

# **EXTRACT MOTION FROM PICTURE SEQUENCE**

By

MOHD SHAFIQ AHMAD DAHALAN

FINAL PROJECT REPORT

Dissertation submitted in partial fulfillment of  
the requirements for the  
Bachelor of Engineering (Hons)  
(Mechanical Engineering)

JANUARY 2008

Universiti Teknologi PETRONAS  
Bandar Seri Iskandar  
31750 Tronoh  
Perak Darul Ridzuan

## **ABSTRACT**

Optical flow is a concept for estimating motion of objects within a visual representation. Motion is represented as vectors at every pixel in a digital image sequence of two frames which are taken at time  $t$  and  $t + \delta t$  respectively. Based on the assumption of the observed brightness (intensity) of any object point is constant over time interval and the movement is small, optical flow equation (OFE) is derived and used in algorithm to calculate optical flow for a particular motion. In order to inject robustness in the algorithm, time-varying uniform illumination and calculation of large range of motion also have been taken into account while formulating the algorithm. The algorithm is used as reference to create a coding input for the MATLAB, which is the software used in this project. As a result, the software generates a separate view of static and moving objects. Further study of the output obtains more accurate data. The experimental process will be carried out using a video as an input to determine whether the project succeeds or need more modifications. The outcome of this project is beneficial for motion analyst to predict the parameter of moving objects, animations, statistics and also robotic eye.

# TABLE OF CONTENTS

<b>ABSTRACT</b> .....	ii
<b>TABLE OF CONTENTS</b> .....	iii
<b>LIST OF ILLUSTRATION</b> .....	iv
<b>CHAPTER 1- INTRODUCTION</b>	
1.1 Optical Flow Concept.....	1
1.2 Problem Statement.....	3
1.3 Objective and Scope of Study.....	4
<b>CHAPTER 2-LITERATURE REVIEW</b>	
2.1 Optical Flow Equation.....	5
2.2 Lukas-Kanade Algorithm.....	6
2.3 Robust Optical Flow Algorithm.....	8
2.4 Camera Model.....	9
<b>CHAPTER 3-METHODOLOGY/ PROJECT WORK</b>	
3.1 Camera Calibration.....	13
3.2 Optical Flow Algorithm.....	15
3.2.1 Robust Optical Flow Algorithm.....	15
3.2.2 Illumination Variation Correction.....	16
3.2.3 Iterative Lukas-Kanade Optical Flow.....	17
<b>CHAPTER 4- RESULT AND DISCUSSION</b>	
4.1 Camera Calibration Result.....	18
4.2 Optical Flow Algorithm Performance.....	20
4.2.1 Synthetic Motion Test.....	20
4.2.2 Real Motion Test.....	24
4.3 Discussion.....	26

<b>CHAPTER 5- CONCLUSION AND RECCOMENDATION</b> .....	28
<b>REFERENCES</b> .....	29
<b>APPENDICES</b> .....	31
APPENDIX A: Matlab Source Code	
APPENDIX B: Pictures of calibration	
APPENDIX C: Gantt Chart	

## LIST OF ILLUSTRATIONS

### LIST OF FIGURE

Figure 1-1: Projection on the image plane of 3D motion	1
Figure 1-2: Optical flow field from sequence of image	2
Figure 2-1: Region that is taken into account for calculation of $k$	8
Figure 2-2: Gaussian pyramids	9
Figure 2-3: Camera standard coordinate system	10
Figure 2-4: Camera system setup	11
Figure 3-1: Camera calibration toolbox	14
Figure 4-1: Camera calibration result	18
Figure 4-2: Image of camera calibration (camera-centered)	19
Figure 4-3: Image of camera calibration (world-centered)	20
Figure 4-4: Image sequence for optical flow algorithm calculation which has undergone illumination effect	21
Figure 4-5: Optical flow result for real motion estimation	24

### LIST OF TABLE

Table 4-1: The result of calculated optical flow with compared to actual displacement	23
Table 4-2: The results of calculated optical flow with illumination effect using original iterative Lukas-Kanade algorithm (i.e. without illumination correction)	24
Table 4-3: The result of estimated velocity, $v_e$ compared to average value, $v_c$	26

# CHAPTER 1

## INTRODUCTION

### 1.1 Optical Flow Concept

In a digital image, each pixel is represented by a value of intensity ( $I$ ), obtained from the projection of objects in 3D motion field to a 2D image plane. As the objects move, the corresponding projections in the 2D image plane also change their position. Optical flow is the vector field that reflects the direction and magnitude of the relative displacement of these pixels over the sequence of images.

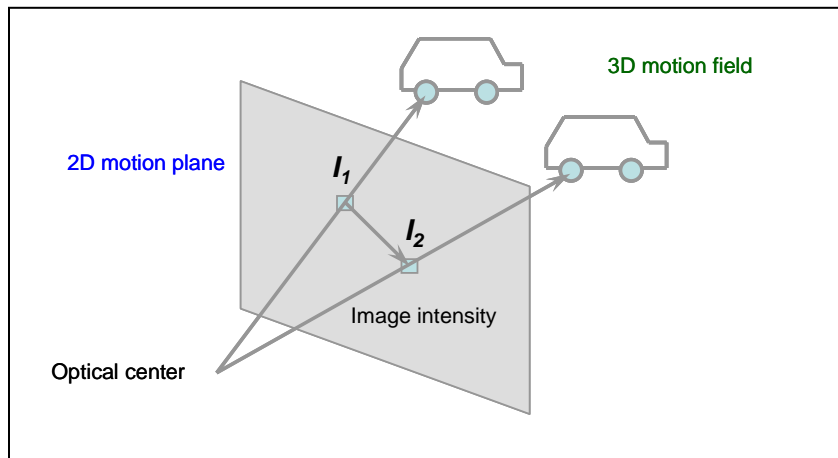


Figure 1-1: Projection on the image plane of the 3D motion

Optical flow is useful in pattern recognition, computer vision, and other image processing applications. It is closely related to motion estimation and motion compensation. Some consider using optical flow for collision avoidance and altitude acquisition system for unmanned air vehicles (UAVs). For motion estimation application, the optical flow algorithm

is used to estimate an optical flow from an image pair. The optical flow is then used to estimate the vehicle velocity.

Figure 1-2 shows the optical flow vectors which represent the motion of object in a digital image sequence.

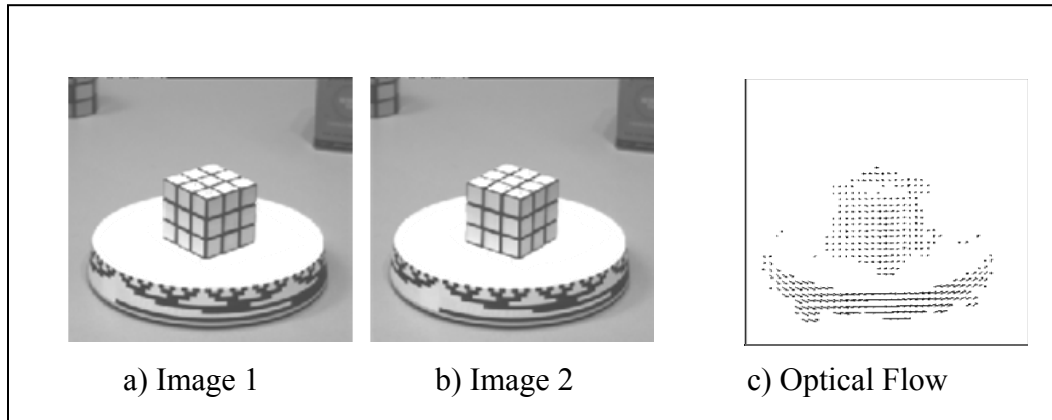


Figure 1-2: Optical flow field from sequence of image.

Horn et. al. [3] did pioneering work on the development of optical flow techniques based on computing spatiotemporal differences from an image sequence. Since then, many methods and algorithms for determining optical flow were developed [4]. According to the result of J.L. Barron et. al. [4], Kanade-Lucas optical flow algorithm [5] is used because it is robust, accurate, insensitive to noise and non-uniform light intensity sources, and suitable for real-time computation. Therefore, a method proposed by Lukas and Kanade is used as the basis for the optical flow algorithm for this project.

## 1.2 Problem Statement

An optical flow algorithm has been developed for synthetic motion estimation. Two images,  $8 \times 8$  pixels<sup>2</sup> in size, are used in the algorithm to represent the simple motion sequences. The first images,  $I(x, y, t)$ , represents the image intensity at time,  $t$  and spatial coordinates,  $x$  and  $y$ . The second image,  $I(x+\delta x, y+\delta y, t+\delta t)$ , represents the synthetic motion of the  $\delta x$  and  $\delta y$ , and the change in time,  $\delta t$ .

However, this algorithm has several limitations that certainly affect its credibility to deliver accurate results as a motion estimator. These limitations are noted as follow:

- a) Under varying illumination condition, it may fail to calculate accurately since the method itself relies on the pixels' intensity value which is assumed to be constant over the time interval.
- b) When dealing with larger motion range (more than 1 pixel), the algorithm failed to estimate motion accurately. This is due to the assumption made in optical flow equation's derivation whereby the motion is small.
- c) This algorithm is yet to be applied with real image in which is very important to determine its applicability in real life application.



### **1.3 Objectives and Scope of Study**

The main objective of this project is to develop a robust optical flow algorithm for the motion estimation. This robust algorithm will be developed to overcome the limitations that have been described in previous section. It is expected to have capability of producing accurate results even under influence of uniform illumination changes and long ranges of motion (more than 1 pixel).

The algorithm is expected to receive input (frames of images) from external devices such as digital camera via MATLAB's Image Acquisition Toolbox. Using a video input object, live data is acquired and analyzed to calculate any motion between two adjacent image frames. Any motion in the stream (optical flow field) is plotted in a MATLAB figure window.

The scope of the project also included experiment to validate the optical flow algorithm by using real image. The experiment enables the evaluation of the algorithm's accuracy and the overall performance with respect to the predefined project's objective.

## CHAPTER 2

### LITERATURE REVIEW

#### 2.1 Optical Flow Equation

Optical flow is a concept for motion estimation of objects within a visual representation. Typically the motion is represented as vectors originating or terminating at pixels in a digital image sequence which are taken at times  $t$  and  $t + \delta t$ . Assuming image intensity,  $I$  (brightness) is roughly constant over short intervals, as a pixel at location  $(x, y, t)$  with intensity  $I(x, y, t)$  will have moved  $\delta x$ ,  $\delta y$  and  $\delta t$  between the two frames, following image constraint equation can be given:

$$I(x, y, t) = I(x + \delta x, y + \delta y, t + \delta t) \quad (2.1)$$

Assuming the movement to be small, we can develop the image constraint at  $I(x, y, t)$  with first order Taylor series to get:

$$I(x + \delta x, y + \delta y, t + \delta t) = I(x, y, t + \delta t) + \frac{\partial I}{\partial x} \delta x + \frac{\partial I}{\partial y} \delta y + \text{Higher order terms}$$
$$I(x + \delta x, y + \delta y, t + \delta t) \approx I(x, y, t + \delta t) + \frac{\partial I}{\partial x} \delta x + \frac{\partial I}{\partial y} \delta y \quad (2.2)$$

From the Equation 2.1 and 2.2, we achieve:

$$[I(x, y, t + \delta t) - I(x, y, t)] + \frac{\partial I}{\partial x} \delta x + \frac{\partial I}{\partial y} \delta y \approx 0$$

$$\frac{\partial I}{\partial t} + \frac{\partial I}{\partial x} \delta x + \frac{\partial I}{\partial y} \delta y \approx 0$$

Rewrites the derivatives,  $\frac{\partial I}{\partial t} = I_t$ ,  $\frac{\partial I}{\partial x} = I_x$  and  $\frac{\partial I}{\partial y} = I_y$ , the optical flow equation is:

$$I_t + I_x \delta x + I_y \delta y = 0 \quad (2.3)$$

Equation 2.3 has two unknowns and so does not have a unique solution. This is the mathematical problem that there is not enough information in a small area to uniquely determine motion. But Horn and Schunck [3] has add an additional constrain by using a global regularization calculation. They assume that images consist of objects undergoing rigid motion, and so over relatively large areas the optical flow will be smooth. They then minimize the square of the magnitude of the gradient of optical flow using the equation

$$\int_D (\nabla I \cdot \vec{v} + I_t)^2 + \lambda^2 \left[ \left( \frac{\partial v_x}{\partial x} \right)^2 + \left( \frac{\partial v_x}{\partial y} \right)^2 + \left( \frac{\partial v_y}{\partial x} \right)^2 + \left( \frac{\partial v_y}{\partial y} \right)^2 \right] \delta x \delta y \quad (2.4)$$

In contrast, Lucas and Kanade [5] use a local least squares calculation to provide the constraint by minimizing in the neighborhood surrounding the pixel, represented in equation form by

$$\sum_{x,y \in \Omega} W^2(x,y) [\nabla I(x,y,t) \cdot \vec{v} + I_t(x,y,t)]^2 \quad (2.5)$$

## 2.2 Lukas- Kanade Algorithm

According to the results presented in [4], Lukas-Kanade optical flow algorithm is commonly adapted in optical flow calculation due to its robustness, accuracy and feasibility for real time computation. The optical flow equation has two unknown and cannot be solved as such. This is known as the aperture problem of the optical flow algorithms. To find the optical flow, we

need another set of equation which is given by some additional constraint. The solution as given by Lukas and Kanade is a non-iterative method which assumes a locally constant flow in small windows.

From Equation 2.3, assuming that the optical flow  $(\delta x, \delta y)$  is constant and numbering the pixels as 1...n, we get:

$$\begin{bmatrix} I_{x1} & I_{y1} \\ I_{x2} & I_{y2} \\ \vdots & \vdots \\ I_{xn} & I_{yn} \end{bmatrix} \begin{bmatrix} \delta x \\ \delta y \end{bmatrix} = \begin{bmatrix} -I_{t1} \\ -I_{t2} \\ \vdots \\ -I_{tn} \end{bmatrix}$$

or

$$A\vec{v} = -b$$

To solve the over-determined system of equation, we use the least squares method:

$$(A^T A) \vec{v} = A^T (-b)$$

$$\vec{v} = (A^T A)^{-1} A^T (-b)$$

then, we achieve:

$$\begin{bmatrix} \delta x \\ \delta y \end{bmatrix} = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix}^{-1} \begin{bmatrix} -\sum I_x I_t \\ -\sum I_y I_t \end{bmatrix} \quad (2.6)$$

### 2.3 Robust Optical Flow Algorithm

The derivation of the optical flow equation assumes that the intensity (brightness) of a pixel does not change along its motion trajectory [5]. This assumption is generally applicable but not true all the time since the illumination may change over the time interval. Assuming the source of illumination is from a far distance, the illumination will have a uniform effect,  $k$  to the entire frame (feature window area). Then, the Equation 2.1 will become:

$$I(x, y, t) + \kappa = I(x + \delta x, y + \delta y, t + \delta t) \quad (2.7)$$

When dealing with optical flow calculation involving illumination variation problem, it is necessary to compensate the effect of changes in one frame in accordance to the other frame. As depicted in Figure 2-1, the value of  $k$  can be estimated by calculating the average difference of intensity values between the second frame and the first frame. The calculation is limited within the highlighted region in which is common for both frames.

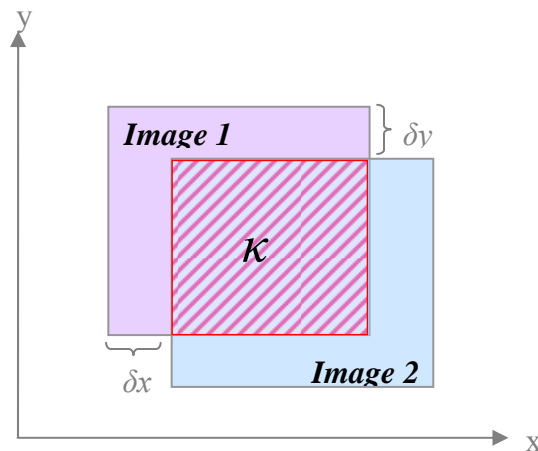


Figure 2-1: Region that is taken into account for calculation of  $k$ .

When the motion is not small (more than 1 pixel), the higher order terms in Equation (2) become more significant and dominant. Thus the algorithm will produce less or no accurate results as a motion estimator. In order to solve this problem, an iterative Lukas-Kanade

algorithm is applied which practices coarse-to-fine optical flow estimation. This approach relies on estimating the flow in an image Gaussian pyramid in Figure 2-2, where the apex is the original image at a coarse scale, and the levels beneath it are warped representations of the images based on the flow estimated at the preceding scale. This ensures that the small motion assumption of Equation 2.2 remains valid.

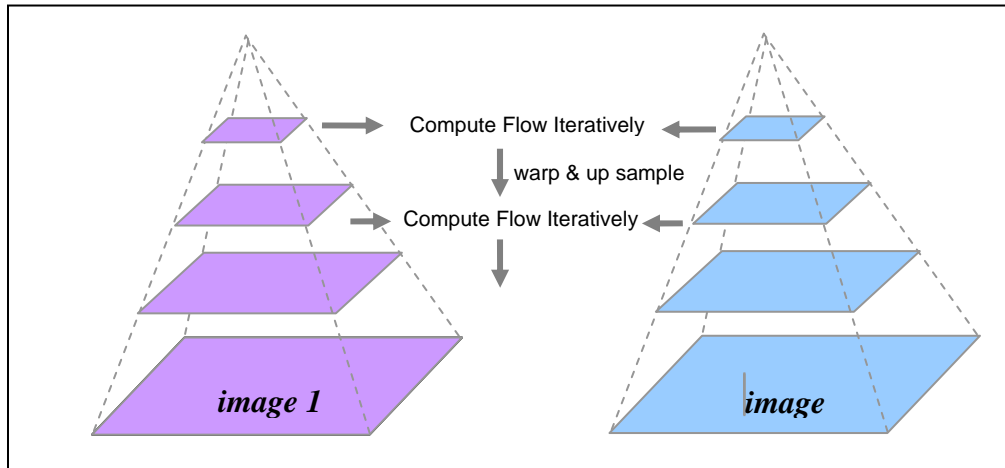


Figure 2-2: Gaussian pyramids of image 1 and image 2

## 2.4 Camera Model

A camera model provides the relationship between an object 3-dimensional (3D) position and its corresponding pixel position in 2-dimensional (2D) image plane. There are two groups of parameters namely the extrinsic parameters and intrinsic parameters. The extrinsic parameters are not constant and depend on the camera orientation while the intrinsic parameters are orientation independent.

Extrinsic parameters are values of the placement and orientation of the camera, i.e. a rotation matrix and translation vector. Intrinsic parameters are the important parameters since they will be used in calculation of motion estimation application and camera orientation is fixed. The intrinsic parameters are listed below:

- a) Focal length :  $f_c = [f_1 \ f_2]'$
- b) Principal point :  $c_c = [c_1 \ c_2]'$
- c) Skew coefficient :  $\alpha_c$
- d) Lens distortion :  $kc = [k_1 \ k_2 \ k_3 \ k_4 \ k_5]'$

Note: In this particular project, only focal length,  $f$  is used in motion estimation calculation.

Figure 2-3 shows a camera standard coordinate system. Projected point  $i_p (x_p, y_p)$  of  $P_o (X_c, Y_c, Z_c)$  on the image plane is given as:

$$\begin{bmatrix} x_p \\ y_p \end{bmatrix} = \begin{bmatrix} f_1(x_{d1} + \alpha_c x_{d2}) + c_1 \\ f_2 x_{d2} + c_2 \end{bmatrix} \quad (2.8)$$

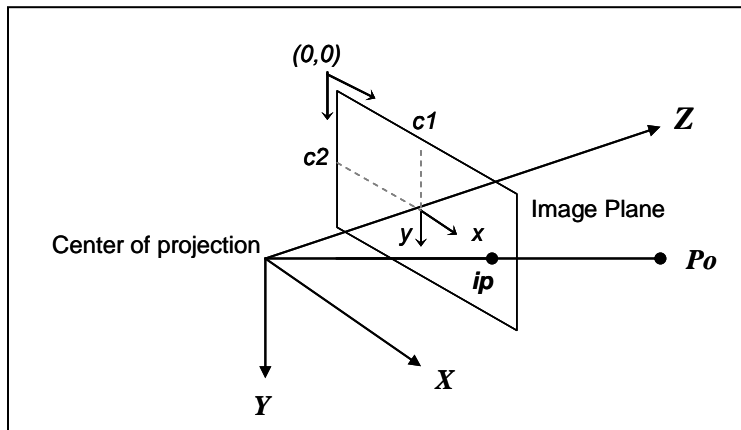


Figure 2-3: Camera standard coordinate system.

For camera velocity estimation, the system setup is prepared as shown in Figure 2-2.

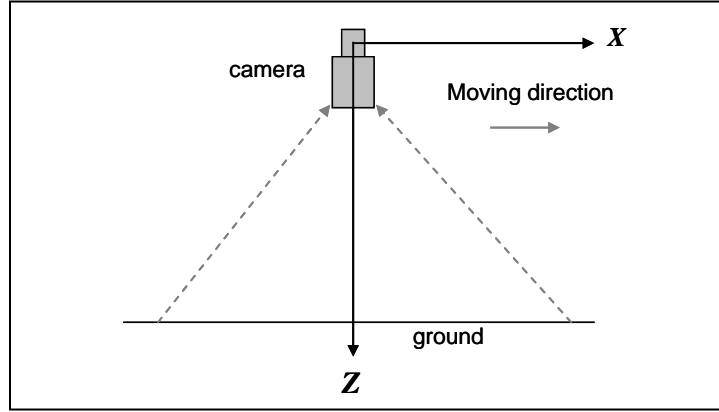


Figure 2-4: Camera system setup

The optical axis of the camera is perpendicular to the ground plane. Therefore, we can further assume the distortion and skew coefficient are negligible. Equation 2.6 can be represented as:

$$\begin{bmatrix} x_p \\ y_p \end{bmatrix} = \begin{bmatrix} f_1 \frac{X_c}{Z_c} + c_1 \\ f_2 \frac{X_c}{Z_c} + c_2 \end{bmatrix} \quad (2.9)$$

Differentiate Equation 2.7 with respect to time, the velocity of the camera in X-Y plane is given as:

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} \frac{Z_c v_x}{f_1} + \frac{1}{f_1} (x_p - c_1) V_z \\ \frac{Z_c v_y}{f_2} + \frac{1}{f_2} (x_p - c_2) V_z \end{bmatrix} \quad (2.10)$$

Then if the ground is flat, i.e.  $Z_c$  is constant, Equation 2.8 becomes:

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} \frac{Z_c v_x}{f_1} \\ \frac{Z_c v_y}{f_2} \end{bmatrix} \quad (2.11)$$



whereby:

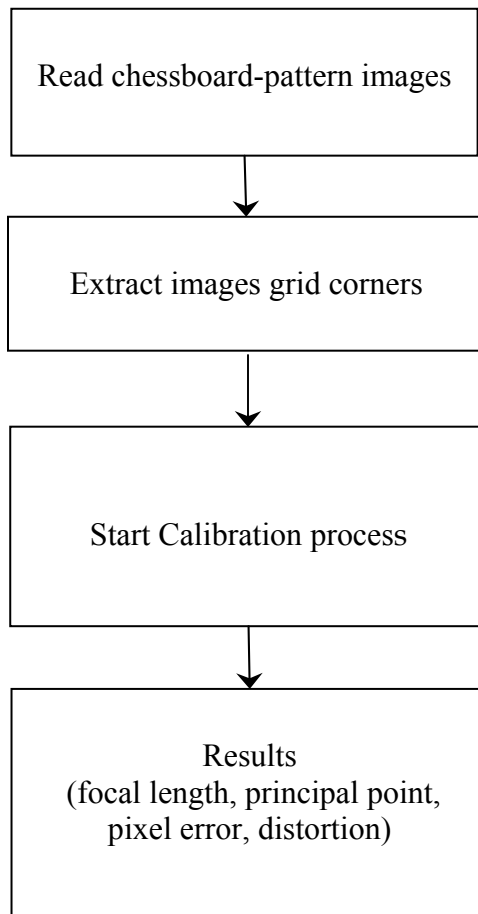
- $V_x$  and  $V_y$  are the camera velocities in  $X$  and  $Y$  directions respectively.
- $Z_c$  is the distance between the feature points on the ground and the centre of projection of the camera.
- $v_x$  and  $v_y$  are the image velocity,  $v_x = \Delta x F$  and  $v_y = \Delta y F$ , where  $F$  is camera's frame rate and  $(\Delta x, \Delta y)$  is the optical flow.
- Focal length:  $f_c = [f_1, f_2]'$ . Obtained from camera calibration.
- Principal point:  $c_c = [c_1, c_2]'$ . Obtained from camera calibration.

## CHAPTER 3

### METHODOLOGY / PROJECT WORK

#### 3.1 Camera Calibration

The intrinsic parameters of a camera can be determined by using Camera Calibration Toolbox for Matlab [2] and several chessboard-pattern images taken by the Philips Webcam SPC 600 NC camera. Figure 3-1 shows the calibration toolbox's main window. The procedures for the calibration process are noted as follow:



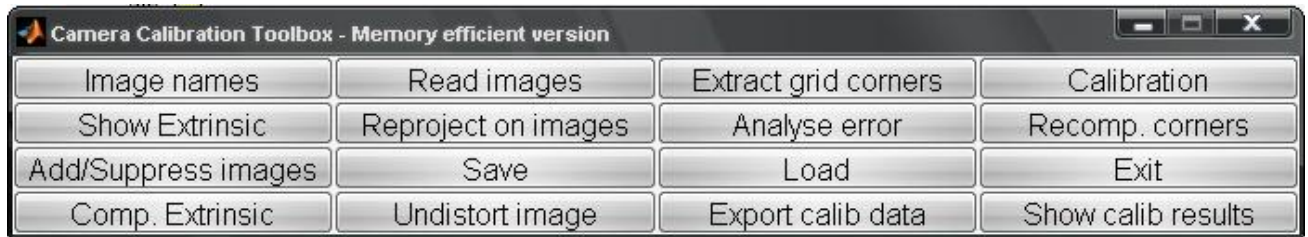


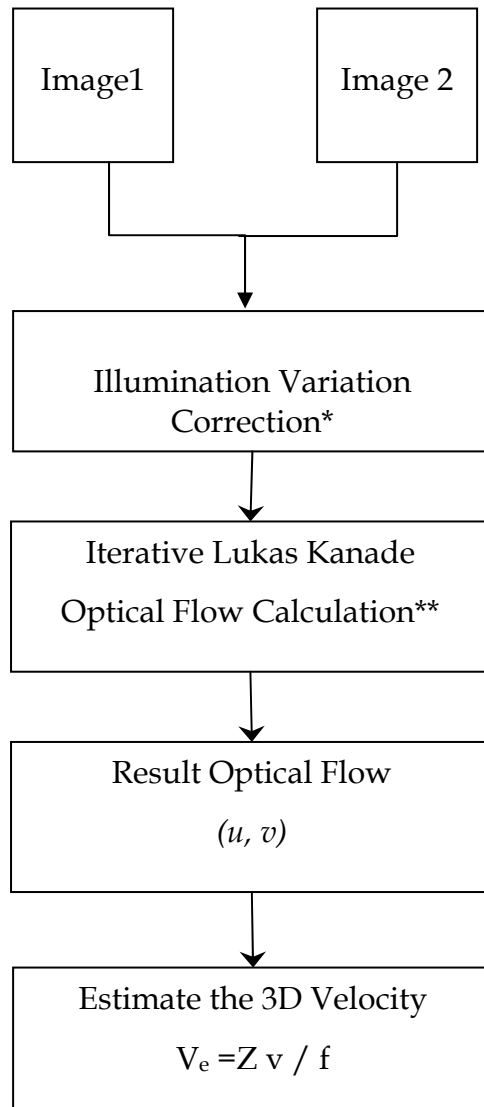
Figure 3-1: Camera Calibration Toolbox (J.Y Bouguet)

The MATLAB application has to read the pattern of the chessboard images before it can extract the grid corner of the images. The application will then start the calibration process to compute the desired result parameters such as focal length, principal point, pixel error and distortion.

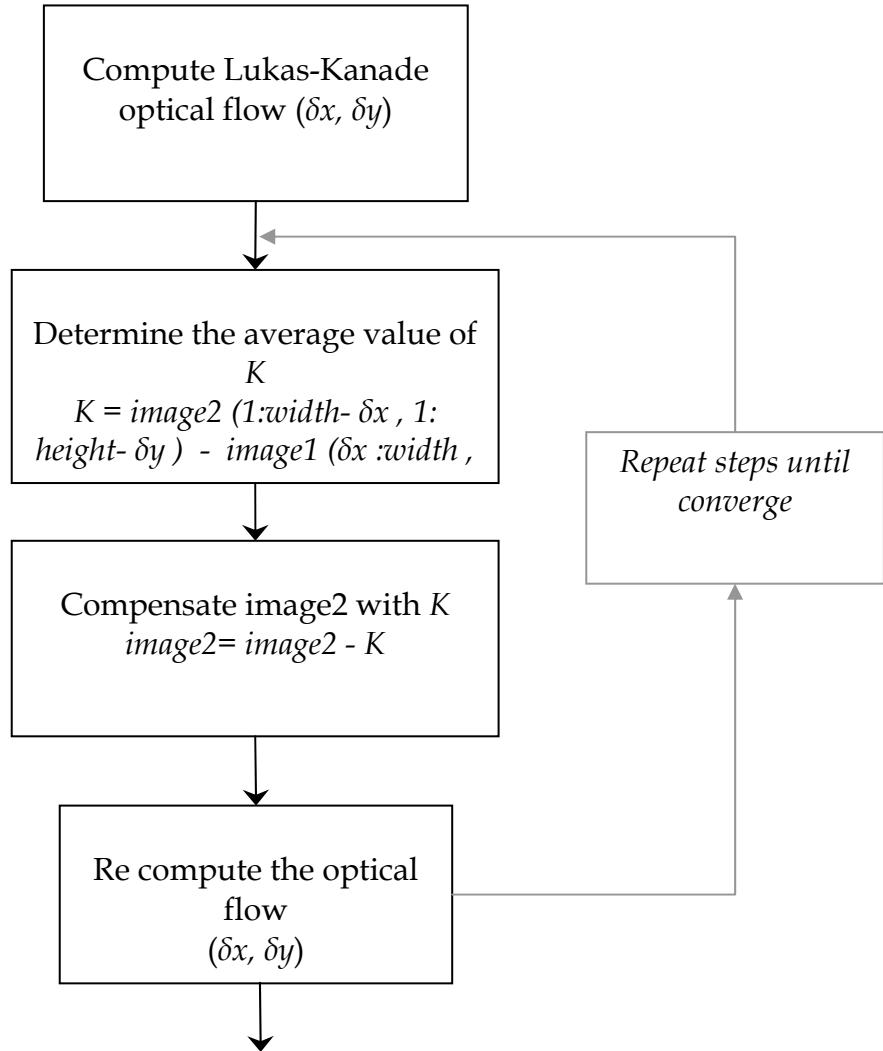
## 3.2 Optical Flow Algorithm

### 3.2.1 Robust Optical Flow Algorithm

Image 1 and image 2 are the input for the algorithm. It will undergo illumination variation correction to compensate the effect of change in one frame in accordance to the other frame. The algorithm will start the calculation process using Iterative Lukas Kanade. Result of the calculation then will be used to estimate the 3D velocity. The procedures for the optical flow algorithm are noted as follow:

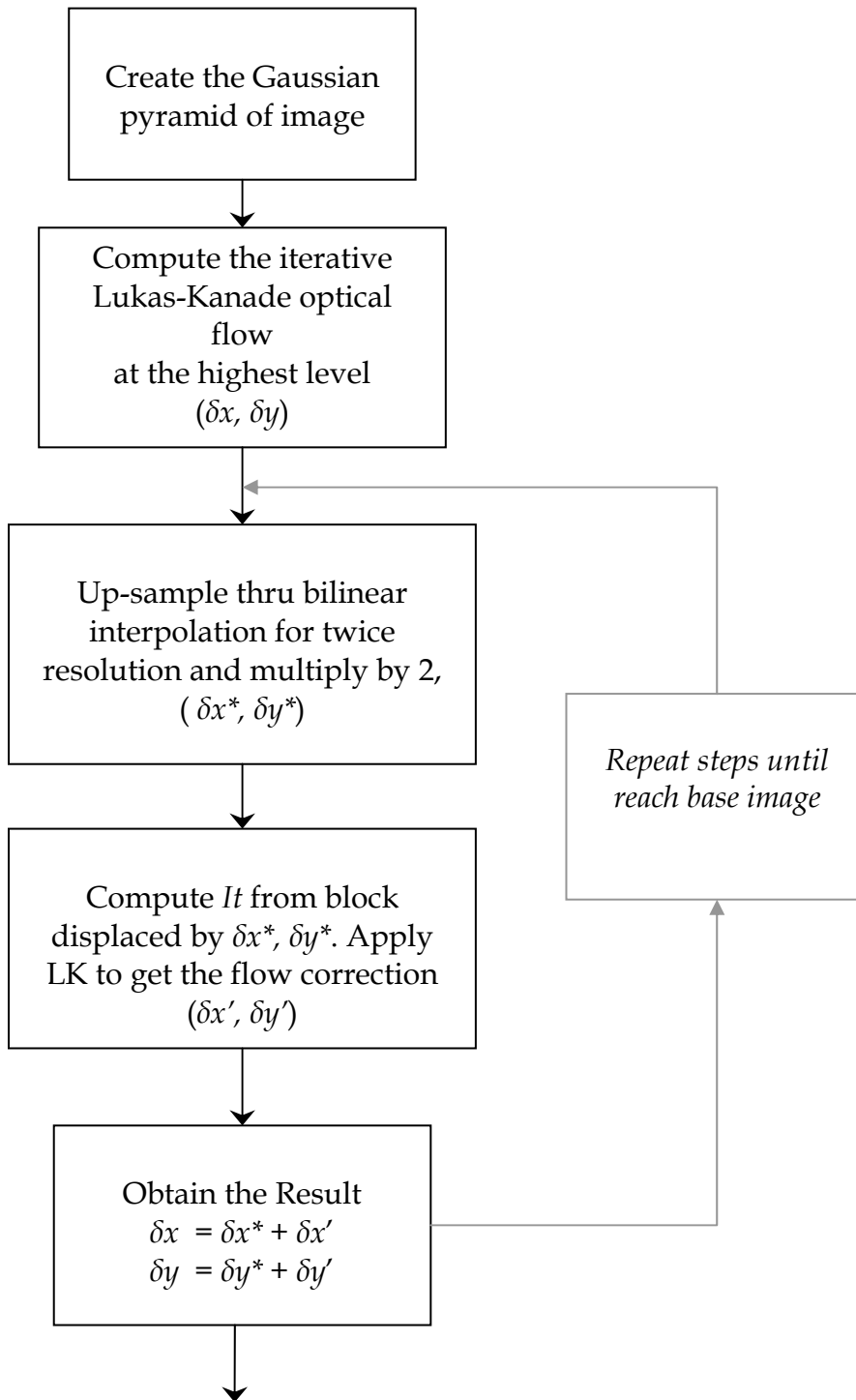


### 3.2.2 Illumination Variation Correction\*



In the illumination variation correction process, the average value,  $K$  is estimated by calculating the average difference of intensity values between the second frame and the first frame. The steps are repeated until it converges.

### 3.2.3 Iterative Lukas-Kanade Optical Flow \*\*



In the Iterative Lukas Kanade process, the Gaussian pyramid of image is created. The calculation will compute the original image at coarse scale before interpolate the result and multiply by 2. Lukas Kanade algorithm is used to get flow correction. The steps repeated until reach the base image.

## CHAPTER 4

### RESULTS AND DISCUSSION

Experiments are performed to check the feasibility, precision and robustness of the proposed algorithm. The first experiment, the camera intrinsic parameters are determined through calibration.

#### 4.1 Camera Calibration Result

Camera Calibration Toolbox has been used to calibrate Philips Webcam SPC 600 NC camera. This step is carried out in order to get the camera's focal length value. The parameters obtained after the experiment are:

```
Calibration results after optimization (with uncertainties):

Focal Length:      fc = [ 657.64375  658.04112 ] ± [ 0.40243  0.43056 ]
Principal point:   cc = [ 303.19239  242.55566 ] ± [ 0.81860  0.74880 ]
Skew:              alpha_c = [ 0.00000 ] ± [ 0.00000 ] => angle of pixel axes = 90.00000 ± 0.00000 degrees
Distortion:        kc = [ -0.25610  0.13089  -0.00019  0.00004  0.00000 ] ± [ 0.00314  0.01251  0.00017  0.00017  0.00000 ]
Pixel error:       err = [ 0.15297  0.13965 ]

Note: The numerical errors are approximately three times the standard deviations (for reference).

>> |
```

Figure 4-1: Camera calibration result



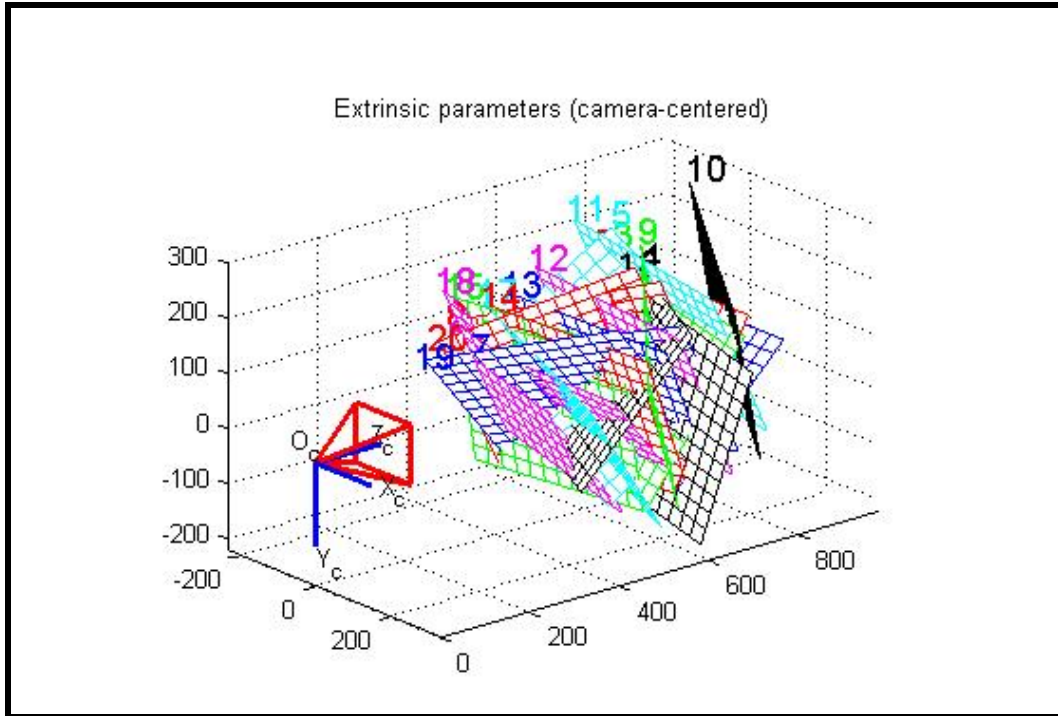


Figure 4-2: Image of camera calibration (camera-centered)

Notice that the coefficient  $\alpha_c$  and the 6<sup>th</sup> order radial distortion coefficient  $k_c$  have not been estimated (this is default mode). Therefore, the angle between the x and y axes is 90 degrees. In most practical situations, this is a very good assumption [2]. From the result, the value of focal length of the camera is approximately 658 mm.

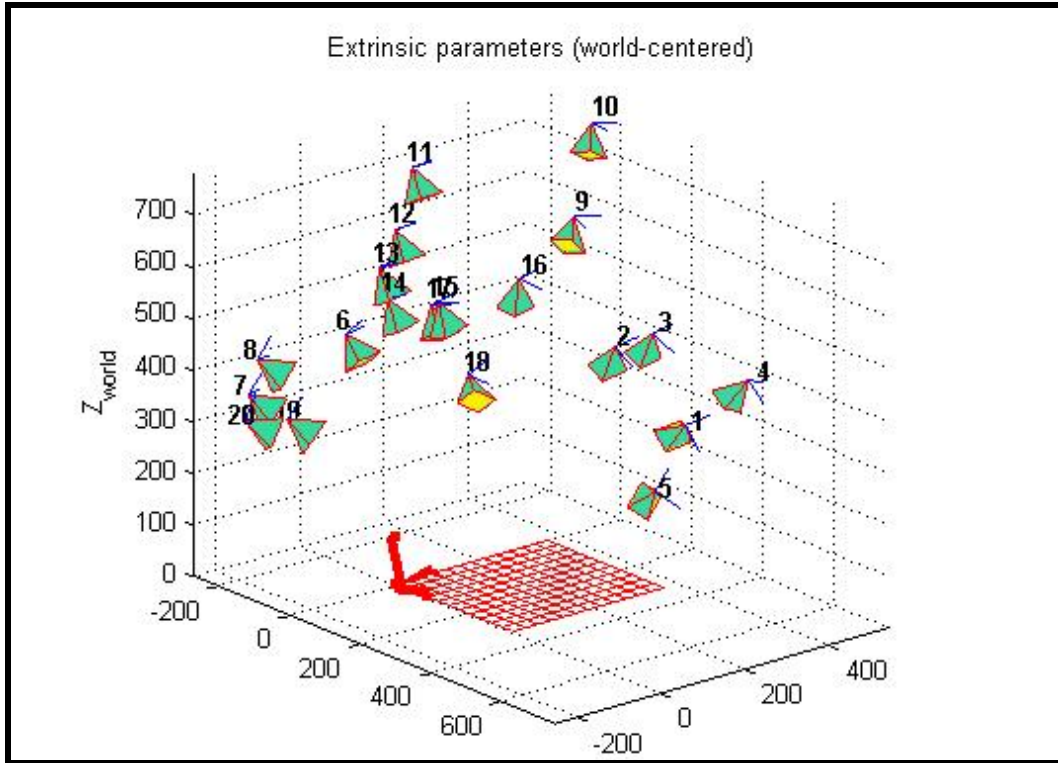


Figure 4-3: Image of camera calibration (world-centered)

## 4.2 Optical Flow Algorithm Performance

### 4.2.1 Synthetic Motion Test

This test is carried out in order to verify the functionality of the algorithm to estimate motion. The performance of the algorithm is measured by cropping 2 features windows of  $60 \times 60$  pixels<sup>2</sup> from an image to represent  $I_i$  be the grayscale image at time  $t_i$  and  $I_{i+1}$  be the grayscale image at time  $t_{i+1}$  (refer to Figure 4-1). During this time interval, let the image be translated by  $d = (\delta x, \delta y)$ . In addition, the second image has been imposed an increases in brightness to simulate the illumination variation condition.

The calculated optical flow  $(\underline{\delta x}, \underline{\delta y})$  is compared to the actual displacement of pixels between the images and results are shown in Table 4-1. The optical flow also calculated

using original iterative Lukas-Kanade algorithm (i.e. without illumination correction) to show how illumination can effect the result (refer Table 4-2).

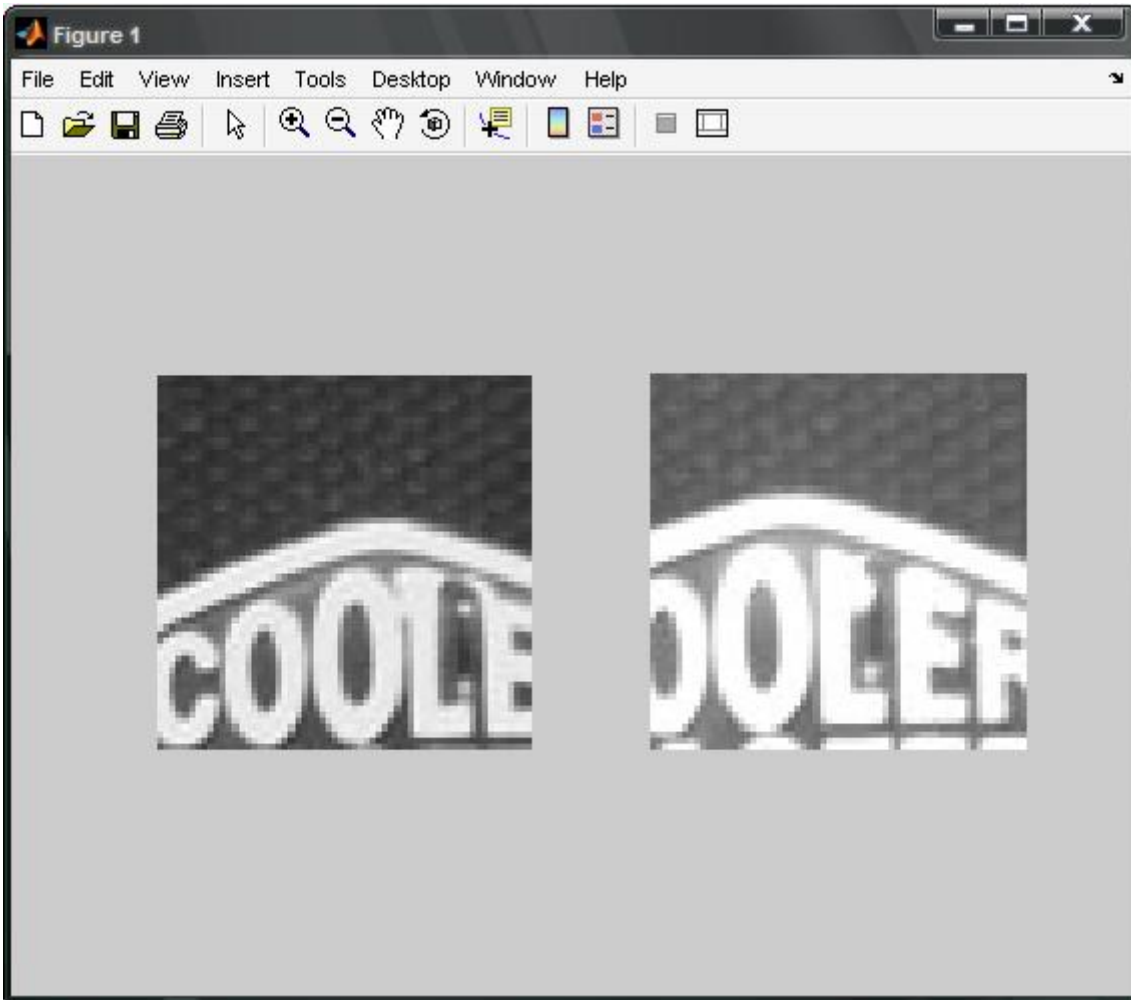


Figure 4-4: Image sequence for optical flow algorithm calculation which has undergone illumination effect

Table 4-1: The result of calculated optical flow with compared to actual displacement

Actual displcaement (unit pixel)		Calculated disp. (unit pixel)		Error measure	
$\delta_x$	$\delta_y$	$\delta_x$	$\delta_y$	$\delta_x$	$\delta_y$
0	0	0.0000	0.0000	-	-
1	0	0.9995	0.1433	0.05	-
2	0	2.0457	0.1268	2.89	-
3	0	3.0198	-0.1359	0.66	-
4	0	3.9699	-0.0252	0.75	-
5	0	4.9236	0.1170	1.53	-
0	1	-0.0071	1.0541	-	5.41
1	1	1.0136	1.1813	1.36	18.13
2	1	2.0768	1.1698	3.84	16.98
3	1	2.9785	1.1543	0.72	15.43
4	1	4.0158	1.1288	0.40	12.88
5	1	5.0864	1.1925	1.73	19.25
0	2	0.0017	2.1082	-	5.41
1	2	1.0550	2.1179	5.50	5.90
2	2	2.0856	2.1577	4.28	7.89
3	2	3.0977	2.1313	3.23	6.57
4	2	4.0544	2.1874	1.36	9.37
5	2	4.9965	2.0981	0.07	4.91
0	3	0.0217	2.9499	-	1.67
1	3	1.0114	3.0844	1.14	2.81
2	3	2.0359	3.1151	1.80	3.84
3	3	3.0099	2.8073	0.33	6.42
4	3	4.0768	3.1632	1.92	5.44
5	3	5.0454	3.0424	0.91	1.41
0	4	0.0213	4.0105	-	0.26
1	4	1.0697	4.0283	6.97	0.71
2	4	2.0982	4.1481	4.91	3.70
3	4	3.0127	3.8421	0.42	3.95
4	4	3.9811	3.9811	0.47	0.47
5	4	5.0663	4.1544	1.33	3.86
0	5	0.0235	5.0342	-	-0.68
1	5	1.0676	5.0881	6.76	1.76
2	5	1.9825	5.0677	0.88	1.35
3	5	3.0512	5.0399	1.71	0.80
4	5	3.9905	5.0377	0.24	0.75
5	5	5.0307	5.3505	0.61	7.01
6	8	6.1287	8.6766	2.15	8.46
8	6	8.1003	6.0477	1.25	7.95
10	10	9.6542	10.4288	3.46	4.29

Table 4-2: The results of calculated optical flow with illumination effect using original iterative Lukas-Kanade algorithm (i.e. without illumination correction)

Actual displacement (unit pixel)		Calculated disp. (unit pixel)		Error measure	
$\delta_x$	$\delta_y$	$\delta_x$	$\delta_y$	$\delta_x$	$\delta_y$
0	0	-1.1811	2.6420	-	-
1	0	-0.4213	2.6328	142.13	-
2	0	-0.7551	2.9261	137.76	-
3	0	-0.8123	2.9203	127.08	-
4	0	-0.3622	2.7739	109.06	-
5	0	-0.8605	2.8054	117.21	-
0	1	-0.7339	3.3298	-	232.98
1	1	-0.2330	3.5920	123.30	259.20
2	1	-1.7740	3.2924	188.70	229.24
3	1	-0.6458	3.2224	121.53	222.24
4	1	-0.0847	3.4046	102.12	240.46
5	1	-0.3630	3.3522	107.26	235.22
0	2	-0.0474	3.9030	-	95.15
1	2	0.0886	4.0680	91.14	103.40
2	2	0.3545	4.1375	82.28	106.88
3	2	-0.4872	4.0798	116.24	103.99
4	2	1.3263	4.1866	66.84	109.33
5	2	0.4397	4.2066	91.21	110.33
0	3	0.5983	5.7594	-	91.98
1	3	0.7803	5.7042	21.97	90.14
2	3	0.5173	5.6206	74.14	87.35
3	3	1.1357	5.5185	62.14	83.95
4	3	0.9154	5.6824	77.12	89.41
5	3	0.9860	5.8954	80.28	96.51
0	4	2.3125	7.6629	-	91.57
1	4	2.5872	8.2083	158.72	105.21
2	4	3.3844	8.1059	69.22	102.65
3	4	2.5118	8.0370	16.27	100.93
4	4	3.2326	8.0571	19.19	101.43
5	4	2.6520	7.8752	46.96	96.88
0	5	3.0548	9.2564	-	85.13
1	5	2.8547	9.8576	185.47	97.15
2	5	2.6632	9.0188	33.16	80.38
3	5	3.6574	9.3584	21.91	87.17
4	5	3.8812	9.8711	2.97	97.42
5	5	3.1542	9.0548	36.92	81.01

### 4.2.2 Real Motion Test

The algorithm performance as velocity estimator needs to be tested and analyzed by using real image sequence. Camera used has a maximum capacity of grabbing 30 frames per second (fps) and frame size of  $176 \times 144$  pixels<sup>2</sup> is used for this test. Then, captured images are cropped into  $60 \times 60$  pixels<sup>2</sup> to minimize computational time and the optical flow algorithm is applied. The result from optical flow algorithm for 2 difference distances between the feature points on the ground and the centre of projection of the camera,  $Z$  is shown in Table 4-3.

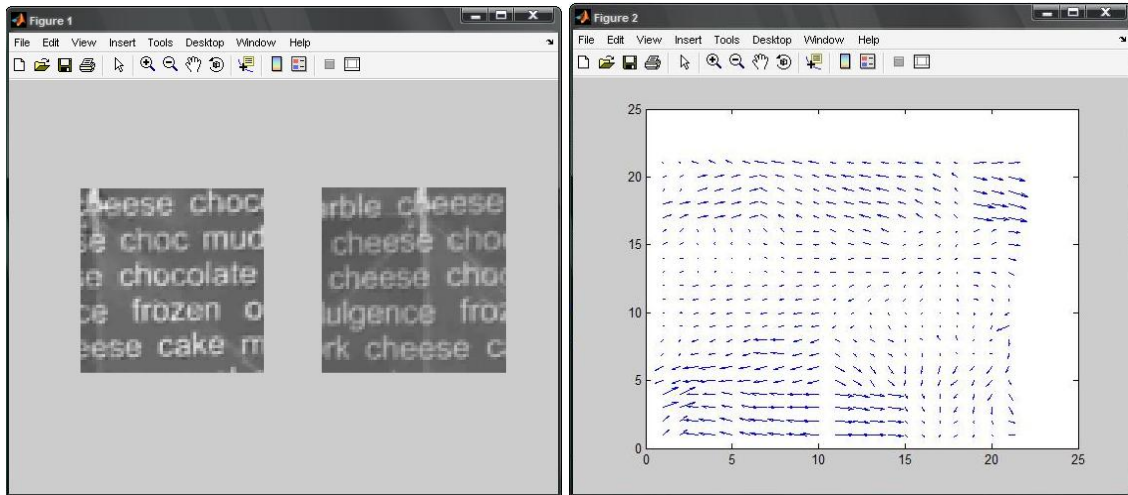


Figure 4-5: Optical flow result for real motion estimation

Table 4-3: The result of estimated velocity,  $v_e$  compared to average value,  $v_c$

Z value	Frame per second	Actual velocity (cm/s)	Optical flow (pixel)	Estimated velocity (cm/s)		Error (%)
				Value	Average	
10	30	2	6.8958	1.6996	1.7745	11.28
			7.5621	1.8637		
			7.1423	1.7602		
		1	3.5772	0.8816	0.8422	
			3.0263	0.7458		
			3.6488	0.8992		
	15	2	10.1592	1.2519	1.2787	36.07
			10.5448	1.2994		
			10.4277	1.2849		
		1	6.7721	0.8345	0.8273	
			6.3372	0.7809		
			7.0329	0.8666		
20	30	2	3.5688	1.7590	1.7345	13.28
			3.4137	1.6826		
			3.5746	1.7619		
		1	2.2598	1.1138	0.9219	
			1.7864	0.8805		
			1.5648	0.7713		
	15	2	6.0188	1.4833	1.5372	23.14
			5.8473	1.4410		
			6.8472	1.6874		
		1	3.5864	0.8839	0.9034	
			3.7485	0.9238		
			3.6621	0.9025		

### 4.3 Discussion

Based on the result, the optical flow algorithm has performed its functionality up to initial expectation. Under uniform illumination changes, the algorithm is capable of estimating the optical flow with error less than 10% in  $\delta_x$  and less than 20% in  $\delta_y$ . When dealing with large displacement ‘motion’, the algorithm successfully estimate the optical flow up to 10 units (pixels).

From the result in Table 4-2, it is clearly shown that without considering illumination effect, the optical flow algorithm will suffers severely from inaccurate results when dealing with motion with illumination variation. As presented in the robust optical flow algorithm, the illumination effect need to be compensated first before proceed with further optical flow calculation. This is important step since the principle of optical flow calculation is based on the intensity values of pixels in a sequence of images.

However, it was found that, in some occasions the estimation in x-direction has shown better result as compared to the y-direction estimation. This situation arises from the hierarchical matching-based methods (coarse-to-fine estimation) itself, in which they provide an efficient estimation for large displacements but are less accurate [12]. The accuracy can be improved by employing coarse-to-over fine approach as presented in [13] but it will affect the algorithm in term of execution time. This approach leads to sharpening of the flow edges and sub-pixel motion is observed to be more accurately estimated as well. Alternatively, by utilizing the high speed imaging capability of CMOS image sensors, more accurate optical flow with wide range of scene velocities in real time [12].

For the real image sequence optical flow test, the estimated value of velocity,  $V_e$  show ability of computing the optical flow. Two different distances between the feature points on the ground and the centre of projection of the camera,  $Z$  are used in this test and the result are compared. Both didn’t produce a very good but the result did show that the more distance between the feature points on the ground and the centre of projection of the



camera, Z produced better result. This is because the setup of image acquisition processes very poor and also the effect of the focal length of the camera. The different value of frame rate (fps) is used in the test to observe the effect to the estimation. From the result, it has given significant effect to result and this shows the more frame rate can produce more accurate result [12]. The velocity calculation is made based on Equation 2.11.

Therefore, this algorithm has proved its robust performance in motion estimation both under synthetic motion and real motion.

## **CHAPTER 5**

### **CONCLUSION AND RECOMENDATION**

The aim of this project is to extract motion from picture sequence using optical flow algorithm. In optical flow calculation, it is necessary to consider illumination effect as it could change with respect to time and the algorithm has to have capability to handle larger range of motion.

As presented in result section, the algorithm has performed well in estimating synthetic motion condition for both motion under uniform illumination variation and large displacement motion. In order to confirm the applicability and its performance in dealing with real image motion, experiments have been conducted and the results have been analyzed. From the findings, the robust optical flow algorithm for real images shows a very fair result but also a positive indication on its practical functionality in real life situation.

For future works, focus should be made on the performance in dealing with real image motion to produce more accurate result so that the algorithm will be more relevant for the real applications. This is important since the result from this report for the real motion is lower than expected. Apart from that, it also necessary to improve the algorithm in term of execution time which is important in order to be relevant for real time application.

## REFERENCES

- [1] Pished Bunnun, Yahya H Zweiri, Lakmal D Seneviratne and Kaspar Althoefer, 1997, *Feasibility of Velocity Estimation for All Terrain Ground Vehicles using an Optical Flow Algorithm*, Marcel Dekker, Inc.
- [2] J.Y. Bouguet, *Camera Calibration Toolbox for Matlab*, September 20, 2006  
<<http://www.vision.caltech.edu/>>
- [3] B.K.P. Horn and B.G. Schunck, 1981, *Determining Optical Flow Artificial Intelligence*, Wiley.
- [4] J.L. Barron, D.J. Fleet, and S.S. Beauchemin, 1994, *Performance of optical flow techniques*, Int. J. Comput.
- [5] B.D. Lucas and T. Kanade, 1981, *An Iterative Image Registration Technique with an Application to Stereo Vision*, DARPA Image Understanding Workshop.
- [6] Yucel Altunbasak, Russel M. Merasereau and Andrew J.Patti, 1990, *A Fast Parametric Motion Estimation Algorithm with Illumination and Lens Distortion Correction*, CRC Press, Taylor & Francis Group.
- [7] Lukas Kanade Method, April 12, 2007, <[http://en.wikipedia.org/wiki/Lucas\\_Kanade](http://en.wikipedia.org/wiki/Lucas_Kanade)>
- [8] Samuel Ben-Ezra, VideoControl – GUI to control image acquisition, February 12, 2008, <<http://www.mathworks.com/matlabcentral/fileexchange/loadAuthor.do?objectType=author&objectid=1094722>>
- [9] M. Black, D.J. Fleet, and Y.Yacoob, 1998, *A framework for modelling appearance change in image sequences*, Int. Conf. on Computer Vision.

- [10] Garcia C., Tziritas G., 2001, "Translational Motion Estimation from 2D displacements", (to appear in) Proceedings of the IEEE International Conference on Image Processing (ICIP2001), Thessalonique, Greece.
- [11] Aleix M. Martinez, 1998, *Image Processing*, Wiley.
- [12] Sukhwan Lim, Abbas El Gamal, 1996, "*Optical flow estimation using high frame rate sequences*", Stanford University, Department Of Electrical Engineering.
- [13] Tomer Amiaz\*, Eyal Lubetzky, Nahum Kiryati\*, 1997, "*Coarse to Over-Fine Optical Flow Estimation*", Tel Aviv University, \*School of Electrical Engineering , School of Computer.

## APPENDICES

# APPENDIX A

## MATLAB SOURCE CODE

### Illumination effect

```
function [new_im1,new_im2] = illumination(im1, im2,numLevels,
windowSize);

ori_image1=im1;
ori_image2=im2;

previous_k =0;

for i=1:3

    %Build Pyramids
    pyramid1 = im1;
    pyramid2 = im2;

    for i=2:numLevels
        im1 = reduce(im1);
        im2 = reduce(im2);
        pyramid1(1:size(im1,1), 1:size(im1,2), i) = im1;
        pyramid2(1:size(im2,1), 1:size(im2,2), i) = im2;
    end;

    image1 = pyramid1(1:(size(pyramid1,1)/(2^(numLevels-1))),
1:(size(pyramid1,2)/(2^(numLevels-1))), numLevels);
    image2 = pyramid2(1:(size(pyramid2,1)/(2^(numLevels-1))),
1:(size(pyramid2,2)/(2^(numLevels-1))), numLevels);

    [height,width]=size(image1);

    [u,v] = LucasKanade(image1, image2, windowSize);
    u1=u(3:size(u,1)-2, 3:size(u,2)-2);
    v1=v(3:size(v,1)-2, 3:size(v,2)-2);

    delta_x= -mean(mean(u1))';
    delta_y= mean(mean(v1))';

    X = abs(delta_x);
    Y = abs(delta_y);

    if (delta_x >= 0 && delta_y >= 0)    % positive displacement
        k = image2(Y+2:height-1,2:width-X-1) - image1(2:height-Y-
1,X+2:width-1);

    elseif (delta_x < 0 && delta_y < 0)
        k = image2(2:height-Y-1,X+2:width-1) - image1(Y+2:height-
1,2:width-X-1);

    elseif (delta_x >= 0 && delta_y < 0)
```

```

        k = image2(2:height-Y-1,2:width-X-1) - imagel(Y+2:height-
1,X+2:width-1);

    else
        k = image2(Y+2:height-1,X+2:width-1) - imagel(2:height-Y-
1,2:width-X-1);

    end

    avg_k=mean(mean(k))';
    avg_k= avg_k + previous_k;
    im1= ori_image1;
    im2= ori_image2 - avg_k;
    previous_k = avg_k;
end

new_im1= im1;
new_im2= im2;

```

### Lukas Kanade

```

function [u,v,cert] = LucasKanadeRefined(uIn, vIn, im1, im2);

uIn = round(uIn);
vIn = round(vIn);

u = zeros(size(im1));
v = zeros(size(im2));

%to compute derivatives, use a 5x5 block...
% take the middle 3x3 block as derivative
for i = 3:size(im1,1)-2
    for j = 3:size(im2,2)-2

        curIm1 = im1(i-2:i+2, j-2:j+2);
        lowRindex = i-2+vIn(i,j);
        highRindex = i+2+vIn(i,j);
        lowCindex = j-2+uIn(i,j);
        highCindex = j+2+uIn(i,j);

        if (lowRindex < 1)
            lowRindex = 1;
            highRindex = 5;
        end;

        if (highRindex > size(im1,1))
            lowRindex = size(im1,1)-4;
            highRindex = size(im1,1);
        end;

        if (lowCindex < 1)
            lowCindex = 1;
            highCindex = 5;
        end;
    end
end

```

```

    if (highCindex > size(im1,2))
        lowCindex = size(im1,2)-4;
        highCindex = size(im1,2);
    end;

    if isnan(lowRindex)
        lowRindex = i-2;
        highRindex = i+2;
    end;

    if isnan(lowCindex)
        lowCindex = j-2;
        highCindex = j+2;
    end;

    curIm2 = im2(lowRindex:highRindex, lowCindex:highCindex);

    [curFx, curFy, curFt]=ComputeDerivatives(curIm1, curIm2);

    curFx = curFx(2:5, 2:5);
    curFy = curFy(2:5, 2:5);
    curFt = curFt(2:5, 2:5);

    curFx = curFx(:);
    curFy = curFy(:);
    curFt = -curFt(:);

    A = [curFx curFy];

    U = pinv(A'*A)*A'*curFt;

    u(i,j)=U(1);
    v(i,j)=U(2);

    cert(i,j) = rcond(A'*A);

end;
end;

u = u+uIn;
v = v+vIn;

```

### Lukas Kanade 2

```

function [u, v] = LucasKanade(im1, im2, windowSize);

fx = conv2(im1,0.25* [-1 1; -1 1]) + conv2(im2, 0.25*[-1 1; -1 1]);
fy = conv2(im1, 0.25*[-1 -1; 1 1]) + conv2(im2, 0.25*[-1 -1; 1 1]);
ft = conv2(im1, 0.25*ones(2)) + conv2(im2, -0.25*ones(2));

% make same size as input
fx=fx(1:size(fx,1)-1, 1:size(fx,2)-1);
fy=fy(1:size(fy,1)-1, 1:size(fy,2)-1);

```



```

ft=ft(1:size(ft,1)-1, 1:size(ft,2)-1);

u = zeros(size(im1));
v = zeros(size(im2));

halfWindow = floor(windowSize/2);
for i = halfWindow+1:size(fx,1)-halfWindow
    for j = halfWindow+1:size(fx,2)-halfWindow
        curFx = fx(i-halfWindow:i+halfWindow, j-halfWindow:j+halfWindow);
        curFy = fy(i-halfWindow:i+halfWindow, j-halfWindow:j+halfWindow);
        curFt = ft(i-halfWindow:i+halfWindow, j-halfWindow:j+halfWindow);

        curFx = curFx(:);
        curFy = curFy(:);
        curFt = -curFt(:);

        A = [curFx curFy];

        U = pinv(A'*A)*A'*curFt;

        u(i,j)=U(1);
        v(i,j)=U(2);
    end;
end;

u(isnan(u))=0;
v(isnan(v))=0;

```

### Optical flow

```

function opticalFlow (im1,im2)

R1 = [60 20 60 60];
R2 = [60 20 60 60];
im1 = imcrop(im1,R1);
im2 = imcrop(im2,R2);

numLevels=3; windowSize=3; iterations=1;

% change color image to grayscale
if size(im1,3)==3
    im1 = rgb2gray(im1);
    im2 = rgb2gray(im2);
end

im1=im2double(im1);
im2=im2double(im2);
if (rem(size(im1,1), 2) ~= 0)
    % warning('image will be cropped in height!');
    im1 = im1(1:(size(im1,1) - rem(size(im1,1), 2^(numLevels - 1))),
:);

```

```

        im2 = im2(1:(size(im1,1) - rem(size(im1,1), 2^(numLevels - 1))),
:);
end;

if (rem(size(im1,2), 2) ~= 0)
    % warning('image will be cropped in width!');
    im1 = im1(:, 1:(size(im1,2) - rem(size(im1,2), 2^(numLevels -
1)))));
    im2 = im2(:, 1:(size(im1,2) - rem(size(im1,2), 2^(numLevels -
1)))));
end;
%
% im2=im2+0.0;
ori_image1=im1;
ori_image2=im2;

subplot(1,2,1), imshow(im1)
subplot(1,2,2), imshow(im2)
%=====
====
[im1,im2] = illumination(ori_image1, ori_image2,numLevels, windowSize);
%=====
====

%Build Pyramids
pyramid1 = im1;
pyramid2 = im2;

for i=2:numLevels
    im1 = reduce(im1);
    im2 = reduce(im2);
    pyramid1(1:size(im1,1), 1:size(im1,2), i) = im1;
    pyramid2(1:size(im2,1), 1:size(im2,2), i) = im2;
end;

% base level computation
%disp('Computing Level 1');
baseIm1 = pyramid1(1:(size(pyramid1,1)/(2^(numLevels-1))),
1:(size(pyramid1,2)/(2^(numLevels-1))), numLevels);
baseIm2 = pyramid2(1:(size(pyramid2,1)/(2^(numLevels-1))),
1:(size(pyramid2,2)/(2^(numLevels-1))), numLevels);
[u,v] = LucasKanade(baseIm1, baseIm2, windowSize);

%propagating flow 2 higher levels
for i = 2:numLevels
    %disp(['Computing Level ', num2str(i)]);
    uEx = 2 * imresize(u,size(u)*2);
    vEx = 2 * imresize(v,size(v)*2);

    curIm1 = pyramid1(1:(size(pyramid1,1)/(2^(numLevels - i))),
1:(size(pyramid1,2)/(2^(numLevels - i))), (numLevels - i)+1);
    curIm2 = pyramid2(1:(size(pyramid2,1)/(2^(numLevels - i))),
1:(size(pyramid2,2)/(2^(numLevels - i))), (numLevels - i)+1);

    [u, v] = LucasKanadeRefined(uEx, vEx, curIm1, curIm2);

```

```
    for r = 1:iterations
        [u, v, cert] = LucasKanadeRefined(u, v, curIm1, curIm2);
    end
end

u=u(10:size(u,1)-10,10:size(u,2)-10 );
v=v(10:size(v,1)-10,10:size(v,2)-10 );

figure; quiver(u(20:40,10:30), v(20:40,10:30))

%final result
u=mean(mean(u))'
v=mean(mean(v))'
```

## APPENDIX B

Pictures of calibration

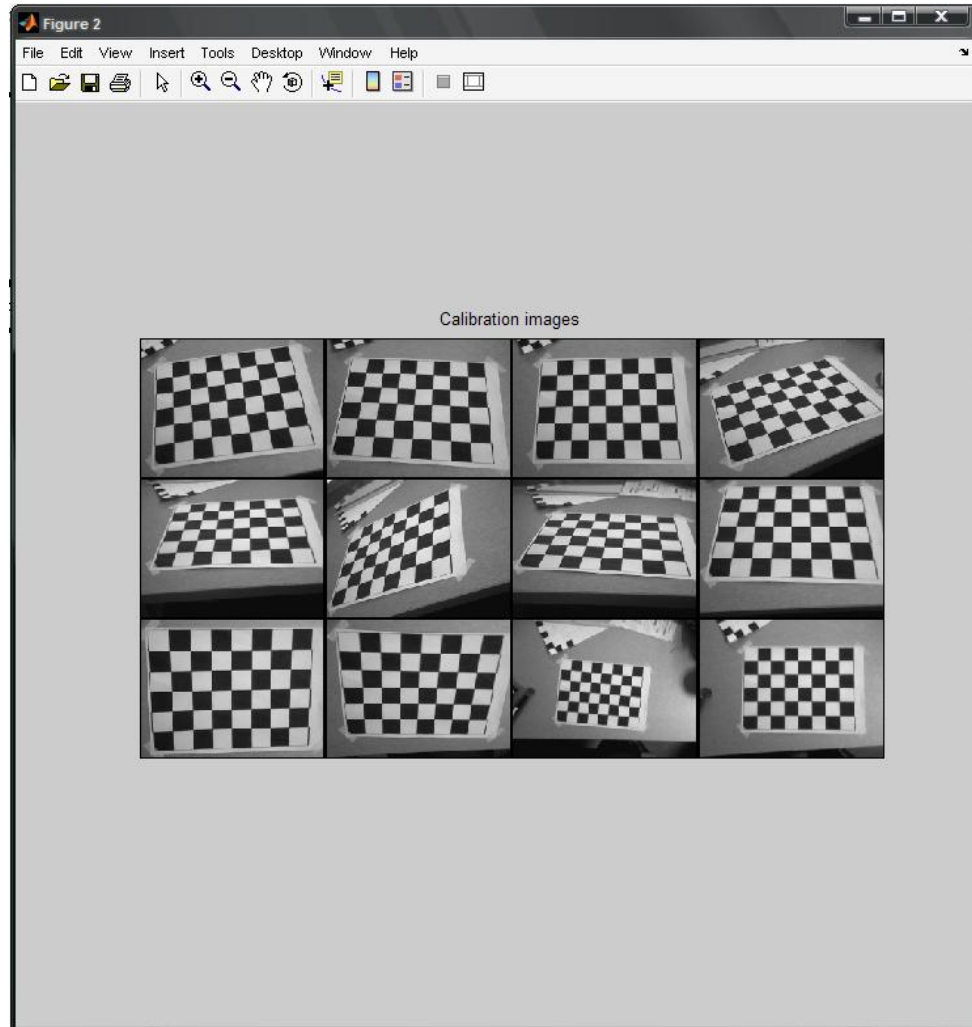


Figure A-1: Chessboard before calibrate

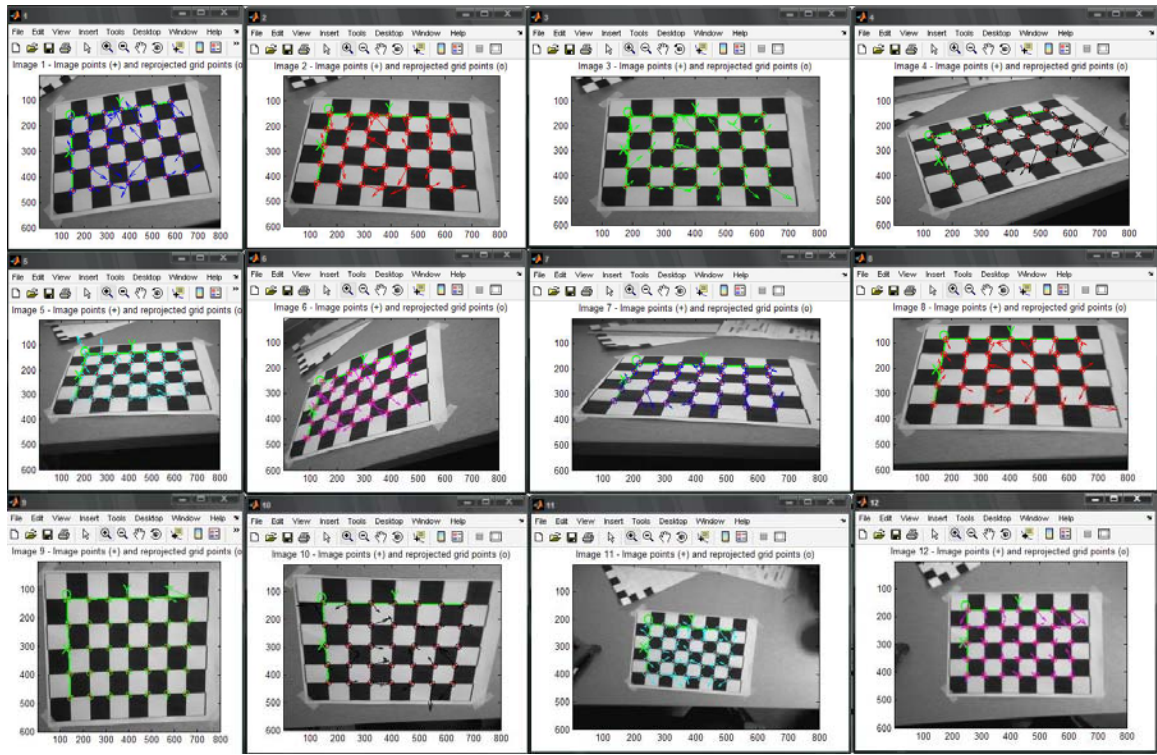


Figure A-2: Chessboard after calibrate

## FYP II GANTT CHART


**Table A-1: Suggested Milestone for the First Semester of 2-Semester Final Year Project**

No.	Detail/ Week	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	Selection of Project Topic														
2	Preliminary Research Work														
3	Submission of Preliminary Report														
4	Seminar 1 (optional)														
5	Project Work														
6	Submission of Progress Report														
7	Seminar 2 (compulsory)														
8	Project work continues														
9	Submission of Interim Report Final Draft														
10	Oral Presentation														

Suggested milestone  
 Process

**Table A-2: Suggested Milestone for the Second Semester of 2-Semester Final Year Project**

No.	Detail/ Week	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	Project Work Continue														
2	Submission of Progress Report 1														
3	Project Work Continue														
4	Submission of Progress Report 2														
5	Seminar (compulsory)														
5	Project work continue														
6	Poster Exhibition														
7	Submission of Dissertation (soft bound)														
8	Oral Presentation														
9	Submission of Project Dissertation (Hard Bound)														

Suggested milestone  
 Process

**Table A-3: Final Year Project Workflow for the First Semester of 2-Semester Final Year Project**

Process Flow Details	Week													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1) Research								<b>Semester Break</b>						
a)Optical flow														
b)Lukas kanade														
c)Robust Optical flow														
d)Camera model														
e)Motion based foreground segmentation														
e)Data gathered														
f)Report prepared														
2) Research on algorithm														
a)Optical flow														
b)Lukas kanade														
c)Robust Optical flow														
d)Motion based foreground segmentation														
e)MATLAB														
f)C++														
g)Comparison advantages and disadvantages														
3) Design the algorithm														
a)select camera														
b)Find focal length														
c)design the algorithm														
4) Test														
a)synthetic motion														
b)real motion														
c)report prepared														



**Table A-4: Suggested Milestone for the First Semester of 2-Semester Final Year Project**

Process Flow Details	Week													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1 ) Research								<b>Semester Break</b>						
a)Optical flow														
b)Lukas kanade														
c)Robust Optical flow														
d)Camera model														
e)Motion based foreground segmentation														
e)Data gathered														
f)Report prepared														
2) Research on algorithm														
a)Optical flow														
b)Lukas kanade														
c)Robust Optical flow														
d)Motion based foreground segmentation														
e)MATLAB														
f)C++														
g)Comparison advantages and disadvantages														
3 )Design the algorithm														
a)select camera														
b)Find focal length														
c)design the algorithm														
4) Test														
a)synthetic motion														
b)real motion														
c)report prepared														

