levels.....	- 32 -
4 Optical flow algorithm's results.....	- 35 -
4.1 Capturing and Collecting best optical flow vectors	- 35 -
5 Event detection	- 39 -
5.1 Making grayscale and binary image	- 39 -
5.2 Segmentation and Moment calculation.....	- 41 -
5.3 Results on Events Detection	- 44 -
5.4 Conclusions and suggestions for future works	- 46 -
Appendix	- 48 -
1 Developing of the "PL" algorithm.....	- 48 -
1.1 Starting with making video samples	- 48 -
1.2 How to set input parameters.....	- 51 -
2 Calculation of error vectors (displacement vector) for each pixel ..	- 54 -

2.1 Estimated Optical flow Vectors (Red) and Calculated flow motions (Blue) with desired speed	- 55 -
2.2 Calculation of MSE (Mean Squared Error).....	- 56 -
References:.....	- 57 -

Table of Figures:

Figure 1: Looking to the homogeneous contour through an aperture .	- 1 -
Figure 2: Lines are moving from right to left.....	- 2 -
Figure 3: Lines are moving top to bottom.....	- 2 -
Figure 4: The sample frame from input video	- 5 -
Figure 5: The barber pole illusion.....	- 8 -
Figure 6: Geometrical explanation of the optical flow constraint equation..	- 9 -
Figure 7: Apertures problem [11]	- 10 -
Figure 8: The uniform motion of a set of points (right) and lines (left) [1]..	- 16 -
Figure 9: Down sampling by an integer factor $M=2$	- 21 -
Figure 10: Gaussian pyramid levels with reduction rate 2.	- 21 -
Figure 11: Estimated optical flow for pyramid Level 0,510*510pixels-	- 24 -
Figure 12: Estimated optical flow for pyramid Level 1,253*253pixels-	- 25 -
Figure 13: Estimated optical flow for pyramid Level 2,125*125pixels-	- 25 -
Figure 14: Estimated optical flow for pyramid Level 3, 61*61pixels ..	- 26 -
Figure 15: Estimated optical flow for pyramid Level 4, 29*29 pixels .	- 26 -
Figure 16: Two consecutive frames in Level 2.....	- 27 -
Figure 17: Error vector, Pyramid Level 0	- 28 -
Figure 18: Error vector, Pyramid Level 1	- 29 -
Figure 19: Error vector, Pyramid Level 2	- 30 -
Figure 20: Error vector, Pyramid Level 3	- 30 -
Figure 21: Error vector, Pyramid Level 4.	- 31 -
Figure 22: The output video builds from different pyramid levels.....	- 35 -
Figure 23: The output video with pyramids levels.....	- 36 -
Figure 24: Collecting the most accurate estimated optical flow	- 37 -
Figure 25: The Gray scale and Binary image of rigid object	- 39 -
Figure 26: The Gray scale and binary image of non-rigid object	- 40 -
Figure 27: The Color coding definition.....	-40 -

Figure 28: Detecting of moving rigid object	- 43 -
Figure 29: Detecting of moving non-rigid object.....	- 43 -
Figure 30: Flow chart object Recognition	- 44 -
Figure 31: Car detection's graph	- 45 -
Figure 32: Human detection's graph.....	- 45 -
Figure 33: Video sample with aliase error(without anti-aliasing filter) -	49 -
Figure 34: Video sample without aliase error (with anti-aliasing filter)-	49 -
Figure 35: Circle check board "video sample"	- 49 -
Figure 36: "Radial lines" video sample.....	- 50 -
Figure 37: The Gaussian pyramid levels.....	- 52 -
Figure 38: Two consecutive frames in level 4 of pyramid,speed $\pi/10$ -	55-
Figure 39: Two consecutive frames in level 4 of pyramid,speed $\pi/50$ -	56-

List of Tabels:

Table 1: Some mean square error examples.....	-31-
Table 2: Comparing displacement vectors in different pyramid levels -	32-

Chapter 1

Introduction

1.1 Problem Statement (Motivation)

Image analysis techniques are used to detect normal and abnormal events and different types of visual objects (person, package, car, etc.) and extract their characteristics (speed, direction, disappearance/appearance, position).

Indeed, event detection in video is becoming a more and more important application for computer vision, mainly in the context of activity recognition [3].

To achieve this goal, many approaches use local descriptors on desired points in images [4] and video [5]. These techniques are based on expressing the local region around an area of interest.

However, these techniques rely on the assumption that we can reliably detect enough stable interest points in the image or the video (image sequences). This means that for space-time interest points the video sequence should have several instances of motion critical events (regions) where the specific object changes its direction of motion (e.g. the black and white striped circle that rotate continuously with desired speed(fig. 36)).

One of the methods that can be used to locate stable interest points is the estimation of optical flow. The optical flow is the pattern of apparent 2D motion of objects in sequences of time-ordered images. In every video frame, every pixel is associated with a two dimensional vector, and this vector tells the apparent motion of that pixel, when it moves from one frame to the next [20].

In order to estimate optical flow, we face some difficulties, including:

a) Aperture problem

The homogeneous contour has ambiguous motion if the detector or motion sensor looks at it through a window or aperture that is smaller than the contour that it observes.



Figure 1: Looking to the homogeneous contour through an aperture

Within that aperture problem we cannot distinguish between, different physical motions and the motion can be towards any direction.

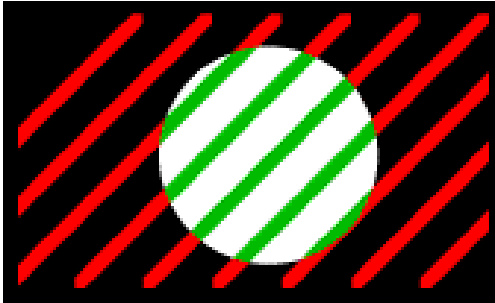


Figure 2: Lines are moving from right to left

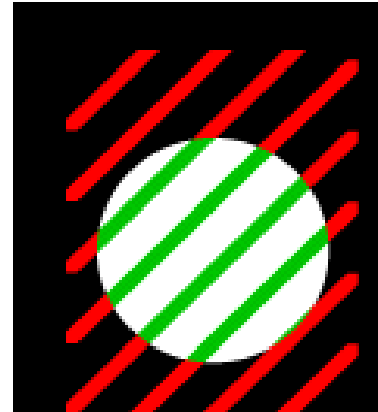


Figure 3: Lines are moving top to bottom

The motion sensor is sensitive to the part of the contour's motion that is perpendicular to the edge of the contour.

In figures 2 and 3 the lines moving top-to-bottom produce the same spatiotemporal structure as the lines moving right-to-left. As a result, the motion sensor cannot detect the correct movement of the contour unless the ends of the bars become visible in the aperture.

b) Motion of lines problem

For estimation of optical flow with full degree of freedom in a specific region, we should consider that this region should not be linearly symmetric.

The degree of freedom (DOF) for moving lines such as in figures 2 and 3 is the amount of motion perpendicular to the orientation of the line.

If the region contains corners and well distributed structure, then we have a point motion and we can estimate two degrees of freedom of motion (displacement in x and y) but on the other hand, like the figures 2 and 3 if region contains linearly symmetry then we have a line motion and we can estimate only one degree of freedom (the degree of motion perpendicular to the edges).

This method will help us to solve aperture problem that we discuss in the next chapter [20].

c) Problem of high motion

The image has a high frequency component when data is changing rapidly across a short distance and the image has a low frequency component when the pixel values are changing slowly.

The input video sample includes both high and low motion regions. Usually the estimation of optical flow for regions with high frequency or high motion (e.g. boundaries in a rotational spiral) are not accurate

because the data of those pixels in boundaries change faster than in others. We should handle this problem for all pixels in video that have high frequency.

The dense technique in optical flow algorithms is used for calculating motion vectors. It processes all the pixels and provides one flow vector per pixel. This technique is slower (in comparison with the sparse technique) [6] but it can provide more accurate results and it is fast enough to use in real-time applications.

The sparse technique processes only some pixels from the whole image, and generally executes feature tracking and it is usually used in time-critical applications.

As described above using a reliable and accurate method for estimating the geometry of each point (pixel) in an image sequence in general and desired point(s) in specific is one of the most important concerns for those working in this domain. This is also the basis for accurately detecting a desired object in video.

1.2 Approach Chosen to Solve the Problem

The optical flow of an image sequence is defined as a vector field, relating each image to the next image (each frame to the next frame).

Each vector shows the apparent displacement of each pixel from image to image. Assuming that each pixel conserves its intensity, we reach the "Brightness Constancy Constraint" equation, (discussed fully in chapter 2)

$$F(x, y, t) = F(x + dx, y + dy, t + dt) \quad (1.1)$$

In this formula F is an image sequence, dy and dx are the displacement vectors for the pixel with coordinate $[x, y]$ and finally t and dt are the frame and temporal displacement of the image sequence.

The ideas of that optical flow and brightness conservation are first proposed by Fennema [7]. One of the methods for solving the equation (1.1) is gradient-based method. This method solve the differential form of (1.1), derived by Taylor expansion.

$$\frac{\partial F}{\partial x} v_x + \frac{\partial F}{\partial y} v_y + \frac{dF}{dt} = 0 \quad (1.2)$$

Here, we have two unknown (v_x and v_y) in only one equation, for solving this equation extra constraints must be imposed.

We can categorize methods for motion estimation into three sections: the differential based methods [8], the energy based methods [9], and the correlation based methods [10].

In our thesis, we use the gradient-based optical flow algorithm proposed in [2], in order to deal with the issue of line motion, we have integrated the algorithm in multi level technique where the image is decomposed into Gaussian pyramid-set of the reduced images in order to deal with high motion issue.

For the optical flow estimation, we have equipped the PL method with multiscale computations because it allows the image to divide to the different scale of resolution in the form of gaussian pyramid. A gaussian pyramid is defined as a hierarchy of low pass filtered versions of the original image, and progressive levels correspond to lower frequencies.

The optical flow estimation for a specific pixel in the image sequence is calculated in five different pyramid levels within a program in the given work. By using these five levels, it makes a refined estimate (the best optical flow vector that has least error) that we use to detect the rigid or non-rigid object in the desired video.

We have tested this algorithm (the optical flow estimation algorithm, equipped with a five level gaussian pyramid) on a few video samples with different shapes (patterns) and speeds, which are detailed in chapters 3, 4 and 5. Results show that how multiscaling can be useful to reach an accurate estimation.

We selected MATLAB program as a suitable programming environment in which to calculate, compute and show the estimated optical flow within related error vectors for all pixels.

The implementation does not need any special hardware although c code was built as a mex-module for fast calculation of the gradient.

MATLAB is widely used in image processing, control design, test, measurement, modeling and analysis.

1.3 Limitations

There are some limitations that affect the estimation of optical flow. For example:

"No structure" problem

If an input video (image frame) includes a spatio-temporal region without structure (e.g. a plain pattern) then the optical flow cannot be estimated except by using global methods [15]. When we were making sample videos or recording video in real situations, we tried to control conditions (noise) that may have negative effect on the estimation of optical flow.

Lighting invariance

Illumination changes can harmfully affect the estimation of the optical flow.

If we assume that, due to automatic adjustments of the camera (in real time samples) or atmospheric conditions, the second image frame is brighter than the first, then as a result the brightness constancy assumption will fail since every pixel of the second frame will be brighter.

Brightness constancy is the basic assumption for the algorithm that we use for the estimation of optical flow, but one can address lightness invariance obstacles by employing methods that counteract it [22].



Figure 4: The sample frame from input video

In the figure 4, if we assume that the car is fixed but the light source goes from the left to the right of the car, then although car does not move (no motion) but we have an optical flow. When we were making our video sample, we were trying to avoid such problems.

AGC (Automatic Gain Control)

In real time cases, when we examine the webcam output, we often notice that the entire image temporarily becomes brighter to compensate for the darkening of the room. This adjustment can be seen as automatic gain control circuits in the camera. It is also result of the shutter speed adjustment of the camera, an automatic function that is beyond our control.

When these global changes occur, optical flow estimations encounter serious problems.

Detecting vehicle or human

Since there is wide range of demands or studies in the event detection domain which individually serve wide-ranging purposes, we decided to narrow the thesis work to limit our video analysis to detect moving objects in the shapes of vehicles or humans.

1.4 Thesis Goals and Contribution

The goal of this thesis work is to overcome the issues of speed (high motion) and line motion and test the method of Karlsson and Bigun [2] for motion based event detection. We ultimately plan to reach a reliable and useful optical flow estimation for each pixel in the input video. Result vectors are calculated and collected automatically from different levels of a multi-scale pyramid and the results can be used for the event detection domain. To prove this method, we show the results of detecting and distinguishing between vehicles (rigid) or humans (non-rigid).

Chapter 2

Background, Definitions and Related works

In this chapter, we provide the reader with some material on the domain of our thesis work. A description of the methods, algorithms and theoretical features that one involved in this domain will give the reader a better understanding of the current work.

2.1 Optical flow and Motion

Optical flow is the pattern of apparent 2D motion of objects in sequences of time-ordered images. Two dimensional image motions in an image plane is the projection three dimensional motion of object ($\mathbb{R}^3 \rightarrow \mathbb{R}^2$) and each image consist of many pixels that all have unique coordinates so they can be describe as a 2D vector, in every pixel of a video.

In time-ordered images that can be video frames, these vectors show motion of that pixel from image 1 to image 2 or video-frame 1 to video frame 2.

As it was mentioned above, image points (pixels) travel from one frame to another, which indicates optical flow. By definition, the apparent motion of the brightness pattern calls optical flow. In this point, it is important that motion field is not equal to optical flow because if we consider uniform sphere with existence of some shadow on the surface then if sphere rotates, shading pattern will not move at all so in this example motion field is not zero but the optical flow is zero because the processed frames are identical.

If shading pattern changes by moving the lighting source and sphere keeps fix then the motion field is zero but optical flow is not zero.

These example shows that these two subjects “optical flow” and “motion field” are not equal but one can assume that for most natural scenes they should be highly correlated.



Figure 5: The barber pole illusion

The barber pole illusion has been shown in figure 5. In fact, it is a visual illusion that shows biases in the estimating of visual motion in the observer's brain. When it rotates right or left in such a way that has been shown in figure 5 (that

top and end are hidden), the observer perceives that stripes are moving up or down (in the direction of its vertical axis).

This illusion happens because contour provides ambiguous information about its true direction of movement. This situation is referred to the aperture problem that is discussed in details in next section.

2.2 Optical flow calculation and Aperture problem

We assume an image $F(x, y, t)$ representing spatio-temporal image of moving particles, and velocity of an image pixel $\vec{V} = [v_x, v_y]^T$ moving in the (x, y) :

$$\vec{v} = \frac{ds(t)}{dt} = \left(\frac{dx(t)}{dt}, \frac{dy(t)}{dt} \right)^T = [v_x, v_y]^T \quad (2.1)$$

Where $s=(x, y)^T$ is the coordinate of a particle and t represents the time coordinates.

If we assume that the intensity of S is the same during dt , we reach equations [1.1] and [1.2] respectively and then:

$$\vec{v}^T \cdot \nabla F + \frac{\partial F}{\partial t} = 0 \quad (2.2)$$

where $\nabla F = \left[\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y} \right]^T$ is image spatial intensity gradient at pixel S and this equation is called the optical flow constraint equation and it defines a single local constrain on image motion.

But we need to find two real variables v_x and v_y with having only one constraint equation which means that we are not able to determine optical flow with only one observation.

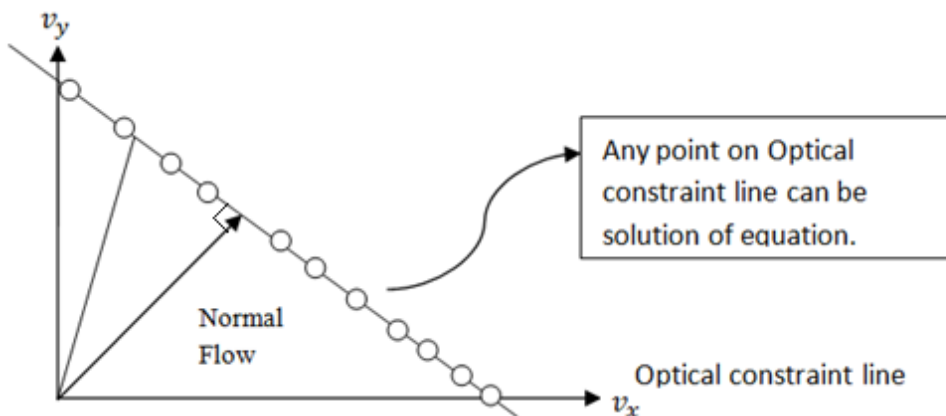


Figure 6: Geometrical explanation of the optical flow constraint equation

In figure 6 any point on the constraint line can be optical flow of given image pixel. Normal velocity is defined as perpendicular vector to the constraint line and it is smallest magnitude on the optical flow constraint line that is only velocity vector that can be estimated in the direction of local gradient of image intensity function.

As discussed in chapter1, the subject is known as “aperture problem”: we cannot determine those flows with direction perpendicular to the image gradient.

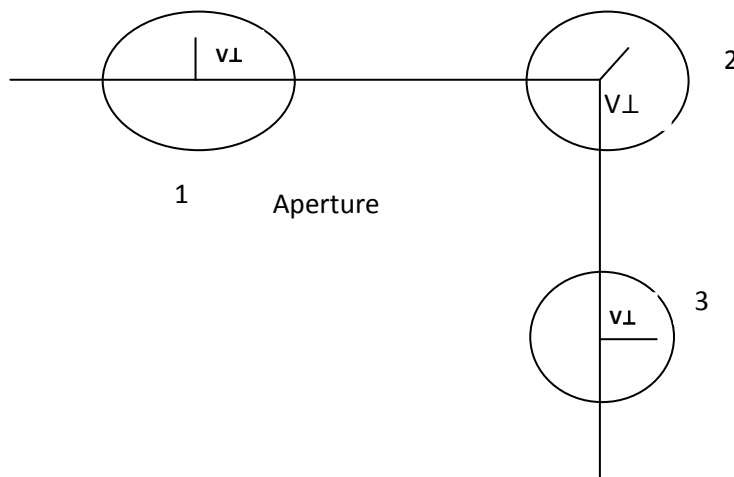


Figure 7: Apertures Problem [11]

Figure 7 shows that the vertical motion is dominant when there is a vertically elongated aperture whereas a horizontal motion is dominant when there is a horizontally elongated aperture. In aperture 1 only a motion that is orthogonal to the square's border can be estimated (vertical motion to the up) and horizontal motion of border cannot be estimated. Indeed the perceived direction of the motion (normal motion or motion orthogonal to the moving line) relates to the termination of the line's end points with border (inside aperture).

In aperture 3, based on explanation above, only a horizontal motion (normal motion) that is orthogonal to the square's border can be estimated; whereas in aperture 2 that shows the corner point, there is enough information(both vertical and horizontal square's edges) for estimating the both motion.

2.3 Optical flow methods

There are some different techniques for solving this problem that all of them try to introduce some additional condition or constraint for estimating actual flow.

Classifications of these techniques based on Beauchemin and Barron's studies [11] are as follows:

1. Intensity based differential methods
2. Frequency based methods
3. Correlation based methods
4. Multiple motion methods
5. Multiconstraint methods
6. Temporal refinement methods

Methods that are using the equation (1.1) are referred to as differential techniques.

In differential techniques image velocity is computed by spatial-temporal derivatives of image intensity. Therefore, it treats the image sequence as a continuous (differentiable) function in time and space domains. Equation (2.2) is a basic formula for computing optical flow in global and local first and second order methods. Except equation (2.2) global methods use additional global constraints and also smoothness regularization term to estimate dense optical flow for the large image regions. Normal velocity information in all of the local neighborhoods is used in Local methods to perform a minimization to find the best fit for \mathbf{v} .

A contour or surface model can also be used to integrate normal velocities (that we have) into full velocity and discontinuous optical flow can be analyzed by parametric models, line process or mixed velocity distribution. These techniques which mentioned above do segmentation of optical flow into the region that is corresponding to different independently moving surfaces or objects.

In this thesis we have chosen differential methods for estimating optical flow but differential methods also have some different subsets:

- Global methods
- Local models
- Surface models
- Contour models
- Multiconstraint methods

Two well-known differential methods are the Lucas-Kanade and Horn-Schunck techniques.

Local techniques for e.g. the Lucas and Kanade method [12] involve the optimization of a local energy functional or the frequency based minimization methods which can be found in [13] and [14].

The global category as in Horn and Schunck [15] refers to methods that determine optical flow by minimizing of a global energy functional.

Differential techniques are used widely because of their high level of performance [16].

Each one of global or local methods has their own advantages and disadvantages.

Global techniques provide dense flow fields with the power to analyze "no structure" regions, but as a disadvantage they have a much larger sensitivity to noise [16] and, on the other hand, local methods offer robustness to noise.

There is also an approach by N.Bauer, P.Pathirana and P.Hodgson that involves developing neighborhood selection for combined Horn-Schunck / Lucas-Kanade robust optical flow [17].

Now we will explain two well-known differential techniques and we will also clarify the techniques that we have chosen to do our thesis after them.

2.3.1 Local model and Lucas and Kanade method

One of the classical approaches for estimation of the optical flow was suggested by Lucas and Kanade (LK). Originally, they did not formulate an analysis of spatio-temporal volumes, but rather considered template matching/registration between two images [12].

In this model we assume that we can use a constant model for estimation of optical flow in small window Ω that Ω is spatial neighborhood, then we define window function $w(s) > 0 \in \Omega$.

The weighted least square solution for optical flow constraint equation (Eq. 2.2) and velocity is computed by minimizing:

$$\begin{aligned} \text{Min } E &= \sum_{s \in \Omega} w^2(s) \left(\mathbf{v}^T \cdot \nabla F + \frac{\partial F}{\partial t} \right)^2 & \text{or} \\ E &= \iint [(F_x v_x + F_y v_y + F_t)^2] dx dy \end{aligned} \quad (2.3)$$

If we consider N points in an image (with the size of neighborhood Ω) at t_0 then we would have these equations:

$$\begin{pmatrix} \frac{\partial F(x_1, y_1, t_0)}{\partial t} \\ \frac{\partial F(x_2, y_2, t_0)}{\partial t} \\ \vdots \\ \frac{\partial F(x_N, y_N, t_0)}{\partial t} \end{pmatrix} = - \begin{pmatrix} \frac{\partial F(x_1, y_1, t_0)}{\partial x} & \frac{\partial F(x_1, y_1, t_0)}{\partial y} \\ \frac{\partial F(x_2, y_2, t_0)}{\partial x} & \frac{\partial F(x_2, y_2, t_0)}{\partial y} \\ \vdots & \vdots \\ \frac{\partial F(x_N, y_N, t_0)}{\partial x} & \frac{\partial F(x_N, y_N, t_0)}{\partial y} \end{pmatrix} \begin{pmatrix} v_x \\ v_y \end{pmatrix} \quad (2.4)$$

For $N > 2$, this is an over determined system of linear equations of the form:

$$\mathbf{d} = -\mathbf{D}\mathbf{v} \quad (2.5)$$

That \mathbf{v} is unknown and

$$\mathbf{d} = \begin{pmatrix} \frac{\partial F(x_1, y_1, t_0)}{\partial t} \\ \frac{\partial F(x_2, y_2, t_0)}{\partial t} \\ \vdots \\ \frac{\partial F(x_N, y_N, t_0)}{\partial t} \end{pmatrix} \quad (2.6)$$

and

$$D = \begin{pmatrix} \frac{\partial F(x_1, y_1, t_0)}{\partial x} & \frac{\partial F(x_1, y_1, t_0)}{\partial y} \\ \frac{\partial F(x_2, y_2, t_0)}{\partial x} & \frac{\partial F(x_2, y_2, t_0)}{\partial y} \\ \vdots & \vdots \\ \frac{\partial F(x_N, y_N, t_0)}{\partial x} & \frac{\partial F(x_N, y_N, t_0)}{\partial y} \end{pmatrix} \quad (2.7)$$

are to be assumed known. The equation (2.5) is solved by multiplying two side of equation with D^T (least square error method) for the 2×2 system of equations for the unknown \mathbf{v} (velocity) so:

$$D^T d = -D^T D \mathbf{v} \quad (2.8)$$

This equation has solution if

$$S = D^T D = \sum \nabla F (\nabla F)^T \quad (2.9)$$

S is called structure tensor matrix and it looks like this:

$$S_{2D} = \begin{pmatrix} m_{20} & m_{11} \\ m_{11} & m_{02} \end{pmatrix} \quad (2.10)$$

The elements of S are called “Spectral moments” and defined as:

$$m_{ijk} = \sum w F_x^i F_y^j F_t^k \quad (2.11)$$

Eigenvalues of the 2D structure tensor (matrix S) define the amount of linear symmetry [1] (a region where gradients are linearly dependent is called linearly symmetric and it is impossible to estimate flow locally except for one component: that of the aligned direction (aperture problem)).

If the window is centered at \mathbf{x} we can write:

$$S_{2D}(\vec{X}) = \begin{pmatrix} m_{20}(\vec{X}) & m_{11}(\vec{X}) \\ m_{11}(\vec{X}) & m_{02}(\vec{X}) \end{pmatrix} \quad (2.12)$$

With this definition of matrix S we can rewrite equation (2.8) like:

$$\begin{pmatrix} m_{101}(\vec{X}) \\ m_{011}(\vec{X}) \end{pmatrix} = - \begin{pmatrix} m_{20}(\vec{X}) & m_{11}(\vec{X}) \\ m_{11}(\vec{X}) & m_{02}(\vec{X}) \end{pmatrix} \vec{v}(\vec{X}) \quad (2.13)$$

if

$$\vec{b}(\vec{X}) = \begin{pmatrix} m_{101}(\vec{X}) \\ m_{011}(\vec{X}) \end{pmatrix} \quad (2.14)$$

then we would have:

$$\vec{b} = -S_{2D}\vec{v}(\vec{x}) \quad \Rightarrow \quad \vec{v}(\vec{x}) = -S_{2D}^{-1}(\vec{b}) \quad (2.15)$$

So the solution is optical flow for the image pixel “s” and we should consider that the reliability of the estimation of v is exposed by the eigenvalues of matrix S_{2D} . Let us assume λ_1 and λ_2 are eigenvalues so if both of them be large then the flow can be determined uniquely. If λ_1 is zero, but λ_2 is large, then it is a linearly symmetry case and only motion of lines can be determined. If both $\lambda_1 = \lambda_2 = 0$, then no motion can be inferred.

The LK algorithm needs regularization (for e.g. Tikhonov regularization) to become stable in the outside of any region except that of point motion, but this adds the difficulty of choosing the regularization parameter.

Since Lucas-Kanade's technique employs a local window to determine the optical flow of a specific image point, this is the cause that it is called a local method. Indeed in Lucas-Kanade's technique, flow of points calculated by finding the intersection of all the flow constraint lines (fig.6) that are corresponding to the image pixels which are in the window of “w”. Those lines will have an intersection, since Lucas-Kanade's technique assume that flow in the window is constant.

2.3.2 Structure Tensor in 3D

As we explained above, for 2D Structure Tensor we have:

$$S_{2D}(\vec{x}) = \begin{pmatrix} m_{20}(\vec{x}) & m_{11}(\vec{x}) \\ m_{11}(\vec{x}) & m_{02}(\vec{x}) \end{pmatrix}$$

Here we choose to express the 2D structure tensor in 3D spectral moments, so 3D structure tensor is defined as:

$$S_{3D}(\vec{x}) = \begin{pmatrix} m_{200}(\vec{x}) & m_{110}(\vec{x}) & m_{101}(\vec{x}) \\ m_{110}(\vec{x}) & m_{020}(\vec{x}) & m_{011}(\vec{x}) \\ m_{101}(\vec{x}) & m_{011}(\vec{x}) & m_{002}(\vec{x}) \end{pmatrix} \quad (2.16)$$

The 3D tensor can be used to estimate optical flow directly by considering its eigensystem: $\{\vec{v}_i, \lambda_i\}$ of $\lambda_1 \leq \lambda_2 \leq \lambda_3$.

As a local algorithm for optical flow estimation, the structure tensor is advantageous because it easily differentiates between the three important cases of motion of points (distributed structure), motion of lines and presence of higher order terms (higher order motions and/or uncorrelated noise).

2.3.3 Point, Line (PL) Method

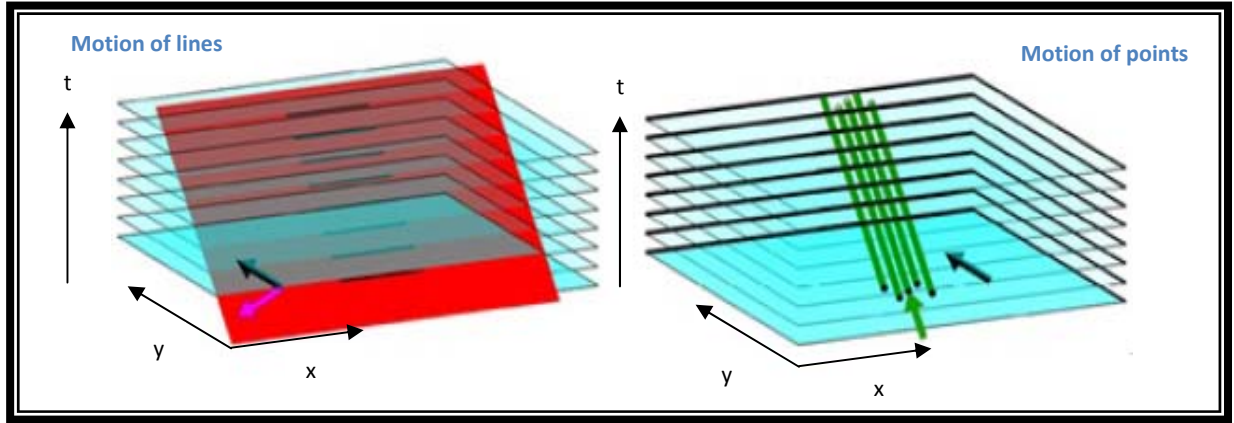


Figure 8: The uniform motion of a set of lines (left) and points (right) [1]

Intuitively, S_{3D} describes the spatio-temporal volume around a region.

The two basic translations types are Point motion and Line motion. The uniform motion of lines in figure 8 left generates isosurfaces (the set of points for which $f(x, y, t) = \text{constant}$.) which are as same as a tilted plane. If the motion plane is more tilted, then the motion is perceived faster. In figure 8 right the uniform motion of points generates a line in the 3D continuous image indeed in image plane when some points translate upwards, the result is many parallel lines.

As described before (section 2.3.1) we use window Ω for local neighborhood in Lucas and Kanada method. For every Ω that we choose (indexed by \vec{x}) eigenvectors of 3D structure tensor matrix denote as:

$\{\vec{v}_i, \lambda_i\}$ of $\lambda_1 \leq \lambda_2 \leq \lambda_3$. Where \vec{v}_{ix} is the x component of the i^{th} eigenvector, and $\gamma \in [0, 1)$ is a threshold then we would have one of these conditions [1]:

If $\lambda_1 + \lambda_2 + \lambda_3 = 0$ then Ω contains no structure (it's close to constant gray value)

1) If $\frac{\lambda_3 - \lambda_2}{\lambda_3 + \lambda_2} > \gamma$ then Ω contains line motion given by:

$$\vec{u}_l = \frac{v_{3t}}{v_{3x}^2 + v_{3y}^2} \begin{pmatrix} v_{3x} \\ v_{3y} \end{pmatrix} \quad (2.17)$$

2) Else if $\frac{\lambda_2 - \lambda_1}{\lambda_2 + \lambda_1} > \gamma$ then Ω contains point motion given by:

$$\vec{u}_p = \frac{1}{v_{1t}} \begin{pmatrix} v_{1x} \\ v_{1y} \end{pmatrix} \quad (2.18)$$

$$3) \text{ Else if } (1 - \frac{2\lambda_2}{\lambda_2+\lambda_1} + \frac{2\lambda_2}{\lambda_2+\lambda_3}) > \gamma \text{ then } \Omega \text{ contains higher order terms or noise} \quad (2.19)$$

The usage of implementation of 3D structure tensor for image processing applications is to compute the optical flow or to detect local 3D structures and their directions.

The 3D structure tensor algorithm (3D-STA) in compare to LK method has one extra moment m_{002} and eigenvalue analysis of local 3*3 matrices; it consists of three main parts: gradient, tensor and smoothing. This algorithm is computationally more expensive than LK method due to many multiplications and additions which are required to calculate the gradient, the tensor and to smooth every pixel of the image.

The Point-line method is the combination of LK and 3D Structure tensor methods for the aim of using beneficial properties of both.

If the region of interest for estimation of optical flow is linear symmetric then S_{2D} becomes singular and equation (2-13) becomes undefined. In this case we desire to estimate motion of line patterns. However we don't want to resort to the full eigenvalue analysis of equation (2-16)

Instead, we note that the normal flow at a single point is given by:

$$\vec{p}_n = \frac{\nabla F \nabla^T F}{\nabla^T F \nabla F} \vec{u} = - \frac{\nabla F}{|\nabla F|^2} F_t \quad (2.20)$$

If region is linearly symmetric then by averaging normal flow over the region we can estimate line flow and with use of smoothing window w , we reach following expression:

$$\vec{u}_n(\vec{x}) = \sum w \vec{p}_n = - \frac{\vec{b}(\vec{x})}{Tr(S_{2D}(\vec{x}))} \quad (2.21)$$

The two methods Eq.(2-15) and Eq.(2-21) have distinct domain of good performance which is determined by α and it is the reason for using weighting function $w(\alpha) \in [0,1]$:

$$\vec{u}_l = w(\alpha) \vec{u}_n \quad (2.22)$$

$$\vec{u}_p = (1 - w(\alpha)) \vec{u}_{LK} \quad (2.23)$$

$$\vec{u}_w = \vec{u}_l + \vec{u}_p$$

We call the \vec{u}_p and \vec{u}_l point flow and line flow respectively. Consider to this point that the normal flow and line flow are two different issues and readers should not be confused. Normal flow is defined for every pixel in the image but the line flow is the average of normal flow if and only if the region is linearly symmetric.

There are many possibilities for a weighting function. For example, if we use ($w = \alpha > \gamma$) for some threshold γ that is called using the brick wall

thresholding function then the output result is near the result of structure tensor method, concerning the differentiation between point and line flow. However, using the following weighting function has promising results [2]

$$w = \alpha^2 = 1 - 4 \frac{|S(\vec{x})|}{\text{Tr}(G(\vec{x}))^2} \quad (2.24)$$

And by substituting equations (2.22) and (2.23) with (2.24) get:

$$\vec{u}_l = - \frac{(m_{200} - m_{020})^2 + 4m_{110}^2}{(m_{200} + m_{020})^3} \begin{pmatrix} m_{011} \\ m_{101} \end{pmatrix} \quad (2.25)$$

$$\vec{u}_p = \frac{4}{(m_{200} + m_{020})^2} \begin{pmatrix} m_{101}m_{110} - m_{011}m_{200} \\ m_{011}m_{110} - m_{101}m_{020} \end{pmatrix} \quad (2.26)$$

The combination of point and line flow by using the weighting function of (2.24) can be seen as a local regularization of the Lukas and Kanade method. Equation (2.15) in Lk method is ill-posed for the case of linear symmetry. It would have more stable result by applying a diagonal Tikhonov regularization (in PL method means adding a small positive value to m_{200} and m_{020}) but it still has the problem of separating the point flow from the line flow, this also causes the another new problem of how to define the regularizing parameter.

In this thesis because, the consistency of estimation and computational efficiency are in focus we choose to derive a local method so we use Point-Line method and for solving the problem of large displacements in the sequence (high motions) we use a gaussian pyramid for processing images in multi scale pyramid that details are presented in next section.

2.3.4 Global model and Horn-Schunck method

The method assumes smoothness in the flow over the whole image and it introduces a global smoothness constraint. Indeed Horn and Schunck method assumes that the following quantity should be low in the image vary smoothly everywhere.

So we have:

$$\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 = \|\nabla u\|^2 + \|\nabla v\|^2 \quad (2.27)$$

Thus, it tries to minimize distortions in flow and prefers solutions which show more smoothness.

We can determine the optical flow velocity v by minimizing the squared error quantity of the brightness equation (2.2) and smoothness constraint (2.28).

Then the error to be minimized is:

$$E = \iint [(F_x u + F_y v + F_t)^2 + \alpha^2 (\|\nabla u\|^2 + \|\nabla v\|^2)] dx dy \quad (2.28)$$

In equation (2.28), the parameter α is regularization constant and identifies the influence of smoothness constraint on the optimization (larger values of α lead to a smoother flow).

Indeed, α^2 is the weighting term and Horn and Schunck chose the magnitude of α^2 to be almost proportional to the estimated noise in $F_x^2 + F_y^2$.

The formula (2.28) can be minimized by using calculus of variations approach [11] yielding a set of differential equations (Euler-Lagrange equations).

$$\frac{\partial L}{\partial u} - \frac{\partial}{\partial x} \frac{\partial L}{\partial u_x} - \frac{\partial}{\partial y} \frac{\partial L}{\partial u_y} = 0 \quad (2.29)$$

$$\frac{\partial L}{\partial v} - \frac{\partial}{\partial x} \frac{\partial L}{\partial v_x} - \frac{\partial}{\partial y} \frac{\partial L}{\partial v_y} = 0 \quad (2.30)$$

These equations can be solved by using iterative procedures. This means that it is in contrast with local methods (such as Lucas and Kanade) which can be estimated with single pass approaches.

2.3.5 Shi and Tomasi Feature Tracking

The assumption of brightness constancy is essential and basic to the successful implementation of gradient or correlation-based optical flow estimation algorithm. Many feature tracking algorithms need reliable optical flow estimation.

One of gradient based methods, as has been explained above, is Lucas – Kanade which has been used in the feature tracking schema that introduced by Shi and Tomasi [18].

The algorithm uses Lucas-Kanada as feature windows that can be selected based on some measure of texturedness or corneriness, such as high standard deviation in the spatial intensity profile, the presence of zero crossings of the Laplacian of the image intensity and corners [18].

It is clear that good features includes those with big spatial gradients in two orthogonal directions and since Lucas-Kanade method solves optical flow equations assuming the displacement is characterized by constant velocity and S_{2D} (two by two spatial gradient matrix) is used to calculate the quality of corner points where the gradients are summed across $n \times n$ blocks [19].

Tomasi suggests that reasonable criterion for feature selection is for examining the minimum eigenvalue of each S_{2D} (2 by 2 spatial gradient matrix) to be no less than some λ , and features are tracked using a Newton-Raphson method of minimizing the difference between the two windows.

This guarantees that the matrix S_{2D} is above the noise level of the image and well conditioned so that its inverse does not increase critical direction, i.e., there are enough spatial gradients in two orthogonal directions. When this features fulfilled, the Lucas and Kanade algorithm can be applied.

2.4 Gaussian Pyramid of Gradients

A general way of solving optical flow equation (2.2) is to apply optical flow techniques in a hierarchical coarse-fine framework. It allows the image to divide into different scale of resolution in the form of Gaussian pyramid. Indeed Gaussian pyramid is defined as hierarchies of low pass filtered versions of the original image, and progressive levels correspond to lower frequencies.

Reducing image size is done although there is a fearing of losing information because of the effectuated low pass filtering but the low pass filtering is executed by using convolution with either a gaussian filter or similar.

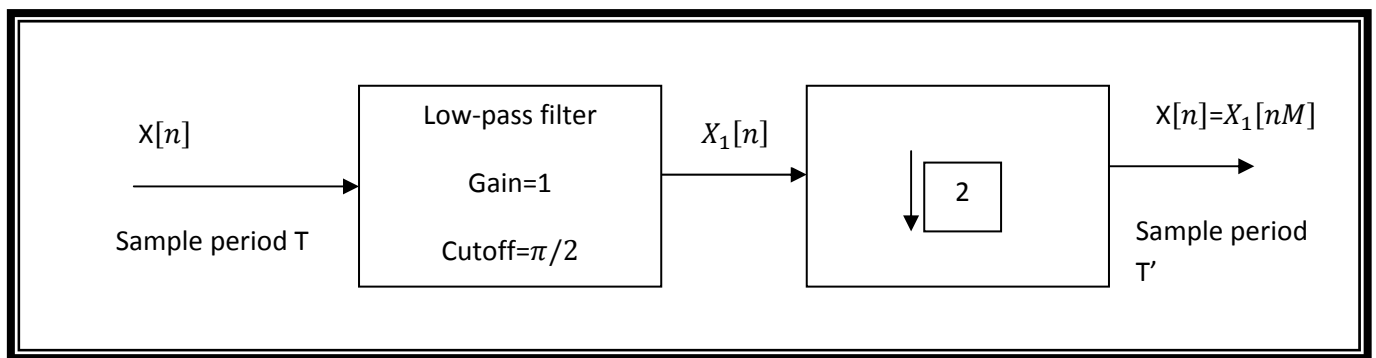


Figure 9: Down sampling by an integer factor $M=2$

In this thesis we applied reduction rate 2 on image that is taken from video as a frame for example for an image I of size 512×512 , the pyramid image levels will be:

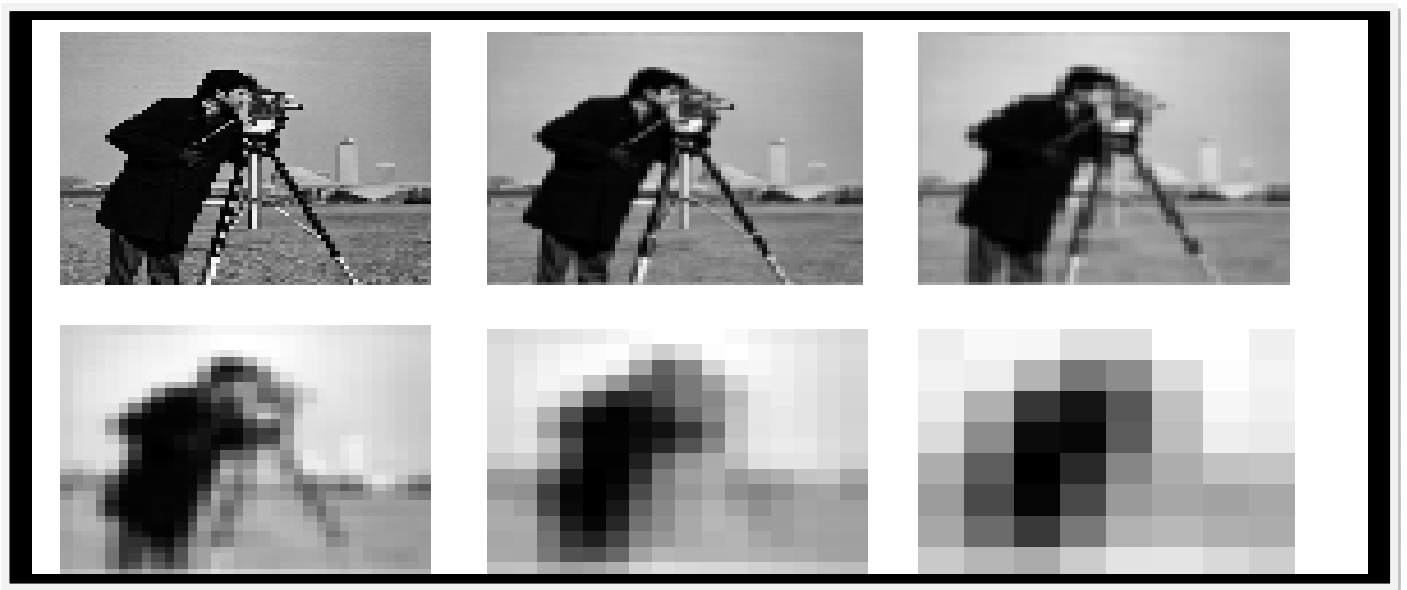


Figure 10: Gaussian pyramid levels for image that first level has taken from video size 512×512 as a sample frame with reduction rate 2.

The sample image which is corresponding to each gaussian pyramid level is shown on figure10. The largest image of the gaussian pyramid contains all frequencies that are marked as level 0 of the pyramid.

Every single level is produced by low pass filtering the level below and retaining every second row and column of the result.

In this thesis `c` code was built as a mex-module for fast calculation of the gradient fields based on extending of the open source implementation of [20].

In the next chapter we use this multi-scale algorithm to estimate optical flow vectors.

Chapter 3

Estimation of optical flow vectors

3.1 Estimated Optical flow vectors for each pyramids' level

We make a specific video sample that includes both high motion (around boundaries) and low motion (center) sections, for estimation of optical flow in different pyramid levels to find out suitable pyramid level for each section of image(depends on their rotation speed) and we show the result before using it in our final video for detecting an object. Details of sample video such as speed factor, radius of spiral and desired program code exist in appendix 1.

All the final estimated results (estimated optical flow for each level) are shown on the original image sample so the image in the background has no relation with the level of pyramid.

Notice that the length of vectors in below figures have amplified for visualization purposes, the true motion has smaller length vector.

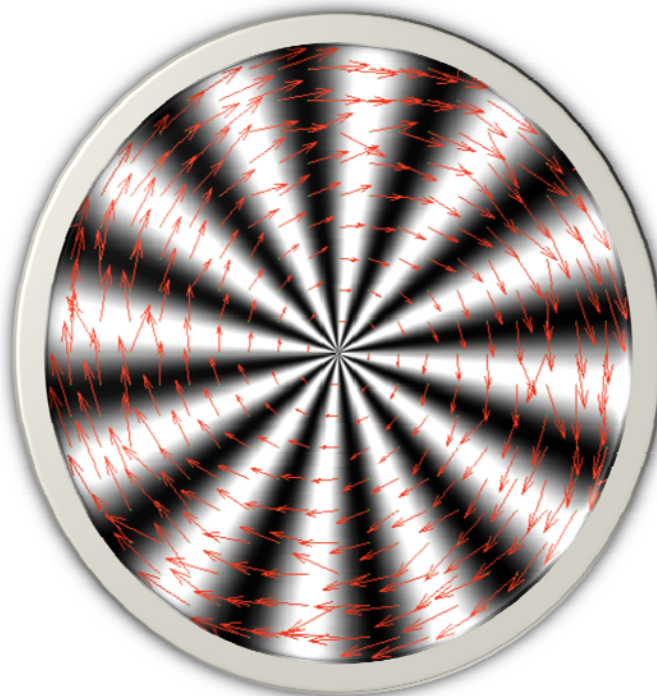


Figure 11: Estimated optical flow for the black and white striped circle video for pyramid Level 0, 510*510 pixels

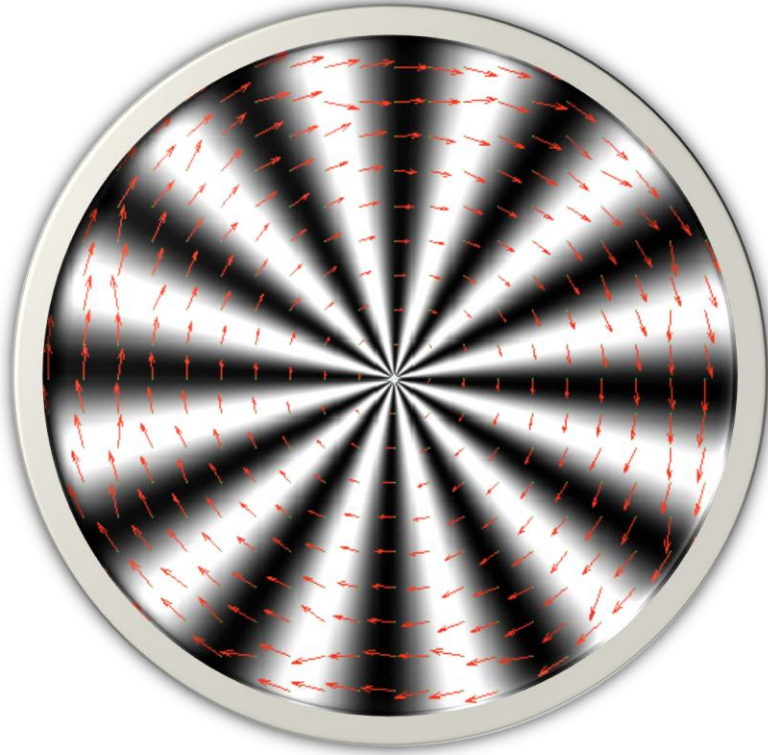


Figure 12: Estimated optical flow for the black and white striped circle video for pyramid Level 1, 253*253 pixels

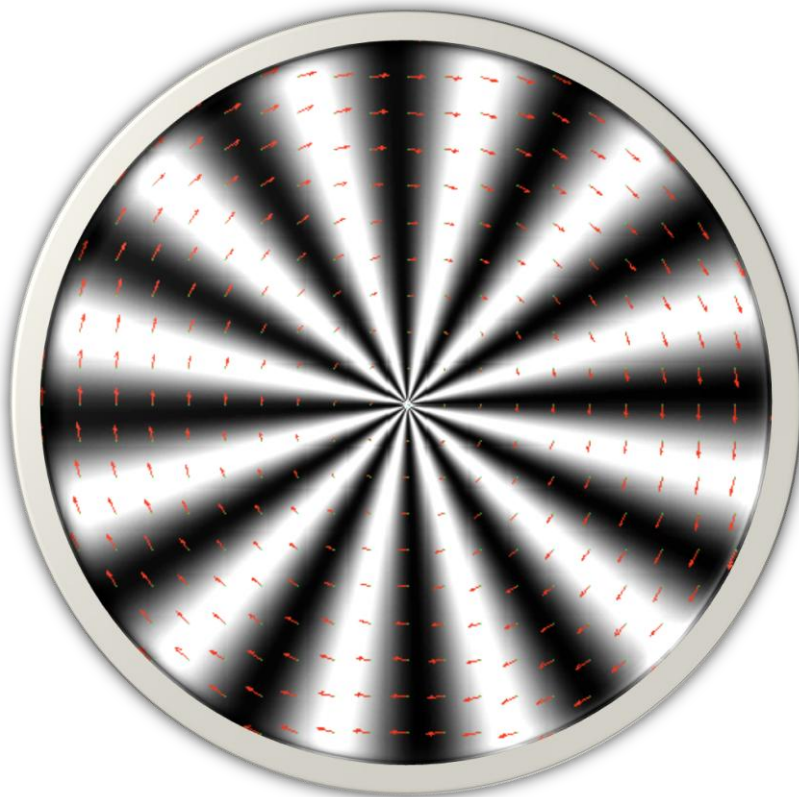


Figure 13: Estimated optical flow for the black and white striped circle video for pyramid Level 2, 125*125 pixels

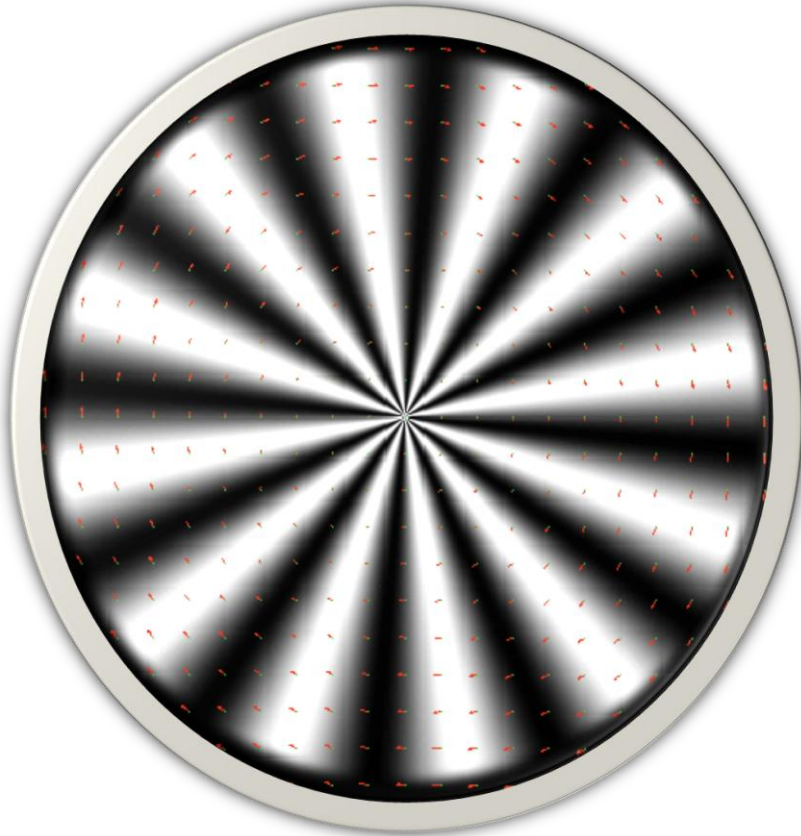


Figure 14: Estimated optical flow for the black and white striped circle video for pyramid Level 3, 61*61 pixels

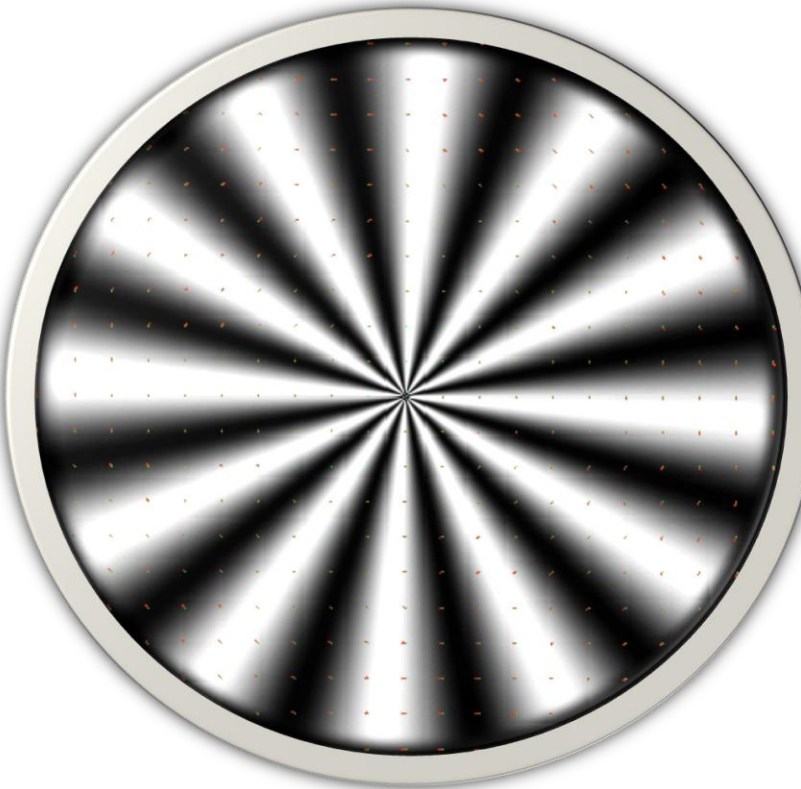


Figure 15: Estimated optical flow for the black and white striped circle video for pyramid Level 4, 29*29 pixels

Now for estimating how accurate results are, we calculate error vectors (displacement vector) for each pixel in each pyramids' level.

Given the motion speed as a known variable, we can estimate position of each point along axis x and y and save them in matrix UG , VG (blue vectors in figure16) and with subtraction of U , V (they are matrixes that include estimated position by optical flow, red vectors in figure16) we can reach to error vectors (displacement vector) and show them on our video sample. (Code exists in section 2 of appendix and figures exist on next chapter)

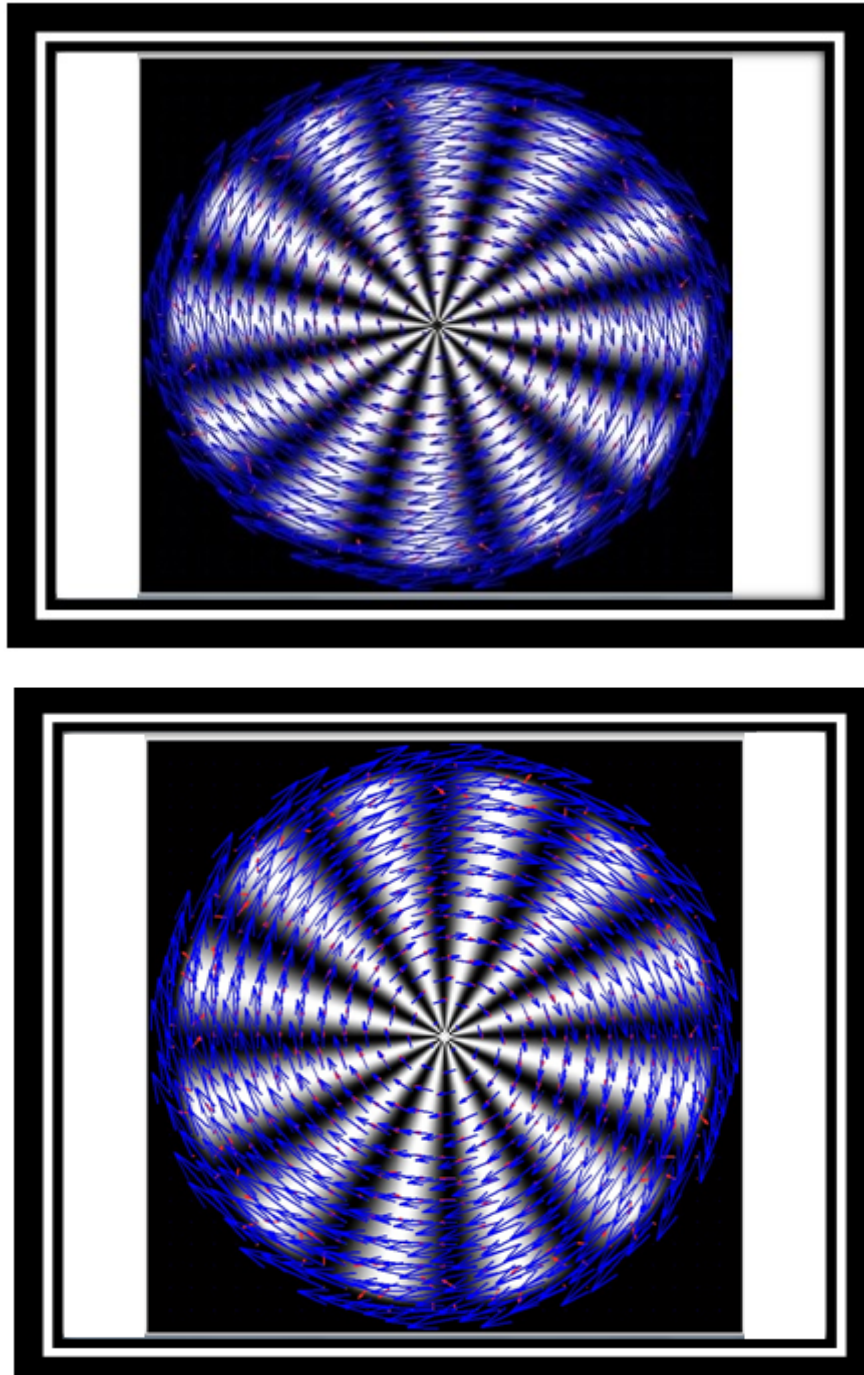


Figure 16: Two consecutive frames in Level 2, more figures are in appendix

3.2 Mean Squared Error (MSE)

We calculate error vectors (displacement vectors) for each pixel that moves from image (video frame) I to image (video frame) $I + 1$ simultaneously we estimate the optical flow vectors and at the end we calculate mean squared error (MSE).

MSE measures the average of the squares of the errors and indeed, it is one of the ways to quantify the difference between values implied by an estimator and the true values of the quantity being estimated (see section 2.1 of appendix for related code and pictures). The error vectors (blue vectors) are shown in one frame of each pyramid level as a sample; magnitude of each vector shows how big it is:

Level 0: Image size is: [510 510]

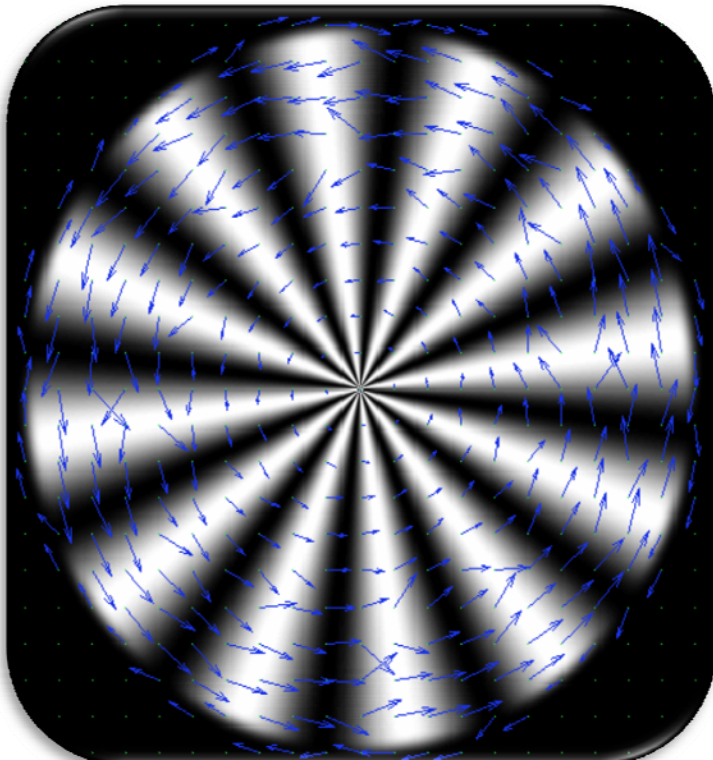


Figure 17: Error vector, Pyramid Level 0

Figure 17 has been taken from an input video in pyramid level 0, resolution is 510*510 pixel, blue vectors are error vectors that are small in center and become bigger away from center (on boundaries). We expect that find best pyramid level for the sample video that has specific rotation speed (see appendix 1 and 2.1). In general those parts that have more rotational speed should have lower errors in higher levels and vice versa. However we consider MSE that relates to the whole picture and not specific part of it. The results for this specific video samples have shown in table 1.

Level 1: Image size is: [253 253]

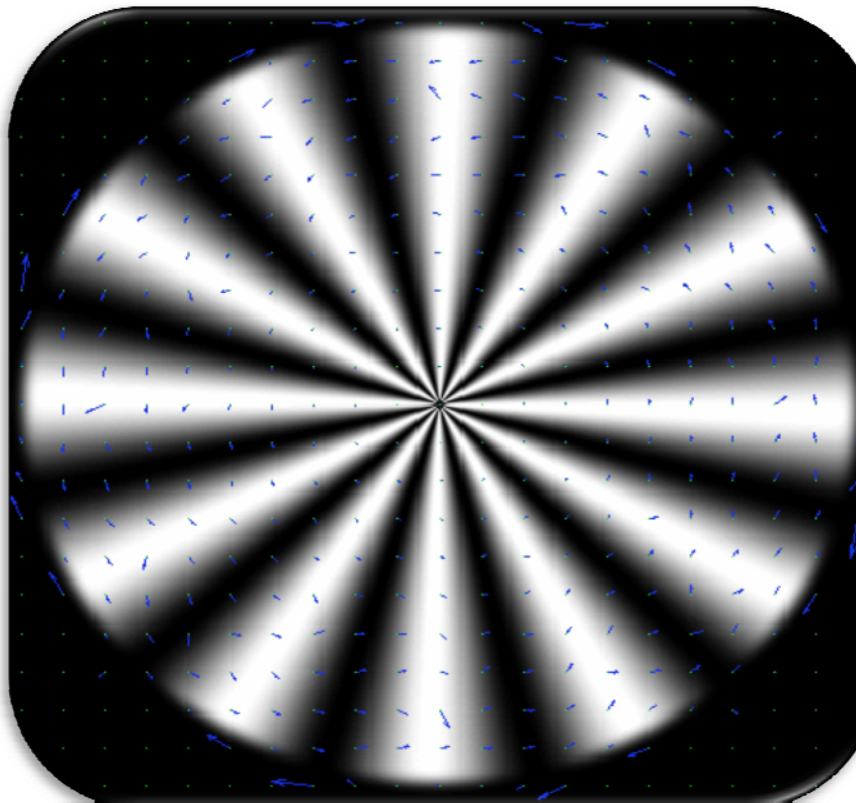


Figure 18: Error vector, Pyramid Level 1

Figure18 has been taken from an input video in pyramid level 1, resolution is 253*253 pixel, the blue vectors are error vectors and MSE numbers show that they are smaller than error vectors in level 0 as it clear in some parts of picture.

Level 2: Image size is [61 61]

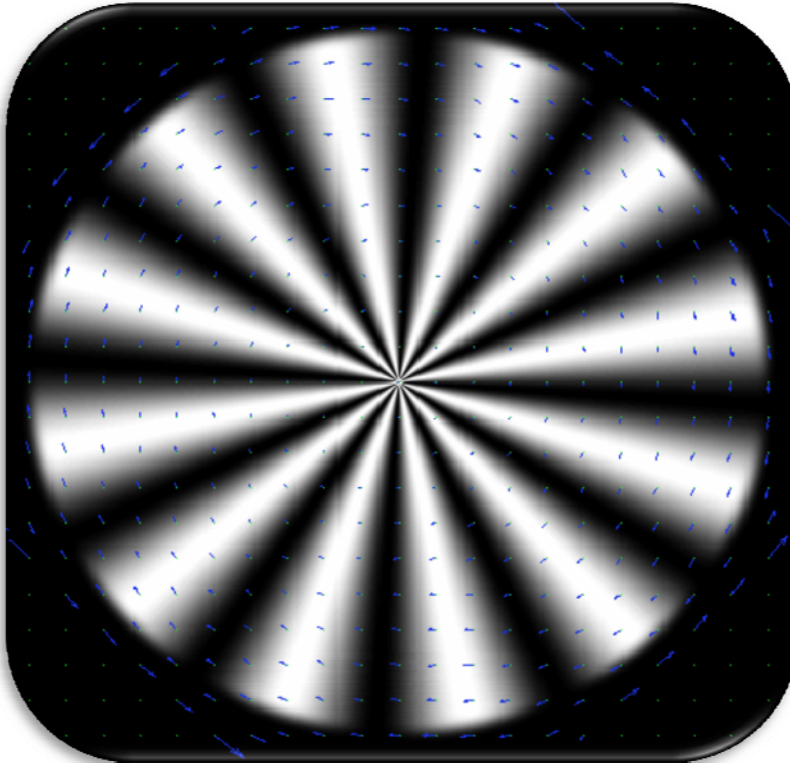


Figure 19: Error vector, Pyramid Level 2

Figure19 has been taken from an input video in pyramid level 2. Resolution is 125*125 pixel and as same as before blue vectors are error vectors and the magnetude of each vectore shows how big it is.

Level 3: Image size is [61 61]

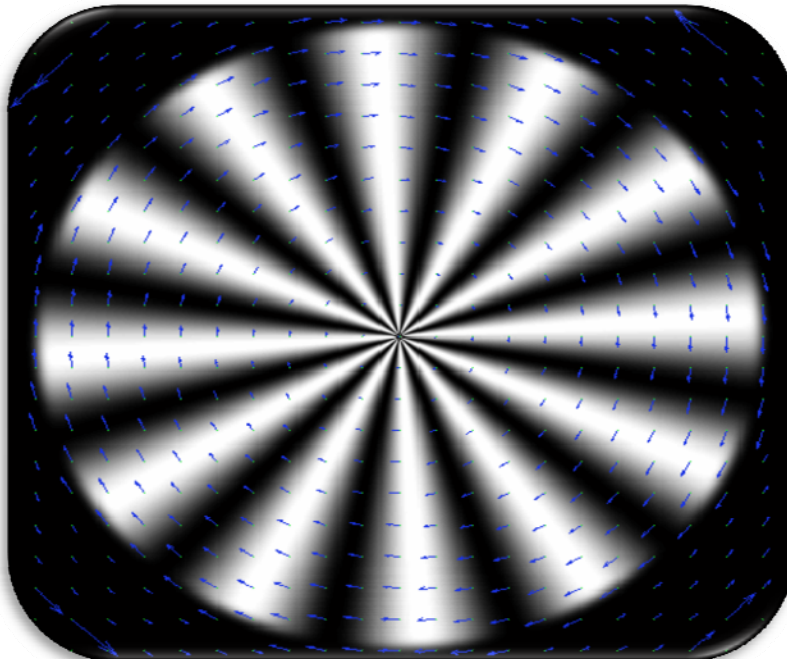


Figure 20: Image has been taken from an input video in pyramid level 3; resolution is 61*61 pixels

Level 4: Image size is [29 29]

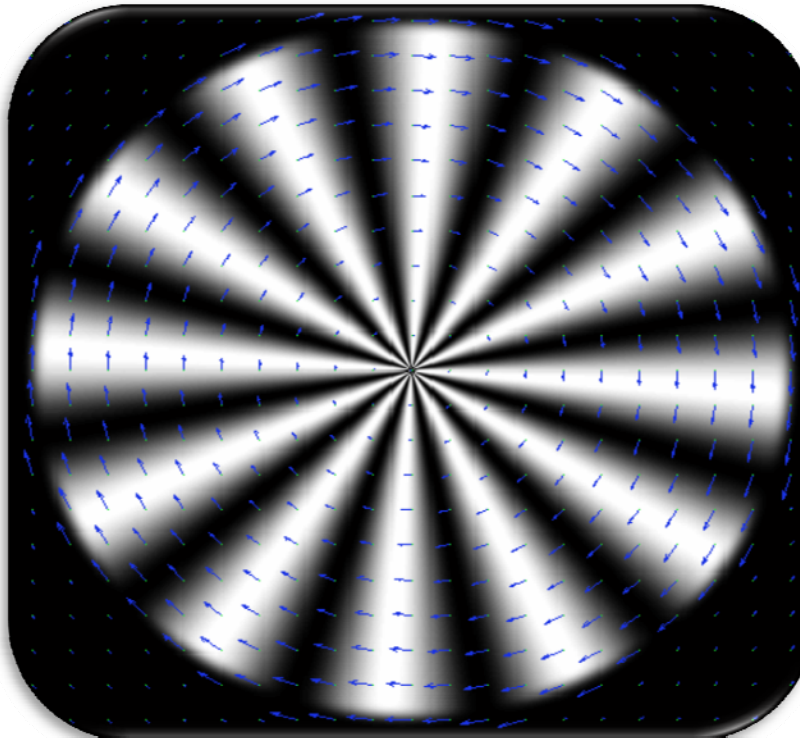


Figure 21: Image has been taken from an input video in pyramid level (the latest level), resolution is 29*29 pixel.

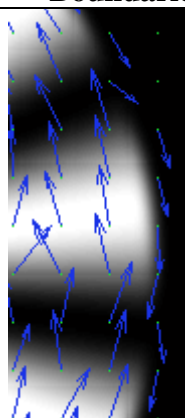
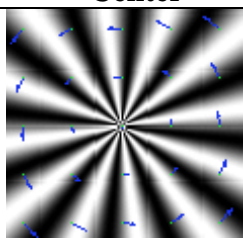
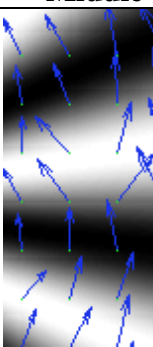
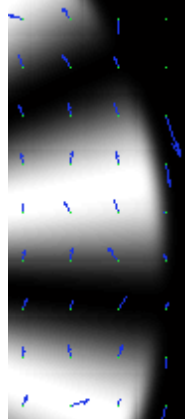
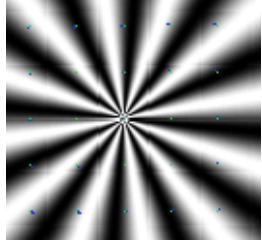
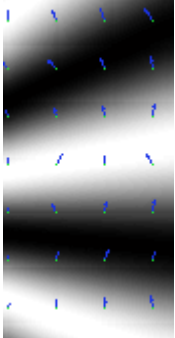
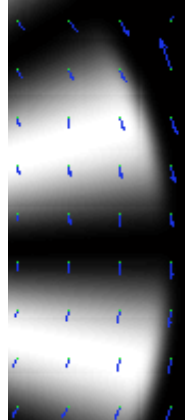
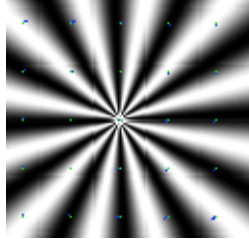
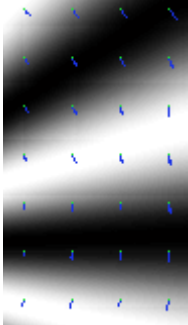
Comparing mean square error in different pyramid levels:

		Level				
		0	1	2	3	4
Frame No.	1	2.8639	0.22361	0.39086	0.67654	1.2009
	2	2.8341	0.22097	0.20847	0.68517	1.1708
	3	2.7992	0.18931	0.36031	0.67936	1.0845
	4	2.7893	0.20863	0.86077	0.77719	0.9144
	5	2.7319	0.19628	0.40873	0.61007	0.81587
	6	2.7624	0.19664	0.37363	0.68144	0.81528
	7	2.7007	0.1914	0.20894	0.63658	0.81653
	8	2.7217	0.19116	0.22075	2.2186	0.81594
	9	2.676	0.17443	0.21918	0.70371	0.81592
	10	2.6882	0.19053	0.23887	0.69214	0.81599
	11	2.7002	0.18059	0.1918	0.71346	0.81648
	12	2.8983	0.20237	0.19962	2.1585	0.81525
	13	2.8949	0.21154	0.22257	1.7901	0.81675
	14	2.7514	0.18899	0.26669	0.66456	0.81636
	15	2.8597	0.20594	0.20383	0.72964	0.81620

Table 1: Some mean square error e.g.

In table 2 we compare error vectors magnitude in different pyramid levels in three important regions: boundaries that have high motion (high frequency), middle parts and center sections that have lower motion speed (low frequency). Consider to this point that error vectors have been shown on the original image sample (as a back ground image) and that's why all pyramid levels that have different resolution have same image this help us to have better overview and clear distinguished sections.

3.3 Comparing of Error Vectors in Different Pyramid Levels

Level	MSE	Boundaries	Center	Middle
0	2.77809			
1	0.19816			
2	0.30500			

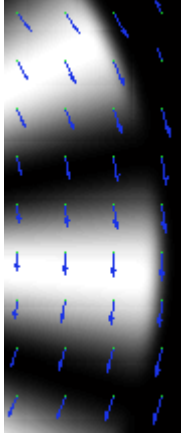
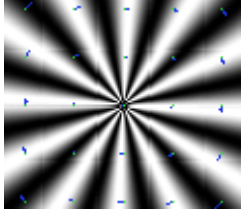
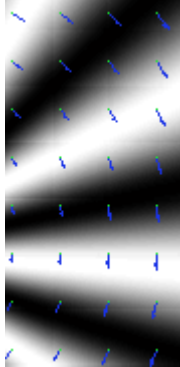
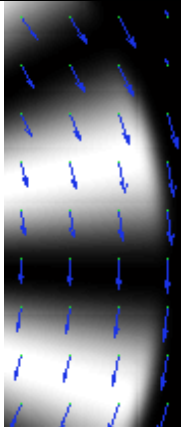
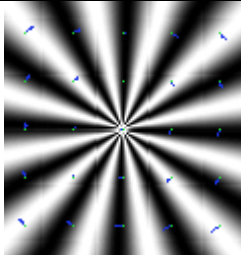
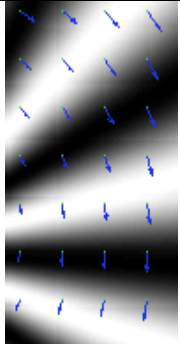
3	0.86548			
4	0.88972			

Table 2: Comparing displacement vectors in different pyramid levels for all pixel

By considering each specific picture section, we understand that the magnitude of error vectors (length) has a relation with the position of each point in frame and chosen pyramid level for estimation of optical flow. In center, because of lower rotational speed there is lower motion and as a result more stability and as an opposite in boundaries rotational speed is bigger and there is high motion so as a results it is more unstable so we reach to this conclusion that high pyramid levels has the good results for high motion video frames and low pyramid levels for low motion respectively but we consider the MSE in our algorithm to decide which pyramid level should be used for optical flow estimation.

So in the next step we explain how we can use these results that have been shown in table2.

Chapter 4

Optical flow algorithm's results

4.1 Capturing and Collecting best optical flow vectors

In the previous section we explained displacement error vectors and estimated optical flow vectors, and we showed the corresponding results.

Now, by writing programming code script, we plan to create an output video that is constructed by using the best optical flow vectors of each pyramid level.

Our sample video is the same as other desired videos that can be chosen as input videos, in that it is composed of different sections with differing motion speeds. One of the main reasons for pyramidal representation is that it helps us to handle large pixel motions. Thus the pyramid height (level) has to be picked properly according to the maximum optical flow that we expect in each section of the image (for e.g. in the center, middle and boundary).

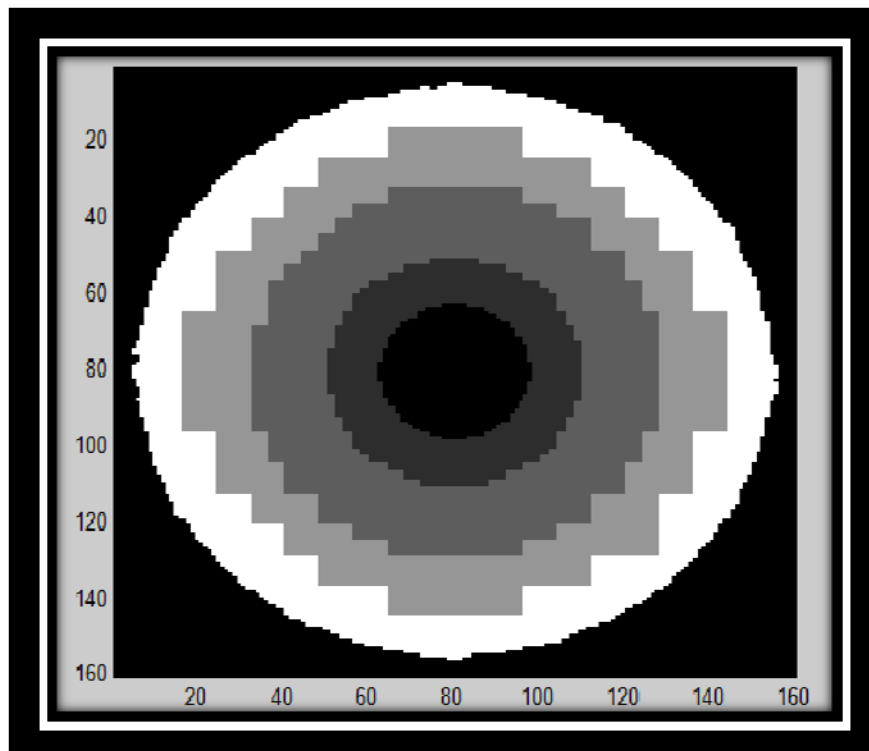


Figure 22: Output video is constructed from five different optical flow vectors which are taken from five different pyramid levels.

In the figure 22 we try to show how our output video is made. The output video is the black and white-striped circle that divides into five parts according to the pyramid level. Each pyramid level provides the most accurate estimated optical flow for every part of output video.

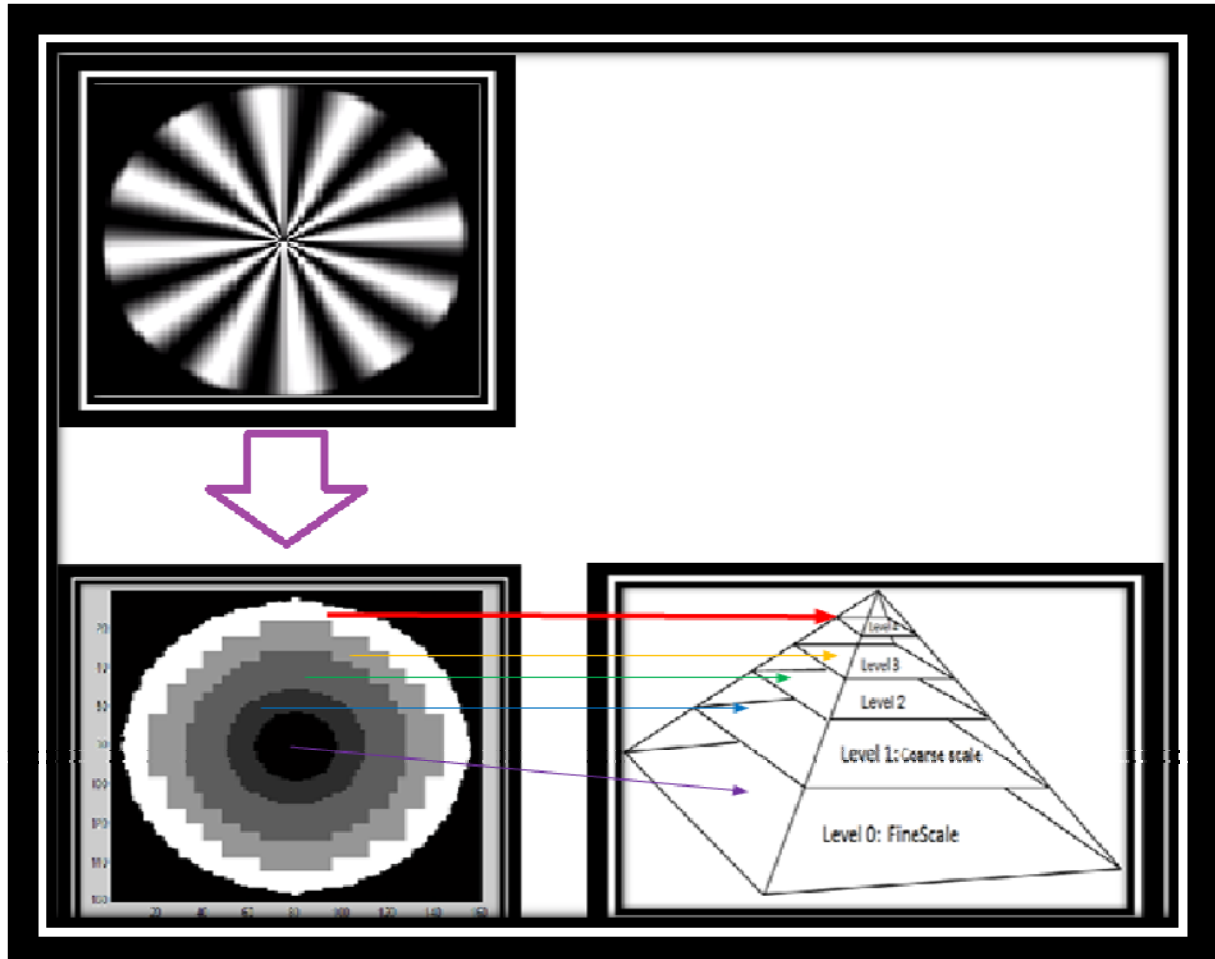


Figure 23: It shows that which pyramids'level is used for building which part of output video

We know that the boundaries have high motion or high frequency, so at this stage, the program automatically chooses the estimated optical flow vectors that are produced in level five of the gaussian pyramid.

The motion speed becomes lower when we move from boundaries to the center of the circle. As a consequence, we understand that the gaussian pyramid level that is used by the program that estimates optical flow vectors becomes less respectively.

The final result of the program is similar to the shape depicted in figure 24 (image that is taken from the output video):

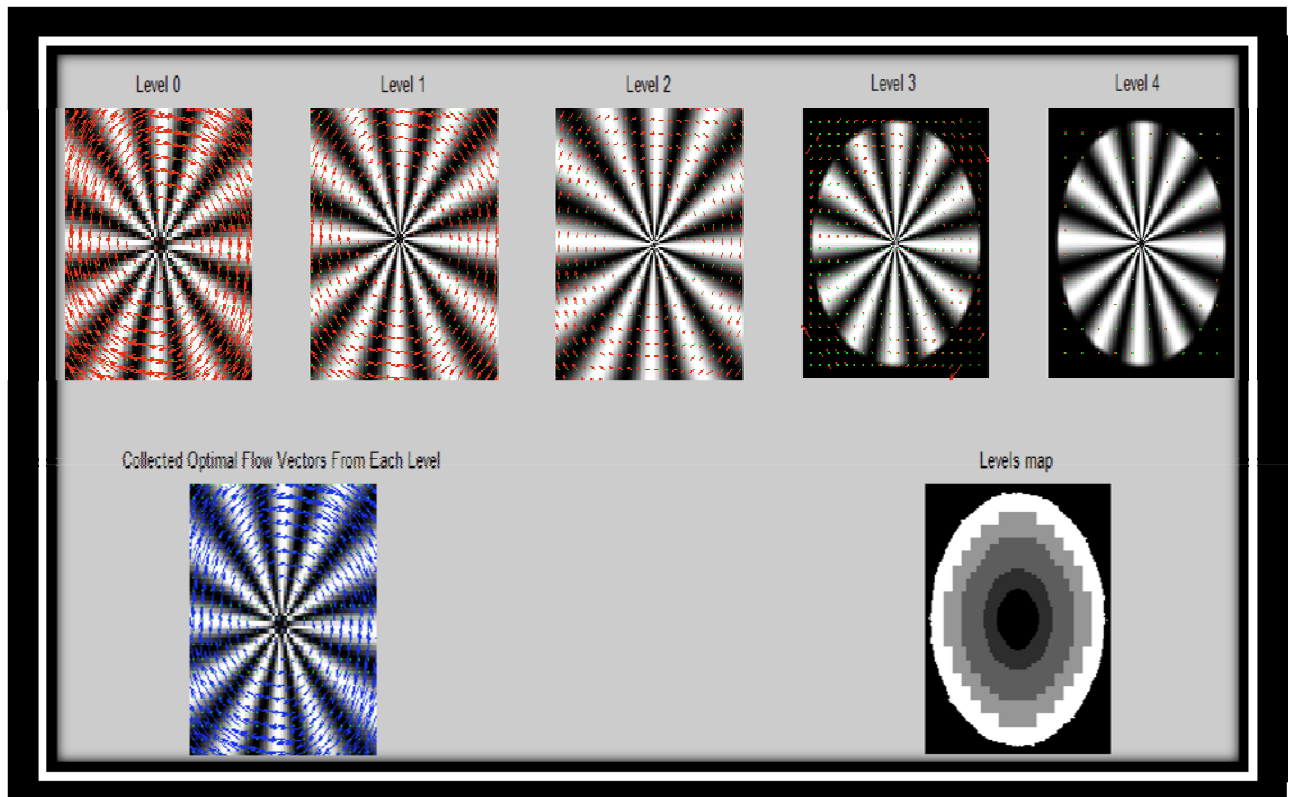


Figure 24: Collecting the most accurate estimated optical flow (down left) from different pyramids levels (top row)

One of the most basic functions for optical flow is the computation of structure tensor and 3D motion. Typically, the chosen algorithm and method play important roles in the accuracy of estimated optical flow and this issue is extremely important. Achieving accurate optical flow estimation requires the analyst to not only pay close attention to details, but also to take into account the realistic imaging property (input video).

In this report, we employed the PL method and the gaussian pyramid (for low pass filtering without fear of losing information) to achieve accurate estimation of optical flow vectors for input videos at any speed motion. In the next chapter, we use the PL method and a multi-scale algorithm for event detection (finding rigid or non-rigid object on input video).

Chapter 5

Event detection

5.1 Making grayscale and binary image

In Chapter 4, we successfully estimated the motion of the image sequence by using the gaussian pyramid. Now, we employ the accurate algorithm for detecting the moving objects (which has an optical flow) and characterizing the object on the video, namely determining whether it is rigid (car) or non-rigid (human).

In our case, we work on video with particular characteristics, such as:

- The location of the camera is fixed. Thus, the background image can be easily separated from the desired objects.
- We observe two different types of visual objects: Vehicles (rigid objects), and Humans (non-rigid objects). We subsequently extract some of their characteristics, such as: direction, appearance/disappearance, speed, position.

After motion estimation, we change every image frame to a binary image. We accomplish this task by generating a magnitude of motion image and assigning a proper threshold to it, so that each pixel is assigned to the object (one) or to the background (zero). In this way, a binary image is created.

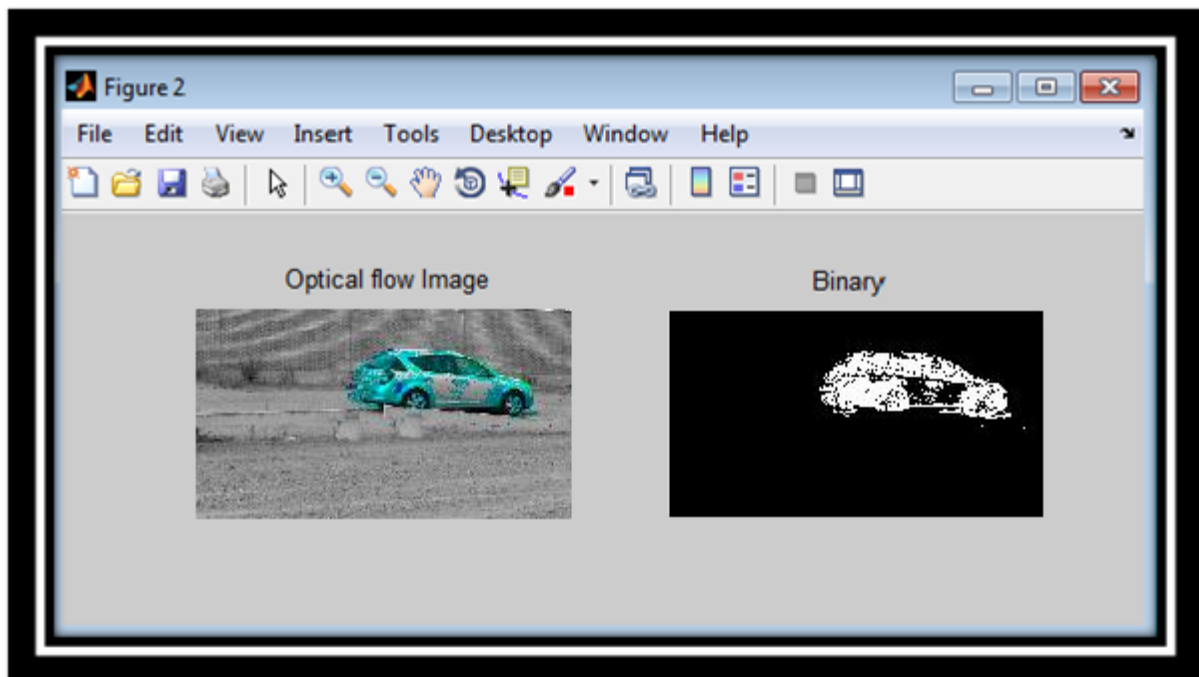


Figure 25: Gray scale and Binary image for one image frame of rigid object

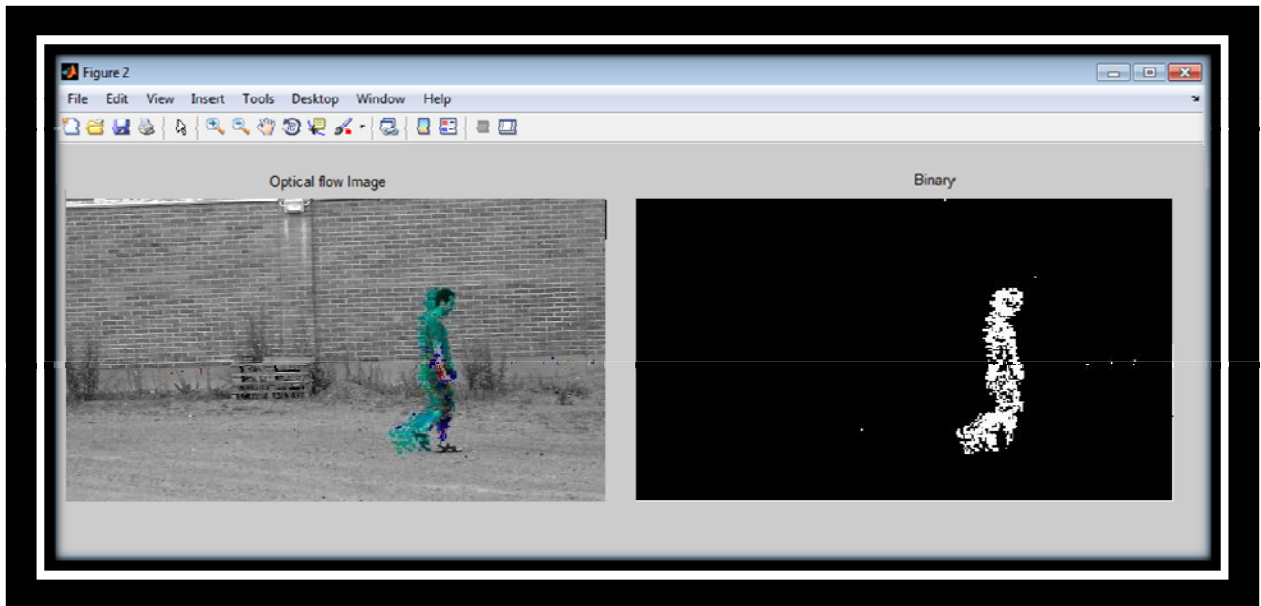


Figure 26: Gray scale and binary image of non-rigid object

In figures 25 and 26 we used color coding in optical flow images. Based on figure 27, if an object goes toward the left (angle is 0), then the optical flow estimation is shown in red, and if it goes toward the right (angle is 180), then it is shown in aqua.



Figure 27: Color coding definition, the hue unit is degree

5.2 Segmentation and Moment calculation

Segmentation is completed by determining which pixel belongs to the object or to the back ground image; it determines the number of objects in each image frame. In an ideal situation, each image frame should consist of one uniform object, as is the case in, figures 26 and 27.

5.2.1 Describe the objects shape

With the help of moment calculation, after generating a binary image, one can describe the objects shape and detect it in every image frame.

A moment m of order $p + q$ for a binary image is defined as:

$$m_{p,q} = \sum_{(x,y) \in obj} x^p y^q \quad (5.1)$$

The area of the object is the moment $m_{0,0}$ and $(\frac{m_{01}}{m_{00}}, \frac{m_{10}}{m_{00}})$ is the center of the object (x_m, y_m) .

For orders higher than 2, central moments are more useful for shape recognition[1], and are defined as :

$$\mu_{pq} = \sum_{(x,y) \in obj} (x - \frac{m_{10}}{m_{00}})^p (y - \frac{m_{01}}{m_{00}})^q \quad (5.2)$$

The central moment $\mu_{p,q}$ is computed from coordinates that are related to the center of the object, and therefore, it is more useful than ordinary moments $m_{p,q}$ as indicators of a shape's features.

When an object moves in video, it changes position in every frame. If it is a rigid body, then we expect a change in its central moments only if its view-point changes .

A human performs frame-by-frame, so non-rigid transformation is expected to change its central moments radically.

A simple detector that is based on central moments of the binary image performs poorly, and to get more reasonable results we analyze on the original flow field.

5.2.2 Rigidity Detector

In a rigid object, the motion flow vectors have the same direction, and they do not omit or cancel each other. In contrast, in non-rigid objects, some motion flow vectors do not have same direction in image frames. In the other words, when the mean of the motion flow vectors in every image frame is calculated, the motion flow vectors omit or attenuate each other. For example, when a human walks, one hand goes forward and the other goes backward; the same is true for human's feet. The mean of the flow vector is calculated as:

$$v_0 = \sum_{(x,y) \in obj} \frac{V(x,y)}{m_{00}} \quad (5.3)$$

m_{00} is the area of the object. A non-rigidity measure is calculated by:

$$NR = \sum_{(x,y) \in obj} \frac{|V(x,y) - v_0|^2}{m_{00}} \quad (5.4)$$

This equation makes intuitive sense if one considers the fully rigid object moving under a pure translation in the image. For a perfect flow algorithm, all vectors are equal to v_0 , and thus NR is zero. A thresholding of the NR forms the basis of a simple classifier that discriminates between human and car. The quality of such a classifier can only be as good as the underlying optical flow algorithms allow it to be.

In the image of the gaussian pyramid levels in figures 28 and 29 (top right of the left side), every part of the moving object, based on its speed, is collected from different pyramid levels. As the right side of these figures shows, the program can correctly detect the moving object (which is shown by the red circle in the figures).

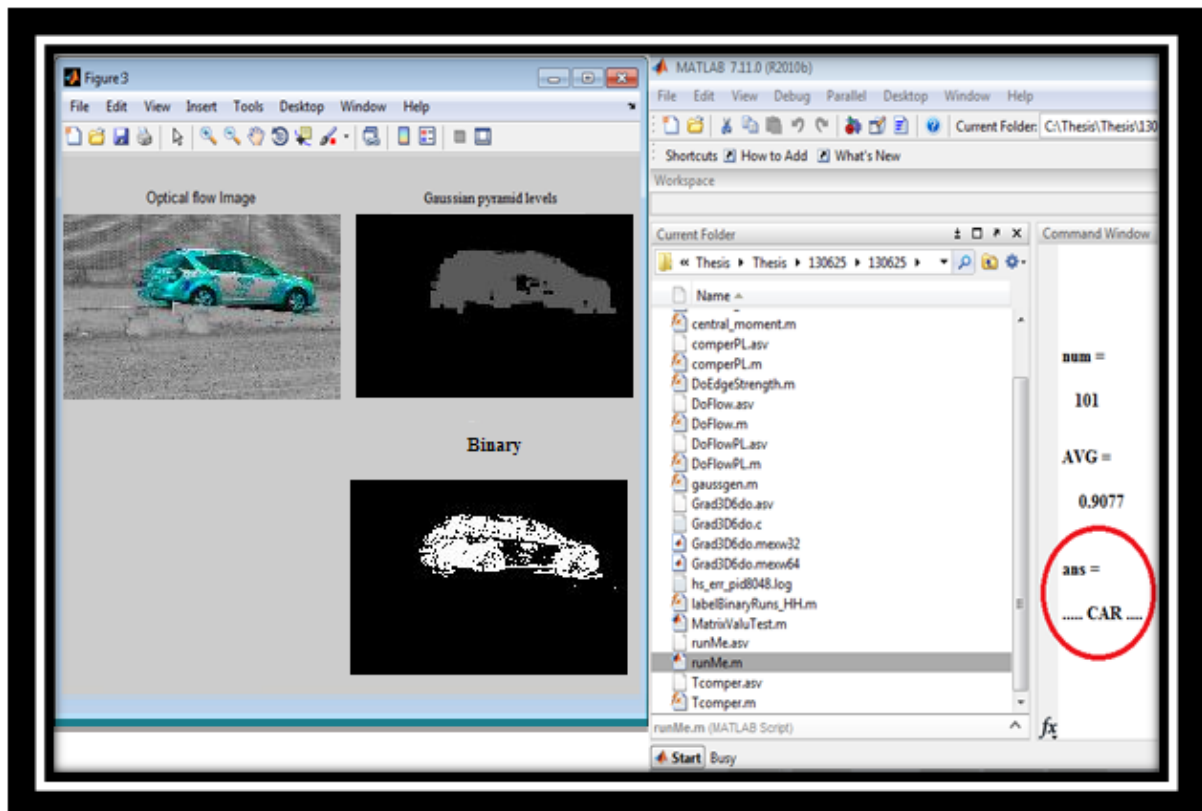


Figure 28: Detecting of moving rigid object

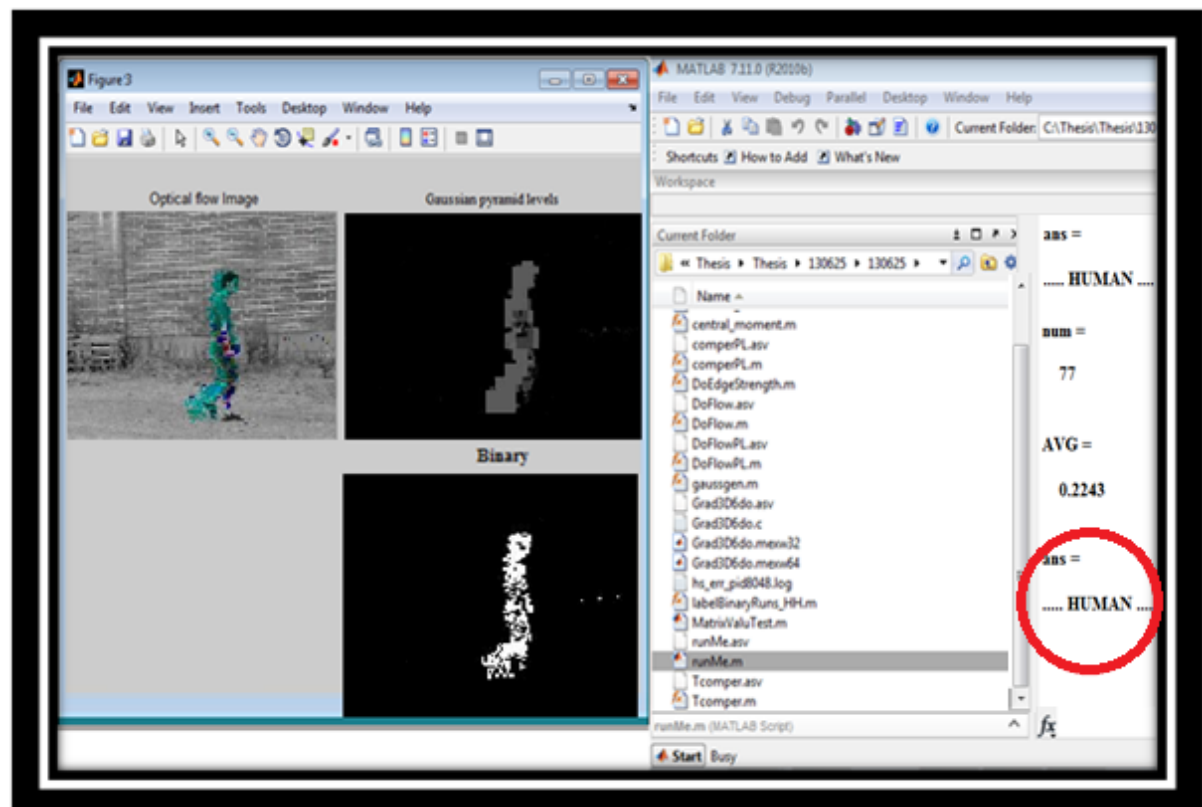


Figure 29: Detecting of moving non-rigid object

5.3 Results on Events Detection

The flow chart used for detection of the object can be simplified, as seen below:

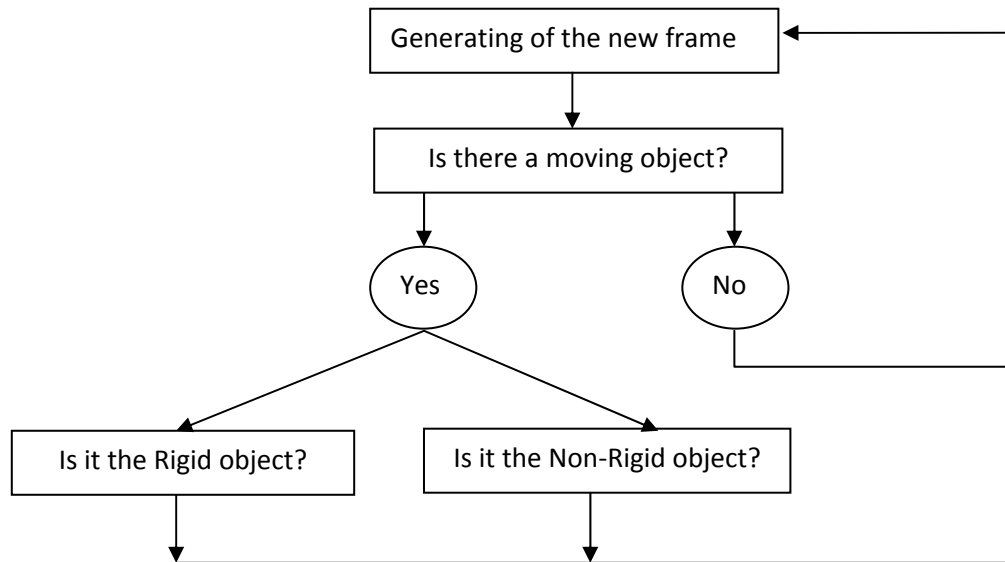


Figure 30: Flow chart of object recognition

The flow chart shows that the algorithm tries to find an object in the frame, and if the object exist the algorithm will find it and use the procedures in sections 5.1 and 5.2 to recognize the object (determine if it is rigid or non-rigid). If there is not any moving object in the frame the algorithm goes to the next frame.

We tried to test the algorithm by using ten different input videos (each with different speeds) to produce the reliable results graph. This graph indicates the percent of correct object detection at different speeds, as seen below:

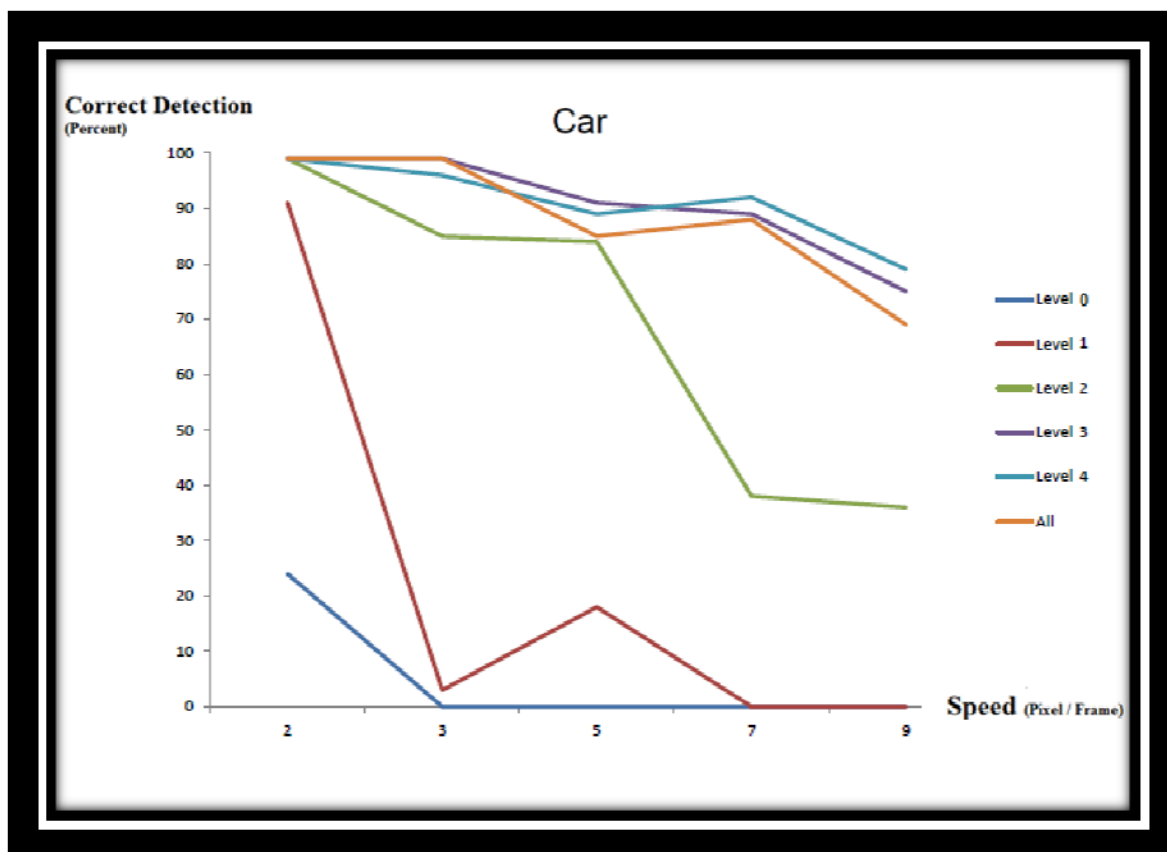


Figure 31: Car detection's graph

The graph in figure 31 shows the function of the five gaussian pyramid levels [see Fig.23] at different speeds for videos that include cars. Figure 32 provides the same graph for those input videos that include humans.

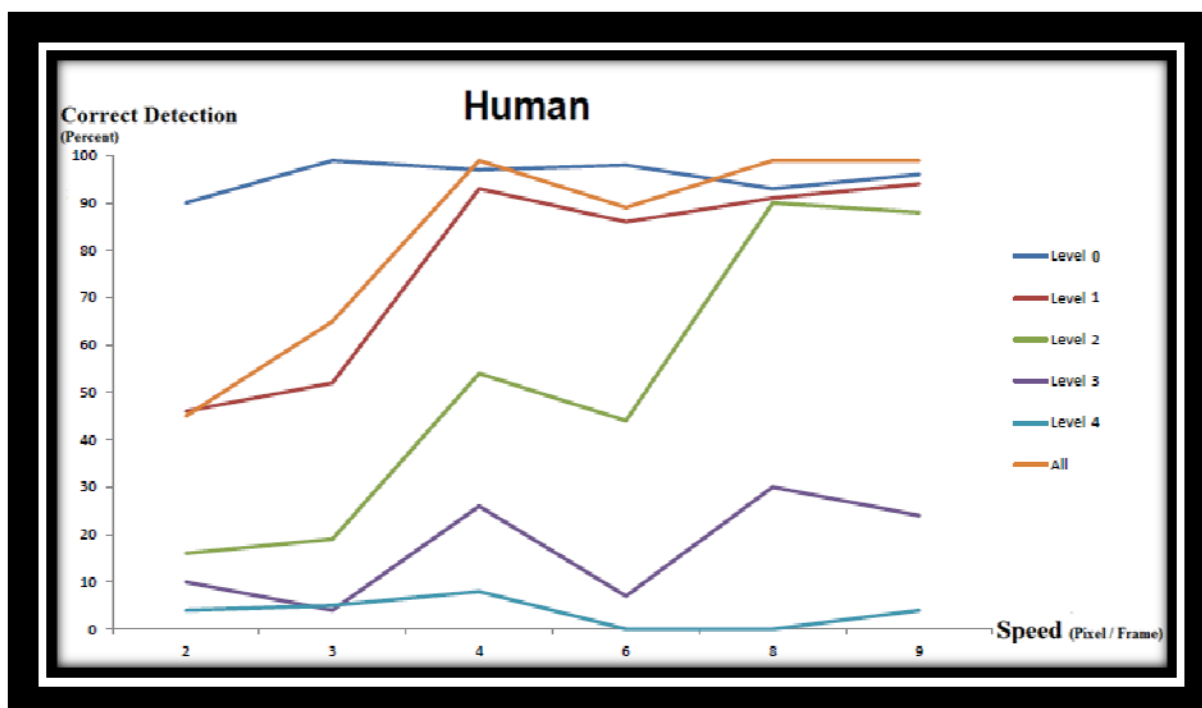


Figure 32: Human detection's graph

In these figures, the orange graphs are the result of our object detection algorithm, which automatically chooses the suitable pyramid levels for the input video.

Figure 31 shows that level 4 of the pyramid outperforms all other levels, and even does slightly better than the combined flow measure for detecting cars. This is expected, because the coarser scale is better adapted to capture the motions of a large solid object. Level 4 has a bias for detecting cars. Conversely, in figure 32 we see that level 0 performs best because the fine level is better adapted to the motions of a smaller object (compared to a car, a human is small); with lots of non-rigid motion. The important result is that the proposed method overcomes the bias of the individual levels; and yields a method that more reliably distinguishes between car and human.

5.4 Conclusions and suggestions for future works

In this thesis, we tried to develop a reliable algorithm able to handle high motion for optical flow estimation and detect rigid or non-rigid objects. To reach these goals, we used five gaussian pyramid levels based on the PL method which has already been introduced [2]. As a conclusion, these points can be noted:

- The multi-scale algorithm accurately estimates the optical flow compared to the two-scale algorithm (fine and coarse scale).
- The PL algorithm has been tested to be useful in a multi-scale algorithm. The results are promising, because of it is a fast optical flow algorithm.
- A simple method of scale selection has proven successful for the task of classifying human versus rigid body movements. Future work will possibly extend this application, using a higher order statistical measure that picks up on the dynamics of moving objects, to further precision than just rigidity.
- The method of calibrating the scale selection using synthetic image sequences has proven effective, even though the test data looked nothing like the final recorded video.

In this report, the subject has been narrowed by only including one moving object in the video and trying to detect it correctly. In future works it will be possible to develop this algorithm for detection of a specific object in a video that includes two or more different objects. Through this method, the efficiency and accuracy of our developed optical flow

estimation algorithm can be examined one more time, especially for videos that have more than one fast moving object simultaneously.

Appendix

1. Developing of the “PL” algorithm

1.1 Starting with making video samples

In this thesis we work with some different video examples that we made by different patterns and specifications, such as:

1.1.1 'Aliased Disk' Video

It displays test image with common Aliasing error.

1.1.1.1 Aliasing Error

When we make reconstruction process for at make digital image ,if the image data is not correctly proceed during reconstruction or sampling then the result image(reconstructed image) will be different from the original image and can be recognizable easily.

Let's look at the code briefly:

❖ General Configurable Parameters :

- DispFramerate = 30(frames/second),
- vidObj.Quality =100;

❖ Image generation Parameters:

- spdMotion = 0.55; (speed of motion of the patterns generated)
- cen1 = 0.55; (centre of circle 1(x offset))
- cW = 0.9; (radius of big circles)
- cW2 = 0.3; (radius of small circles)
- cFuz = 2; (fuzziness of the boundary)
- rotAn = 0; (angle of rotation)

spdRot = 0;spdRot2 = 0;cDetail = 0.7;L = 12;thet = pi/8;imSize = 250;

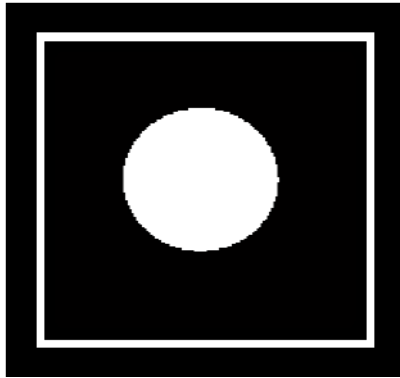


Figure 33: video sample with aliase error (without anti-aliasing filter)

1.1.2 'Fuzzy Disk'

Displays same disk as above, but with a method of proper anti-aliasing

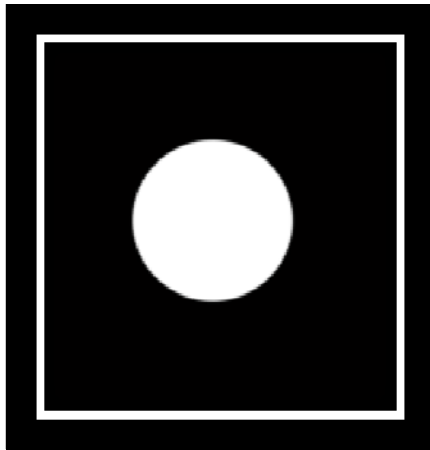


Figure 34: video sample without aliase error (with anti-aliasing filter)

1.1.3 'Check board'

It's a moving circle check board pattern

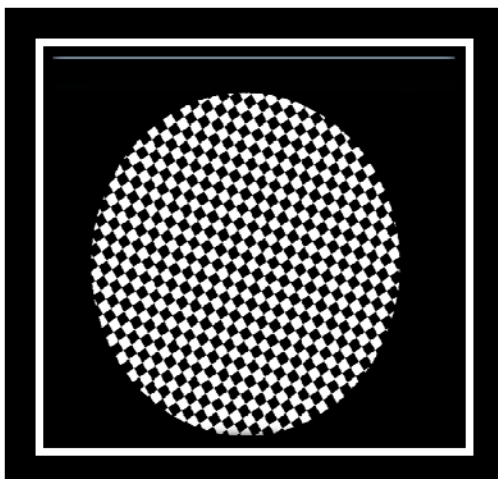


Figure 35: Circle check board "video sample"

```

checkboard = ( checkerboard(16,16,16) >.5); 512 *
512
I=imread('checkerboard512.jpg');
m=size(I,1);
n=size(I,2);
j=1;
radius=m/2;
[xx,yy] = ndgrid((1:m)-radius,(1:n)-radius);
mask = (xx.^2 + yy.^2)<radius^2;
b = cast(mask,'uint8');
J(:, :,3)=b;
J(:, :,2)=b;
J(:, :,1)=b;
I=I.*J;

```

1.1.4 'MazMoh'

It is a radial line of different scale with simple rotation

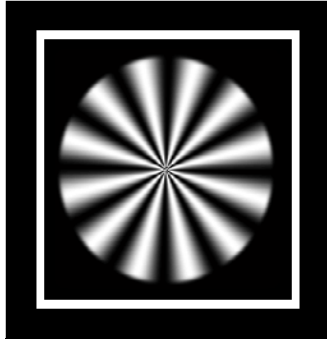


Figure 36: "Radial lines" video sample

```
thet=pi/2;  
  
%The log-spiral  
  
IMS=256*(1+cos((L/abs(sin(thet)))*(cos(thet)*  
log(sqrt(iX.^2+iY.^2))+sin(thet)*atan2(iY,iX))))/2;  
  
%Anti-alias cut off  
  
IMS=IMS.*sig(cW^2-iX.^2-iY.^2, cFuz*cW/50);
```

```
%Speed of rotation, when activated  
  
spdRot=spdRotMax; %(spdRot=-pi/80)  
  
apdRot2=spdRotMax;
```

One can find details of making these video samples such as Motion speed, number of frame per second,... in attached Matlab file that calls Generatrmmotionstimulus.m , we use them for generating frames (from video)and building Gaussian pyramids levels .

Way we do is to estimate optical flow of desired sampled that exist in video clip in different levels of Gaussian pyramid and find error between estimated optical flow and calculated flow which will find with knowing of speed of frames and thus specific pixel in next frame and comparing the results in each level to find and collect best pixel in each level and using them to build a new video that is constructed from best pixel of each pyramid level..

The main Matlab file (m file) which we made is "Vidprocessing" that we explain it step by step in next section.

1.2 How to set input parameters

The input parameters of the “PL” algorithm set as below:

```
Function [dx, dy, dt, gradInd] = vidProcessing (movieType, method,  
spdFactor,bFineScale,nofTimeSlices)
```

It generates a sequence of test images, with estimated flow over a sequence, indicated by 'movieType'.

'movieType': It is the path to an avi file in the current folder for generating images indeed it can be 'filename.avi'(file on disk in current folder),

'synthetic'(manufactured test sequence) or 'camera' (setups the default video input device for capturing video for this application).

For example: VidProcessing ('sample.avi') assumes sample.avi file that is in the current folder as input video which is used to generate images.

Output dx, dy and dt are all WxHxT matrices containing the x, y, and t

That is partial derivatives over time also W and H are the height and width of the input video, and T is "nofTimeSlices" .

For example, dx(:, :, 1), will hold one of the last x derivative images of the video and together with dy(:, :, 1) makes up the 2D gradient of one of the last images of the video.

Method :It is the method that calculate optical flow, for example it can be:

"LK" (Lukas and Kanade method)

"TS" (3D structure tensor method)

"PL" (Point, Line method)

These will not give flow output:

"gradient" Displays the gradient values

"edge" Displays the 2D edge detection

1.2.1 Speed Factor

It is to change the speed of the sample video generation (if $\text{spdFactor}=2$, then the sample sequence is twice as fast)

1.2.2 Scale

It determines that in what scale the differential operations would take a place (which pyramid level)

0: Fine Scale

1: Coars Scale

3:Fourth level of pyrmid

2:Third level of pyramid

4:Fifth level of pyramid

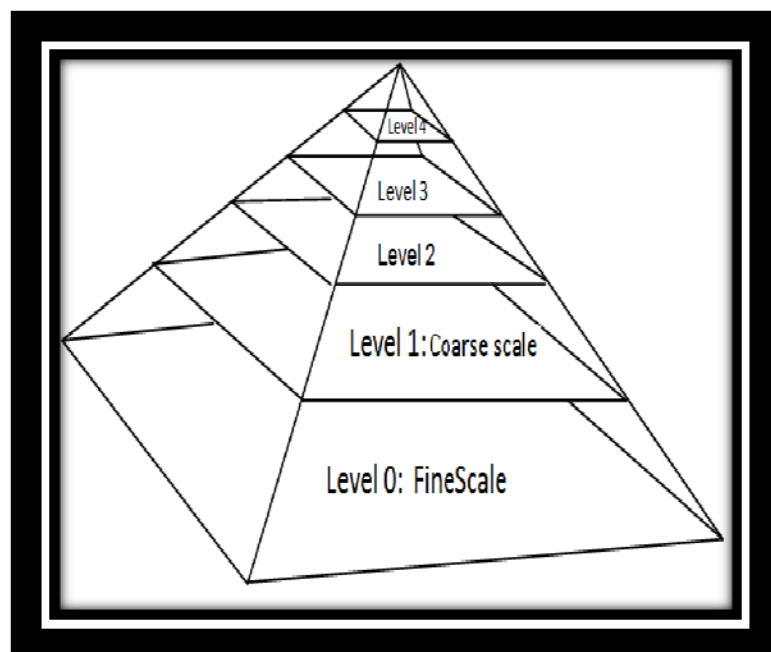


Figure 37: Gaussian Pyramid

Scales can be made as follows:

If Scale=0, It means original image for example: Image's Height: 512 pixel and Image's width:512 pixel

If Scale=1, It means image level 1 and Image's Height: $\frac{512}{2} = 256$ pixel and Image's width: $\frac{512}{2} = 256$ pixel

If Scale=2, It means image level 2 and Image's Height: $\frac{256}{2} = 128$ pixels and Image's width: $\frac{256}{2} = 128$ pixel

If Scale=3, It means image level 3 and Image's Height: $\frac{128}{2} = 64$ pixel and Image's width: $\frac{128}{2} = 64$ pixel

If Scale=4, It means image level 4 and Image's Height: $\frac{64}{2} = 32$ pixel and Image's width: $\frac{64}{2} = 32$ pixel

Then by using matlab "mex" interface a C file that calls "Grade 3D" is used for calculation of convolution of each level and outputs (derivatives) send back to matlab file "VidProcessing".

The idea is to iterate through the video, frame by frame, calculate 3D gradients at each point and then use those 3D gradients to calculate the flow. The flow is displayed at each new frame.

As it has described before, for calculation of optical flow, 3D structure tensor matrix Eq.(2.16) should be solved, so after getting the results from C file(Grade 3D) matlab file "Vidprocessing" starts to calculate moments (elements of 3D structure tensor Matrix) by running this script [U1, V1, UG, VG] = DoFlow(dx,dy,dt,method) and as a result calling 'Doflow.m' file :

m_{200} : This moment, calculated in three steps explicitly:

make element wise product, and sum along time direction (time integration):

`momentIm = sum(double(dx).^2,3);`

smooth with large separable Gaussian filter (spatial integration)

`momentIm = filter2(gg',filter2(gg, momentIm));`

down sample to specified resolution:

`m200=imresize(momentIm,[TensorRes TensorRes],'nearest')/normFactor;`

That means:

`m200= imresize(filter2(gg',filter2(gg,(sum(double(dx).^2 ,3)))),[TensorRes TensorRes],'nearest')/normFactor;`

The remaining moments are calculated in exactly the same way:

`m020=imresize(filter2(gg',filter2(gg,(sum(double(dy).^2 ,3)))),[TensorRes TensorRes],'nearest')/normFactor;`

```
m002=imresize(filter2(gg',filter2(gg,(sum(double(dt).^2,3)))),[TensorRes
TensorRes],'nearest')/normFactor;
```

```
m110=imresize(filter2(gg',filter2(gg,(sum(double(dx).*double(dy),3)))),[TensorRes
TensorRes],'nearest')/normFactor;
```

```
m101=imresize(filter2(gg',filter2(gg,(sum(double(dx).*double(dt),3)))),[TensorRes
TensorRes],'nearest')/normFactor;
```

```
m011=imresize(filter2(gg',filter2(gg,(sum(double(dy).*double(dt),3)))),[TensorRes
TensorRes],'nearest')/normFactor;
```

The calculated flow will be held in matrices U1, V1 :

```
U1 = ( m101.*m110 - m011.*m200)./tensDet;
V1 = (-m101.*m020 + m011.*m110)./tensDet;
tensDet = (m020.*m200 - m110.^2);
```

The formulation of Lucas and Kanade calls for an inversion of the 2nd moment matrix multiplied with the 2D vector (-m101,-m011).

Indeed moments as defined above are sums of products of derivatives.

Then we can show calculate optical flow on each video frame by following scripts:

```
set(hImObj,'cdata',curlm);
```

This Paints the line and point flow:

```
set(hQvObjLines,'UData',sc*U1,'VData',sc*V1);
```

2. Calculation of error vectors (displacement vector) for each pixel

```
vDx = - spdRotMax*y.*disk;
vDy = spdRotMax*x.*disk;
vDy = imresize( vDy ,[TensorRes TensorRes],'nearest');
vDx = imresize( vDx ,[TensorRes TensorRes],'nearest');
UG=vDx-U1;
VG= vDy-V1;
Set (hQvObjErrs,'UData',sc*UG,'VData',sc*VG);
```

2.1 Estimated Optical flow Vectors (Red) and Calculated flow motions (Blue) with desired speed

As we describe it above (section 2) for estimating how accurate results are, we calculate error vectors (displacement vector) for each pixel in each pyramids' level.

Given the motion speed as a known variable so we can estimate position of each point along axis x and y and save them in matrix UG , VG and with subtraction of U , V (they are matrixes that include estimated position by optical flow) we can reach to error vectors (displacement vector) and show them on our video sample. (Code exists in section 2.2 of appendix)

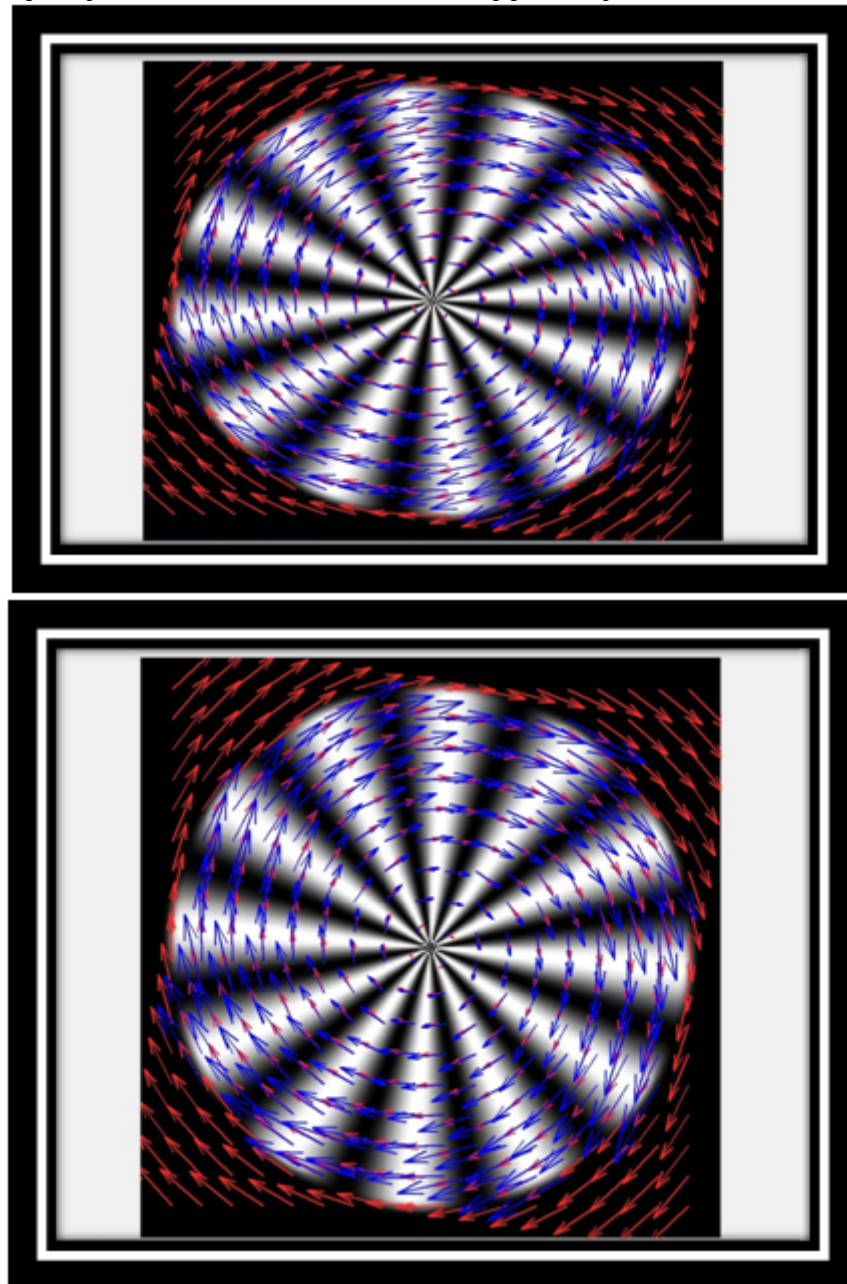


Figure 38: Two consecutive frames in level 4 of pyramid, motion speed $\pi/10$

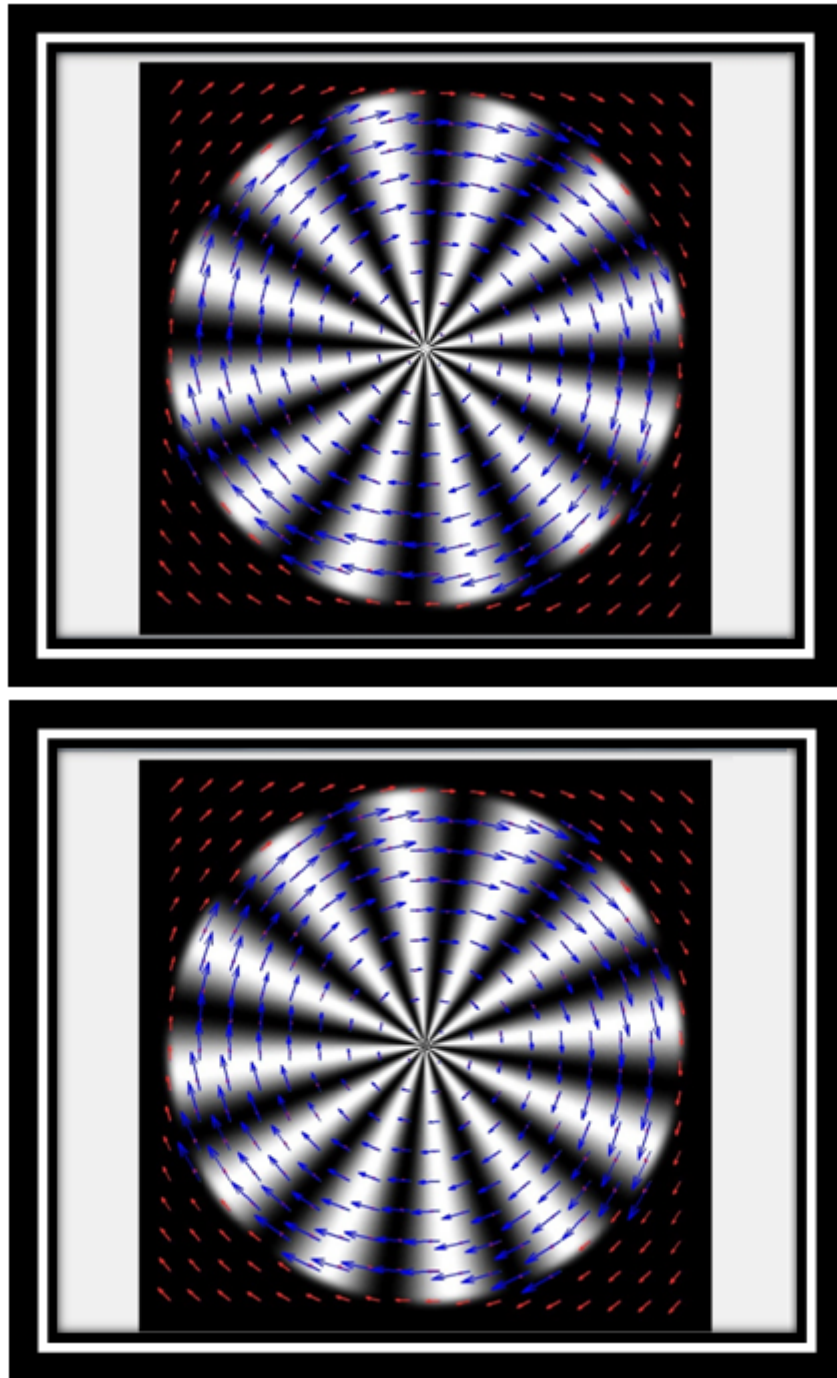


Figure 39: Same frames as Fig.38 but motion speed is $\pi/50$

2.2 Calculation of MSE (Mean Squared Error).

Code is:

```
MSEu = mean (mean ((UG). ^2, 2), 1);
```

```
MSEv = mean (mean ((VG). ^2, 2), 1);
```


References:

- [1] J. Bigun, "Vision with direction", *a systematic introduction to image processing and computer vision*. Berlin ; Springer, 2006.
- [2] S. M. Karlsson and J. Bigun, "Lip-motion events analysis and lip segmentation using optical flow", *CVPR Workshops*, pp. 138–145, 2012.
- [3] J. K. Aggarwal and Q. Cai, "Human Motion Analysis: A Review", *Computer Vision and Image Understanding*, vol. 73, pp. 428–440, 1999.
- [4] I. Laptev and T. Lindeberg, "Space-time Interest Points", *ICCV*, pp. 432–439, 2003.
- [5] D. Lowe, "Object Recognition from Local Scale-Invariant Features", pp. 1150–1157, 1999.
- [6] A. Wedel, C. Rabe, T. Vaudrey, T. Brox, U. Franke, and D. Cremers, "Efficient Dense Scene Flow from Sparse or Dense Stereo Data", *Computer Vision – ECCV*. Berlin ; Springer, pp. 739–751, 2008.
- [7] W. B. T. Claude and L. Fennema, "Velocity determination in scenes containing several moving objects", *Computer Graphics and Image Processing*, no. 4, pp. 301–315, 1979.
- [8] C. Lodato and S. Lopes, "A New Parallel Differential Method for Optical Flow Estimation", *Math Imaging Vis*, vol. 26, no. 3, pp. 345–356, Dec. 2006.
- [9] J. Kim and T. Sikora, "Hybrid recursive energy-based method for robust optical flow on large motion fields", *IEEE International Conference on Image Processing, ICIP*, vol. 1, pp. I–129–132, 2005.
- [10] T. Kuremoto, K. Koga, K. Kobayashi, and M. Obayashi, "Computing Slow Optical Flow by Interpolated Quadratic Surface Matching", presented at the Visualization, Imaging, and Image Processing, 2004.
- [11] S. S. Beauchemin and J. L. Barron, "The Computation of Optical Flow", *ACM Comput. Surv.*, vol. 27, no. 3, pp. 433–466, Sep. 1995.
- [12] B. D. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision", *DARPA Image Understanding Workshop*, pp. 121–130, April.1981.
- [13] J. Bigun and G. H. Granlund, "Optical Flow Based on the Inertia Matrix of the Frequency Domain", presented at the Proceedings from SSAB Symposium on Picture Processing : Lund University, Sweden, pp. 132–135, 1988.
- [14] J. Bigun, G. H. Granlund, and J. Wiklund, "Multidimensional orientation : texture analysis and optical flow", presented at the Proceedings of the SSAB Symposium on Image Analysis : Stockholm, Sweden, 1991.

- [15] B. K. P. Horn and B. G. Schunck, "Determining Optical Flow," *Artificial Intelligence*, vol. 17, pp. 185–203, 1981.
- [16] J. L. Barron, D. J. Fleet, and S. S. Beauchemin, "Performance of optical flow techniques," *Int Comput Vision*, vol. 12, no. 1, pp. 43–77, Feb. 1994.
- [17] N. Bauer, P. Pathirana, and P. Hodgson, "Robust optical flow with combined Lucas-Kanade/Horn-Schunck and automatic neighborhood selection," *Sustainable development through effective man-machine co-existence*, pp. 378–383, Jan. 2006.
- [18] J. Shi and C. Tomasi, "Good features to track," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Proceedings CVPR*, pp. 593–600, 1994.
- [19] H. Eltoukhy and K. Salama, "Multiple camera object tracking," *Power Show*, [Online]. Available:
http://www.powershow.com/view1/c24e6-ZDc1Z/Multiple_Camera_Object_Tracking_powerpoint_ppt_presentation
 [Accessed: 11-Jan-2014].
- [20] S. M. Karlsson and J. Bigun, *Open source program*, [Online]. Available:
<http://www2.hh.se/staff/josef/downloads/education/labs.html> (Exercise 5)
 [Accessed: 01-Sep-2013].
- [22] J. S. Serra, "Optical Flow of points and lines for Human Machine Interfacing", *Master Thesis of Computer Machine*: Halmstad university, Sweden, 2013.



PO Box 823, SE-301 18 Halmstad
Phone: +35 46 16 71 00
E-mail: registrator@hh.se
www.hh.se