VEHICLE TRACKING SYSTEM USING OPTICAL FLOW TECHNIQUES

A project report submitted
in partial fulfillment of the requirement
for the award of the degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE ENGINEERING

By

Rahul Ponnada (Reg.No 309106410025)

Sree Vishnu Satarajupalli (Reg.No 309106410028)

Under The Esteemed Guidance of

Prof. M. Surendra Prasad Babu
Chairman Board of Studies

DEPARTMENT OF COMPUTER SCIENCE AND SYSTEMS ENGINEERING
ANDHRA UNIVERSITYCOLLEGE OF ENGINEERING(A)
ANDHRA UNIVERSITY
VISAKHAPATNAM

2013-2014

ANDHRA UNIVERSITY
COLLEGE OF ENGINEERING, VISAKHAPATNAM
ANDHRA PRADESH

DEPARTMENT OF COMPUTER SCIENCE AND SYSTEMS ENGINEERING

CERTIFICATE

This is to certify that the project work entitled **"VEHICLE TRACKING SYSTEM USING OPTICAL FLOW TECHNIQUES"** is a bonafide work done by **Rahul Ponnada** bearing Regd.No:309106410025**, Sree Vishnu Satarajupalli** bearing Regd.No:309106410028 in partial fulfillment of requirements for the award of the degree of **Bachelor of Technology in Computer Science & Engineering** in the Dept. of Computer Science & Systems Engineering during 2012-13.

Prof. M. Surendra Prasad Babu
Chairman BOS
Department,
Department of CS&SE,
A.U. College of Engineering (A),
Engineering (A),
Andhra University .

Prof. P Srinivasa Rao
Head of the

Department of CS&SE,
A.U. College of

Andhra University.

ACKNOWLEDGEMENT

We are extremely thankful to our project guide **Prof. M. Surendra Prasad Babu**, Chairman, Board Studies, Department of Computer Science and Systems Engineering, Andhra University, Visakhapatnam for his timely help, suggestions and guidance for the project dissertation.

We are very thankful to **Prof. P Srinivasa Rao, Head of the Department,** Computer Science and Systems Engineering, Andhra University, Visakhapatnam, for his co-operation and providing the required facilities throughout the course of B.Tech. We are also thankful to our Teaching, Non Teaching & Technical staff of the Dept. of CS&SE for their valuable suggestions & advices from time to time.

We are grateful to **Prof. Ch. V. R. Murthy, Principal**, Andhra University College of Engineering(A) for his encouragement throughout B.Tech programme. Lastly, we acknowledge our sincere gratitude to all those who directly and indirectly helped us.

Rahul Ponnada
&
Sree Vishnu Satarajupalli

ABSTRACT

Object tracking is one of the important techniques in digital Image processing. Speed of the moving object can be calculated from the images taken from it. The most important phase in this algorithm is the detection of the repeating objects in sequence. Lucas-Kanade Algorithm is a one of the widely used Object tracking algorithm in the 2001, which uses differential method.

By combining information from several nearby pixels, the Lucas-Kanade Algorithm can often resolve the ambiguities of the optical flow. The Lucas-Kanade Algorithm is implemented using MATLAB library, a library of programming functions mainly aimed at real time computer vision. From the tracked objects, first the pixel difference is calculated from the subsequent images. Then this measurement is converted into meter. With the known value of the meter readings in constant time intervals between the subsequent images, the velocity is calculated.

A system also be developed, which can track vehicle's route can by first identifying the vehicle using an identification number and then recording the location, time and speed of the vehicle into the database, which can later be retrieved accordingly.

TABLE OF CONTENTS

1. INTRODUCTION

As digital cameras and powerful computers have become wide spread, the number of applications using vision techniques has increased enormously. One such application that has received significant attention from the computer vision community is object tracking and motion detection. This approach can be used to detect motion of objects and perform functions like speed calculation. A new method is proposed for speed calculation system that works without prior, explicit camera calibration, and has the ability to perform speed calculation tasks in real time. The approach here makes use of optical flow calculation.

The estimation of optical flow plays a key-role in several computer vision problems, including motion detection and segmentation, frame interpolation, three-dimensional scene reconstruction, robot navigation, video shot detection, mosaic creation and video compression. Works in the field of an optical flow calculation has been conducted for more than 30 years. There are many articles that have been written on the subject of optical flow methods. Methods of an optical flow appear to be useful for segmentation of images, and also for detection of obstacles from moving objects.

The project performs real world experiments for finding the speed. It conducted tests on road using a motorcycle and a digital camera. The various results were used to generalize the speed calculation technique. This technique is proved to be very effective with minimum error rate.

2. LITERATURE SURVEY

System analysis is an activity that encompasses most of the tasks that are called computer system engineering. It is a detailed study of various operations performed by a system and their relationships within and outside of the system. Once the analysis is completed the analyst has firm understanding of what is to be done. In other words it is the reduction of an entire system by studying the various operations performed and their relationships within the system; an examination of business activity, with a view to identify problem areas and recommending alternative solutions. The objectives of the system analysis are:

- Identifying the need.
- Analyzing the existing and proposed system.
- Evaluating the feasibility study.
- Perform economical and technical analysis.
- Identifying the hardware and software requirements.
- Allocating functions to the hardware and software.
- Creating system software.

2.1. Proof of concept

Proof of concept is a short or incomplete realization (or synopsis) of a certain method or idea(s) to demonstrate its feasibility, or a demonstration in principle, whose purpose is to verify that some concept or theory is probably capable of being useful.

In the field of applied research people working on a project or proposal will often undertake internal research initially, to verify that the core ideas are functional and feasible, before going further. This sometimes includes limited performance testing. This use of proof of concept helps establish viability, technical issues, and overall direction, as

well as providing feedback for budgeting and other forms of commercial discussion and control. It is not related to a scientific proof.

2.1.1. Need of proof of concept

The need of the proof of concept is to determine the feasibility of object detection using optical flow method. This is to have some assurance that the objective can be fulfilled. By using Lucas-Kanade tracking, one should be able to detect object motion and its optical flow. This enables. Without the detection of common object in two input images, there is no guarantee of whether or not the project can be accomplished. This is the reason we require a proof of concept. Once there is a proof that the concept of object detection is possible, and then the continuation of the project is reasonable.

2.1.2. Problem Statement

Let I and J be two 2D grayscaled images. The two quantities I (x) = I(x, y) and J (x) = J(x, y) are then the grayscale value of the two images are the location $x = [x\ y]^T$, where x and y are the two pixel coordinates of a generic image point x. The image I will sometimes be referenced as the first image, and the image J as the second image. For practical issues, the images I and J are discrete function (or arrays), and the upper left corner pixel coordinate vector is $[0\ 0]^T$. Let n_x and n_y be the width and height of the two images. Then the lower right pixel coordinate vector is $[n_{x-1}\ n_{y-1}]^T$.
Consider an image point $u = [u_x\ u_y]^T$ on the first image I. The goal of feature tracking is to find the location $v = u + d = [u_x + d_x\ \ u_y + d_y]^T$ on the second image J such as I(u) and J (v) are "similar". The vector $d = [d_x\ d_y]^T$ is the image velocity at x, also known as the optical flow at x. Because of the aperture problem, it is essential to define the notion of similarity in a 2D neighborhood sense. Let $_x$ and $_y$ be two integers. We define the image velocity d as being the vector that minimizes the residual function defined as follows:

$$\epsilon(\mathbf{d}) = \epsilon(d_x, d_y) = \sum_{x=u_x-\omega_x}^{u_x+\omega_x} \sum_{y=u_y-\omega_y}^{u_y+\omega_y} \left(I(x,y) - J(x+d_x, y+d_y)\right)^2 \quad (1)$$

2.1.3. Description of the tracking algorithm

The two key components to any feature tracker are accuracy and robustness. The accuracy component relates to the local sub-pixel accuracy attached to tracking. Intuitively, a small integration window would be preferable in order not to "smoothout" the details contained in the images (i.e. Small values of x and y). That is especially required that occluding areas in the images where two patches potentially move with very different velocities.

The robustness component relates to sensitivity of tracking with respect to changes of lighting, size of image motion, etc. In particular, in order to handle large motions, it is intrusively preferable to pick a large integration window. Indeed,

Considering only equation 1, it is preferable to have $d_x < w_x$ and $dy < w_y$ (unless some Prior matching information is available). There is therefore a natural tradeoff between local accuracy and robustness when choosing the integration window size. In provide to provide a solution to that problem, we propose a pyramidal implementation of the classical Lucas-Kanade algorithm. An iterative implementation of the Lucas-Kanade optical flow computation provides sufficient local tracking accuracy.

2.1.4 Image pyramid representation

Define the pyramid representation of a generic image *I* of size n_x x n_y. Let L = 1, 2…. be a generic pyramidal level, and let I^{L-1} be the image at level L-1. Denote n_x^{L-1} and n_y^{L-1} the width and height of I^{L-1}. The image I^{L-1} is then defined as follows:

$$
\begin{aligned}
I^L(x,y) \;=\; & \frac{1}{4} I^{L-1}(2x,2y) + \\
& \frac{1}{8}\left(I^{L-1}(2x-1,2y) + I^{L-1}(2x+1,2y) + I^{L-1}(2x,2y-1) + I^{L-1}(2x,2y+1)\right) + \\
& \frac{1}{16}\left(I^{L-1}(2x-1,2y-1) + I^{L-1}(2x+1,2y+1) + I^{L-1}(2x-1,2y+1) + I^{L-1}(2x+1,2y+1)\right).
\end{aligned}
\qquad (2)
$$

The width n^L_x and height n^L_y of I^L are the largest integers that satisfy the two conditions:

$$n_x^L \leq \frac{n_x^{L-1}+1}{2} \quad (3)$$

$$n_y^L \leq \frac{n_y^{L-1}+1}{2}. \quad (4)$$

Equations (2), (3) and (4) are used to construct recursively the pyramidal representations of the two images I and J.

2.1.5. Pyramidal Feature Tracking

Following the definition of the pyramid representation equations (2), (3) and (4), the vectors u^L are computed as follows:

$$\mathbf{u}^L = \frac{\mathbf{u}}{2^L} \quad (5)$$

The overall pyramidal tracking algorithm proceeds as follows: First, the optical flow is computed at the deepest pyramid level L_m. Then, the result of that computation is propagated to the upper level L_{m-1} in a form of an initial guess for the pixel displacement (at level L_{m-1}). Given that initial guess, the refined optical flow is computed
at level L_{m-1}, and the result is propagated to level L_{m-2} and so on up to the level 0 (the original image)..

In order to compute the optical flow at level L, it is necessary to find the residual pixel displacement vector $d^L = [d^L_x \;\; d^L_y]^T$ that minimizes the new image matching error function $E^{L:}$

$$\epsilon^L(\mathbf{d}^L) = \epsilon^L(d_x^L, d_y^L) = \sum_{x=u_x^L-\omega_x}^{u_x^L+\omega_x} \sum_{y=u_y^L-\omega_y}^{u_y^L+\omega_y} \left(I^L(x,y) - J^L(x+g_x^L+d_x^L, y+g_y^L+d_y^L)\right)^2.$$

(6)

The final optical flow solution d is given by:

$$\mathbf{d} = \sum_{L=0}^{L_m} 2^L \, \mathbf{d}^L.$$

(7)

2.2.Study of existing system

2.2.1. Speedometer

A speedometer is a gauge that measures and displays the instantaneous speed of a land vehicle. Now universally fitted to motor vehicles, they started to be available as options in the 1900s, and as standard equipment from about 1910 onwards.

Most speedometers have tolerances of some ±10%, mainly due to variations in tire diameter. Sources of error due to tire diameter variations are wear, temperature, pressure, vehicle load, and nominal tire size. Vehicle manufacturers usually calibrate speedometers to read high by an amount equal to the average error, to ensure that their speedometers never indicate a lower speed than the actual speed of the vehicle, to ensure they are not liable for drivers violating speed limits.

Excessive speedometer error after manufacture can come from several causes but most commonly is due to nonstandard tire diameter, in which case the error is

Percent error = 100 *x* (1 - "new diameter"/"standard diameter").

Nearly all tires now have their size shown as "T/A_W" on the side of the tire, and the tire's Diameter in inches = T x A/1270 + W.

For example, a standard tire is "185/70R14" with diameter = 185x70/1270 + 14 = 24.20 in. Another is "195/50R15" with 195x50/1270 + 15 = 22.68 in. Replacing the first tire (and wheels) with the second (on 15" wheels), a speedometer reads 100 * (1 - 22.68/24.20) = 6.28% higher than the actual speed. At an actual speed of 60 mph, the speedometer will indicate 60 * 1.0628 = 63.77 mph, approximately. In the case of wear, a new "185/70R14" tyre of 24.4 inch diameter will have ~8mm tread depth, at legal limit this reduces to 1.6mm, the difference being 12.8mm in diameter or 0.5 inches which is 2% in 24.4 inches.

2.2.2 Vehicle speed sensor (VSS) and Wheel speed sensor (WSS)

VSS sends a varying frequency signal to the TCU to determine the current speed of the vehicle. The TCU uses this information to determine when a gear change should take place based in the various operating parameters. Modern automatic transmissions also have a wheel speed sensor input to determine the true speed of the vehicle to determine whether the vehicle is going downhill or uphill and also adapt gear changes according to road speeds, and also whether to decouple the torque converter at a standstill to improve fuel consumption and reduce load on running gear.

2.3. Study of proposed system

Video sequences of road and traffic scenes are currently used for various purposes, such as studies of the traffic character of freeways. Speed of the vehicles is an important parameter in traffic analysis.

Several approaches have been used to determine velocities from image sequences. This includes development of applications using image processing and pattern recognition algorithm, which is computationally expensive. Another method for motion estimation is background subtraction. However, this method is not widely used for velocity calculation.

The estimation of optical flow plays a key-role in several computer vision problems, including motion detection and segmentation, frame interpolation, three-dimensional scene reconstruction, robot navigation, video shot detection, mosaic creation and video compression. Works in the field of an optical flow calculation has been conducted for more than 30 years. There are many articles that have been written on the subject of optical flow methods. Methods of an optical flow appear to be useful for segmentation of images, and also for detection of obstacles from moving objects.

A sparse optical flow method was developed achieve maximum efficiency and to decrease computing expenses. The optical flow is not applied throughout the picture, but only at feature points. Methods to calculate velocity of a vehicle using moving camera have also been implemented and finds application in robotics.

The proposed method can be used for speed estimation of moving robots or even moving traffic. The method is simple and computationally inexpensive. The methods used in the existing literature for velocity estimation are quite complex and even require different terrine for velocity estimation which is impossible for a moving traffic.

3. SYSTEM ANALYSIS

3.1 Requirement Specification

To be used efficiently, all computer software needs certain hardware components or other software resources to be present on a computer. These pre-requisites are known as (computer) system requirements and are often used as a guideline as opposed to an absolute rule. Most software defines two sets of system requirements: minimum and recommended. With increasing demand for higher processing power and resources in newer version of software, system requirements tend to increase over time. Industry analysts suggest that this trend plays a bigger part in driving upgrades to existing computer systems than technological advancements.

Recommended system requirements:

Recommended system requirements are often suggested by software vendors for optimal performance of the software. Although not a necessity, this set of requirements is often sought after by power users who expect to gain a better experience of software usability. Recommended system requirements do not promise best possible performance of the software and are treated as more of a guideline than a rule. Almost always a better system is available, or will be in future, to provide better performance. Also, exceeding by far these requirements does not guarantee to the user that everything will run with absolute smoothness and look its best. More often than not games are a bit disappointing in this respect, presenting issues that may or may not be corrected with future modifications.

3.2. Software Specification

Operating system : WINDOWS

Language : MATLAB

Library : MATLAB library

Windows

An Operating system is a program that acts as an interface between a user and hard disk. The OS manages computer hardware and system resources, such as memory and hard disk. It also manages the interaction between users and the computer by accepting input from the user, interpreting the input and generating output. The Operating System forms the base on which any application software is developed.

Microsoft Windows is a series of graphical interface operating systems developed, marketed, and sold by Microsoft. Microsoft introduced an operating environment named Windows on November 20, 1985 as a graphical operating system shell for MS-DOS in response to the growing interest in graphical user interfaces (GUI). Microsoft Windows came to dominate the world's personal computer market with over 90% market share, overtaking Mac OS, which had been introduced in 1984. In addition to full Windows-packages, there were runtime-only versions that shipped with early Windows software from third parties and made it possible to run their Windows software on MS-DOS and without the full Windows feature set

Features and utilities in Windows:

- Multiprogramming (Several programs can be executed simultaneously).
- Time-Sharing (CPU time is shared among the programs).
- Multitasking (Handle the execution of multiple task simultaneously. So waiting time reduced).
- Virtual Memory (Logical portion of the hard disk can be utilized as artificial or virtual memory).
- File Transfer and Sharing (Derived from the Server Message Block Protocol and is used by the Microsoft operating system to share files and printers)

MATLAB

MATLAB (matrix laboratory) is a numerical computing environment and fourth-generation programming language. Developed by Math Works, MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, Java, and Fortran.

Although MATLAB is intended primarily for numerical computing, an optional toolbox uses the MuPAD symbolic engine, allowing access to symbolic computing capabilities. An additional package, Simulink, adds graphical multi-domain simulation and Model-Based Design for dynamic and embedded systems.

In 2004, MATLAB had around one million users across industry and academia.[2] MATLAB users come from various backgrounds of engineering, science, and economics. MATLAB is widely used in academic and research institutions as well as industrial enterprises.

MATLAB was first adopted by researchers and practitioners in control engineering, Little's specialty, but quickly spread to many other domains. It is now also used in education, in particular the teaching of linear algebra and numerical analysis popular amongst scientists involved in image processing.

3.4. Hardware specification

- Processor : Intel(R) Core(TM)2 Duo
- Primary Memory : 3GB RAM
- Hard Disk Capacity : 320GB 7200rpm
- Monitor : 14'' HD WLED
- Digital Camera : Canon Powershot 8MP

3.4 LIBRARIES USED

MATLAB is a high-level language and interactive environment for numerical computation, visualization, and programming. Using MATLAB, you can analyze data, develop algorithms, and create models and applications. The language, tools, and built-in math functions enable you to explore multiple approaches and reach a solution faster than with spreadsheets or traditional programming languages, such as C/C++ or Java™. You can use MATLAB for a range of applications, including signal processing and communications, image and video processing, control systems, test and measurement, computational finance, and computational biology. More than a million engineers and scientists in industry and academia use MATLAB, the language of technical computing. MATLAB is a programming language developed by MathWorks. It started out as a matrix programming language where linear algebra programming was simple. It can be run both under interactive sessions and as a batch job. Most MATLAB scripts and functions can be run in the open source programme octave. This is freely available for most computing platforms. GNU Octave and LabVIEW MathScript are systems for numerical computations with an m-file script language that is mostly compatible with MATLAB. Both alternatives can replace MATLAB in many circumstances. While a good deal of the content of this book will also apply to both Octave and LabVIEW MathScript, it is not guaranteed to work in exactly the same manner. Differences and comparison between MATLAB and Octave are presented in Comparing Octave and MATLAB.

Basic MATLAB Concepts

1. *The Current Directory and Defined Path*: It is necessary to declare a current directory before saving a file, loading a file, or running an M-file.

2. *Saving Files*: There are many ways to save to files in MATLAB.

save - saves data to files, *.mat by default

uisave - includes user interface

hgsave - saves figures to files, *.fig by default

diary [filename] - saves all the text input in the command window to a text file.

All of them use the syntax:

save filename.ext

3. *Loading Files:* File can directly loaded from file menu or by using the command

>> load filename.ext

4. *File Naming Constraints:* You can name files whatever you want (usually simpler is better though), with a few exceptions.

4. SYSTEM DESIGN

4.1. Introduction

Design is the first step in the development phase for an engineered product or system. It may be defined as the process of applying various techniques and principles for the purpose of defining a devise, a process or a system in sufficient detail to permit its physical realization. “Computer software design lik e engineering design approaches in other disciplines changes continually as new methods, better analysis, and broader understanding evolve.

Using one of a number of design methods the design step produces a data design, an architectural design and a procedural design. Preliminary design is concerned with the transformation requirements to data and software architectures. Detail design Focus on refinements to architectural representation for software. The data design transforms the information domain model created during analysis in to the data structures that will be required to implement the software. The architectural design defines the relationship among major structural components in to a procedural description of the software.

The most creative and challenging phase of the system life cycle is system design. The term design describes a final system and the process by which it is developed. It refers to the technical specification that will be applied in implementing the candidate system. The question involved here is “How the problem is solved”.

System design is a transition from the user-oriented document to the document – oriented program or database personnel. It emphasizes translating performance specification into the design specification and it involves conceiving and planning and then carrying out the plan for generating the necessary reports and outputs. Design phase act as a bridge between the software requirement specification and implementation phase, which specifies the requirements.

i. Input design:

In input design user oriented inputs are converted into computer formats. Input forms are

always an important part of the design functions. Proposed system accepts inputs with complete validations to ensure erroneous data from being entered into the system. Appropriate messages and warnings are displayed to the user as information during any communication with the system.

a) Input screen

The input screen is designed in such a way that it is easier for the user to enter the data in a logical order. The output screen should use the same format, must not be overcrowded and must use consistent terminology.

b) Input validation

The correctness of the entire data is checked by various methods to identify whether the data is acceptable. The entry of duplicate data should be prevented.

ii. Output Design

Computer output is the most important direct source of information to the user. Efficient intelligible output design should improve systems relationship with the user and help in decision making. There is a chance that some of the end users will not actually operate the input data or information through workstations, but will use the output from the system. Two phases of the output design are: output definition and output specification. Output definition take into account the type of output contents, its frequency and its volume. Once the output media is chosen, the detail specifications of output documents are carried out.

The nature of output required from the proposed system is determined during logical design stage itself. The physical design stage takes the outline of the output from the logical design and produces the output as specified during the logical design phase.

iii. Program Design

On the analysis based on the requirement the program is designed taking into consideration all program aspects. The structure requiring, the control flow etc are decided for efficient functioning of the system that was to be developed.

4.2. System architecture

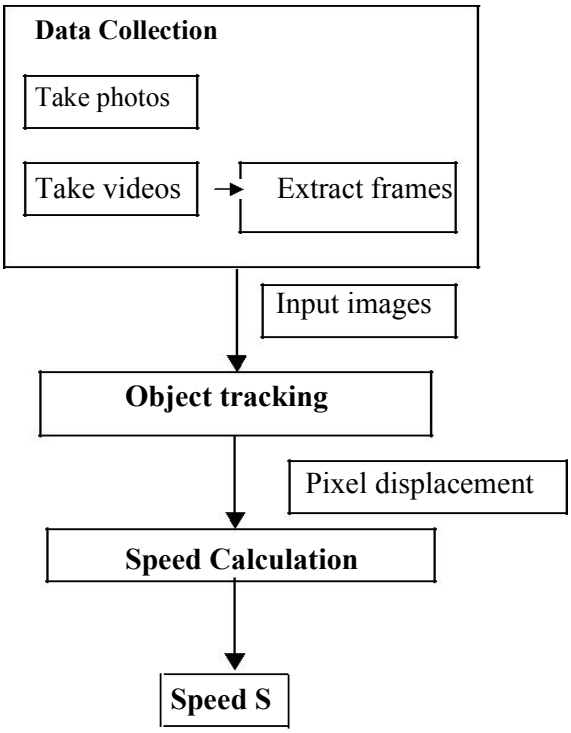


Figure 3.1 System Architecture

System architecture is an important phase in the accomplishment of the project. System architecture is a step by step procedure.

From the figure 3.1, it can be observed that system architecture is implemented in steps. Here, the first step is the data collection step. Data collection includes the collection of images and videos. The collection of images and videos forms a database.

The second step is implemented using the built-in functions provided by the OpenCV library.

The second step consists of object tracking. This is done by inbuilt functions for finding optical flow provided by the OpenCV. The output of this step is the position of the new pixel in the second image.

The third step involves calculating speed. This is the most crucial phase. A MATLAB

program is written to display the input images. The pixel can be marked on the first image and the new position will be automatically marked in the next image. Finally the speed is calculated.

4.3. Flowcharts

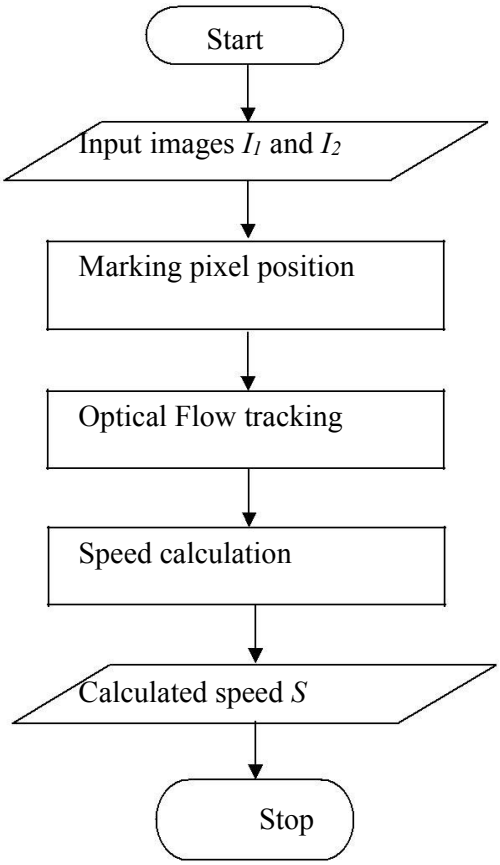


Figure 3.3 Flowchart1

The figure 3.3 represents the design steps. The steps involved in the procedure are as follows:

i) Input two images I_1 and I_2. The images can be still photos or those extracted from a video. Care should be taken to input images that are taken successively within a small time gap. This would ensure that both the images will contain some common feature or object to track.

ii) Now mark a pixel on the image I_1 using the mouse on the display window. This pixel will be tracked on the next image. Thus I_1 will serve as the reference image in our case.

iii) Calculation of the speed of the vehicle is needed, For this calculate the pixel displacement and multiply it by a factor and divide this by the time to give the speed of the vehicle.

iv) Finally the speed S is calculated as the output.

4.4. Physical design

Input Design and Output Design

Input design is the process of converting user inputs into computer-based format. The project requires a set of information from the user to prepare a report. In order to prepare a report, when-organized input data are needed.

In the system design phase, the expanded DFD identifies logical data flow, data stores and destination. Input data is collected and organized into groups of similar data. The goal behind designing input data is to make the data entry easy and make it free from logical errors. The input entry here is negative and positive images for the training the system and a live video stream for hand detection.

Objectives:

- To produce a cost-effective method of input.
- To achieve the highest possible level of accuracy.
- To increase that the input is acceptable and understandable.

Outputs are the most important and direct source of information to the user and to the management. Efficient and eligible output design should improve the system's relationship with the user and help in decision making. Output design generally deals with the results generated by the system i.e. window showing detected hands. The result window shows the detected hands by marking the detected hands using a rectangle which is makes it easy

for the user to identify the result. The input should be a live video stream taken from a web camera.

4.5. Code Design

A code is an ordered collection symbols to provide unique identification of data. Codes can be used by people who do not with data processing; the following are characters of a good code generation

Characteristics of Good Coding

Uniqueness

Meaningfulness

Stability

Uniform size and Format

Simplicity

Conciseness

Versatility

4.5.1Design of Modules

The project is divided into following modules:

1) Data Collection – Data collection includes the collection of test images. The images could be either still photographs taken with specified time gap or could be frames extracted from a video. For obtain these test images or video we will be using a high quality digital camera. The one we have used is Canon Powershot. The vehicle can be of any kind, with or without in-built mechanisms for finding out the speed. We have used a random car on the road for this purpose. And thus the necessary photos and videos were taken at different speeds. The speeds at which each photo and video were taken were also noted. For extracting frames from a video, there is a separate program in OpenCV, which examines each frame of the video and saves it automatically numbering each image in the sequence 0,1,2,3 and so on. We can then manually examine the images and select appropriate images as input to the main program.

2) Lucas-Kanade object tracking – In this module, the major part of the project comes. The two images from the first module are passed as input here. As the first step of this process, we will mark a common object that is present on both the images. Since the photos are taken within a small time interval there is a high probability for the object to appear in the next image too.Using Lucas-Kanade tracking algorithm, we can easily detect the position of the marked pixel in the second image too. OpenCV implementation of Lucas-Kanade Pyramidal tracking provides subpixel accuracy.

3) Speed calculation – In this module we calculated the actual speed of the vehicle using the available information. The first step is to find the pixel displacement of the object in the images. This is given by $d = ((x_2-x_1)^2 + (y_2-y_1)^2)$. Where (x_1,y_1) and (x_2,y_2) are this pixel values in the first and second images respectively. We had taken many test cases and assumed a factor which estimates the speed of the vehicle. This factor multiplied with the pixel distance and then divided by the time, gives the actual speed of the vehicle during that time interval. The calculated speed S is the final output.

5. IMPLEMENTATION

5.1 Introduction

System implementation is a final phase i.e., putting the utility into action. The implementation is a state in the project where theoretical design turned into working system. The most crucial state is achieving a new successful system and giving confidence in new system that it will work efficiently and effectively. The system is implemented only after thorough checking is done. It involves careful planning, investigation of current system and constraints on implementation design of method to achieve.

The implementation is a stage of the project where the theoretical design is converted into a working system. The implementation state is a system project in its own right. It involves careful planning, investigation of the current system and constraints on implementation design of method to achieve the change over and the evolution method. Once the planning has been completed, the major effort is to ensure that the programs in the system are working properly. Implementation phase is as important one in which the source code put into the operation. Before implementing the software careful documentation is necessary. Implementation should provide with well-defined software requirements, design specifications. The major milestone of project implementation is successful integration of software components in the functioning system. During the implementation the configuration management and quality assurance of requirement, design specification and source code are performed.

It involves careful planning, investigation of the current system and constraints on implementation, design of methods to achieve. Two checking is done and if it is found working according to the specification, major task of preparing the implementation are educating, training the users.

5.2. Implementation plan

The major implementation procedures are,

- Test Plans
- Training
- Equipment Installation
- Conversion

5.2.2. Test Plans

The implementation of a computer-based system requires that test data be prepared and that the system and its elements be tested in a planned, structured manner. The computer program component is a major subsystem of the computer-based information system, and particular attention should be given to the testing of these system elements as it is developed.

5.2.2. Training

The purpose of the training is to ensures that all the personnel who are to be associated with the computer-based business system possess the necessary knowledge skills. Operating, programming and user personnel are trained using reference manuals as training aids.

Programmer Training

Programmers are assigned to the computer-based business system project at the beginning of the development phase. The programmer's reference manual informs an experienced programmer, unfamiliar with the system, about all of the aspects of the program.

The manual should enables this person to

- Understand existing program components;
- Modify existing program components; and
- Write new program components.

Operator Training

If new equipment is to be installed, operator training is completed in conjunction with its installation and checkout. If new equipment is not required for the computer-based system, operators still must become familiar with the operational requirements of the new system. Different kinds of personnel may be involved in the operation of the system, such as computer operators, console operators and data entry operators. Training programs for operators are scheduled to coincide with the needs of the computer-based business system as it is developed tested and approaches operational status. Users, analysts, and programmers may participate in the training of operators.

User Training

After the system is implemented successfully, training of the user is one of the most important subtasks of the developer. Even well designed and technically elegant systems can succeed or fail, because of the way they are operated and used. For this purpose user manuals are prepared and handled over to the user to operate the developed system. Both the hardware and software securities are made to run the developed system successfully in future. In order to put new application system into use, the following activities were taken care of:

- Preparation of user and system documentation
- Conducting user training with demo and hands on
- Test run for some period to ensure smooth switching over the system.

5.2.3 Equipment Installation

Equipment vendors can provide the specifications for equipment installation. They usually work with the project's equipment installation team in planning for adequate space, power and light, and a suitable environment. After a suitable site has been completed, the computer equipment can be installed. Although equipment normally installed by the manufacturer, the implementation team should advice and assist. Participation enables the team to aid in the installation and, more importantly, to become familiar with the equipment.

5.2.4. Conversion

Conversion is the process of performing all of the operations that result directly in the turnover of the new system to the user. It has 2 parts.

- The creation of a conversion plan at the start of the development phase and the implementation of this plan throughout the development phase.
- The creation of a system changeover plan to the end of the development phase and the implementation of the plan at the beginning of the operation phase.

5.3. Area of implementation

- FISAT Entertainment Technology center (ETC).

FISAT management plans to start an **Entertainment Technology Center.** It is actually a virtual reality lab with the 3D facility. While entering in the lab and working with the machines the user feels what is happening on the screen.

5.4. Coding

hw1_2.m

```
function r = hw1_2(imageroot, numimages)

resultdirname = 'results';
mkdir(resultdirname);

imstring = strcat(imageroot, '1.jpg');
F = imread(imstring);
graymap;
image(F);

imsize = size(F);
[X,Y] = meshgrid(1:1:imsize(2), 1:1:imsize(1));
X = double(X);
```

```
Y = double(Y);

roicenter = ginput(1)
r = 10;
roitopy = round(roicenter(2)-r);
roiboty = round(roicenter(2)+r);
roileftx = round(roicenter(1)-r);
roirightx = round(roicenter(1)+r);
roi = F(roitopy:roiboty, roileftx:roirightx);

Fbox = drawBox(F, roitopy, roiboty, roileftx, roirightx);
image(Fbox);
imtowrite = strcat(resultdirname, '/1.jpg');
imwrite(Fbox, imtowrite, 'jpg');

sigma = 3;
g = gauss(sigma);
dg = dgauss(sigma);
kernelx = g'*dg;
kernely = kernelx';

fx = conv2(double(F), double(kernelx), 'same');
fy = conv2(double(F), double(kernely), 'same');
fxroi = double(fx(roitopy:roiboty, roileftx:roirightx));
fyroi = double(fy(roitopy:roiboty, roileftx:roirightx));

C11 = sum(sum(fxroi.^2));
C12 = sum(sum(fxroi.*fyroi ));
C21 = C12;
C22 = sum(sum(fyroi.^2));
C = [C11, C12; C21, C22];
```

```
sigma2 = 1.5;
g2 = gauss(sigma2);
dg2 = dgauss(sigma2);
kernelx2 = g2'*dg2;
kernely2 = kernelx2';

fx2 = conv2(double(F), double(kernelx2), 'same');
fy2 = conv2(double(F), double(kernely2), 'same');
fxroi2 = double(fx2(roitopy:roiboty, roileftx:roirightx));
fyroi2 = double(fy2(roitopy:roiboty, roileftx:roirightx));

C11_2 = sum(sum(fxroi2.^2));
C12_2 = sum(sum(fxroi2.*fyroi2 ));
C21_2 = C12_2;
C22_2 = sum(sum(fyroi2.^2));
C_2 = [C11_2, C12_2; C21_2, C22_2];

uv = [0 0];

for i = 2:numimages
   imname = strcat(imageroot, num2str(i))
   imstring = strcat(imname, '.jpg')
   G = imread(imstring);

   warpedG = G;

   groi = warpedG(roitopy:roiboty, roileftx:roirightx);
   ftroi = double(groi)-double(roi);

   Gdub = double(G);

   j = 0;
```

```
   [fty, ftx] = size(ftroi);
   ftcount = fty*ftx;

  while ( (j<10) && ( (sum(sum((ftroi./255).^2))/ftcount) > .0005 ) )
      XI = double(X+uv(1));
      YI = double(Y+uv(2));
      warpedG = interp2(X,Y,Gdub,XI,YI,'cubic');
      groi = warpedG(roitopy:roiboty, roileftx:roirightx);
      ftroi = double(groi)-double(roi);

      if j<2
         D11 = sum(sum(ftroi.*fxroi));
         D12 = sum(sum(ftroi.*fyroi));
         D = [D11; D12];
         invC = inv(C);
         incruv = invC*D;
      else
         D11 = sum(sum(ftroi.*fxroi2));
         D12 = sum(sum(ftroi.*fyroi2));
         D = [D11; D12];
         invC = inv(C_2);
         incruv = invC*D;
      end
      uv = uv - incruv';
      j = j+1;
  end

   Gbox = drawBox(G, roitopy+round(uv(2)), roiboty+round(uv(2)),
roileftx+round(uv(1)), roirightx+round(uv(1)));
   imtowrite = strcat(resultdirname, '/', num2str(i), '.jpg')
   imwrite(Gbox, imtowrite, 'jpg');
end
figure
```

```
graymap;
Gbox = drawBox(G, roitopy+round(uv(2)), roiboty+round(uv(2)), roileftx+round(uv(1)),
roirightx+round(uv(1)));
image(Gbox)

r = (((uv(1)^2)+(uv(2)^2))^0.5)*1.805 ;
disp(['Speed=',num2str(r),'Km/h']);

function boxed = drawBox(M, ytop, ybot, xleft, xright)
boxed = M;
boxed(ytop, xleft:xright) = 255;
boxed(ybot, xleft:xright) = 255;
boxed(ytop:ybot, xleft) = 255;
boxed(ytop:ybot, xright) = 255;
```

dguass.m

```
function dG = dgauss(sig)

x = floor(-3*sig):ceil(3*sig);
G = exp(-0.5*x.^2/sig^2);
G = G/sum(G);
dG = -x.*G/sig^2;
```

guass.m

```
function G = gauss(sig)

x = floor(-3*sig):ceil(3*sig);
G = exp(-0.5*x.^2/sig^2);
G = G/sum(G);
```

graymap.m

```
map = [0:255;0:255;0:255];
map = map';
map = map/255;
colormap(map);
```

interp2.m

```
function zi = interp2(varargin)
error(nargchk(1,6,nargin));

bypass = 0;
uniform = 1;
if isstr(varargin{end}),
  narg = nargin-1;
  method = [varargin{end} '    ']; % Protect against short string.
  if method(1)=='*', % Direct call bypass.
    if method(2)=='l' | all(method(2:4)=='bil'), % bilinear interpolation.
      zi = linear(varargin{1:end-1});
      return

    elseif method(2)=='c' | all(method(2:4)=='bic'), % bicubic interpolation
      zi = cubic(varargin{1:end-1});
      return

    elseif method(2)=='n', % Nearest neighbor interpolation
      zi = nearest(varargin{1:end-1});
      return

    elseif method(2)=='s', % spline interpolation
      method = 'spline'; bypass = 1;

    else
      error([deblank(method),' is an invalid method.']);

    end
  elseif method(1)=='s', % Spline interpolation
    method = 'spline'; bypass = 1;
  end

else
  narg = nargin;
  method = 'linear';
end

if narg==1, % interp2(z), % Expand Z
  [nrows,ncols] = size(varargin{1});
  xi = 1:.5:ncols; yi = (1:.5:nrows)';
  x = 1:ncols; y = (1:nrows);
  [msg,x,y,z,xi,yi] = xyzchk(x,y,varargin{1},xi,yi);

elseif narg==2. % interp2(z,n), Expand Z n times
  [nrows,ncols] = size(varargin{1});
  ntimes = floor(varargin{2}(1));
```

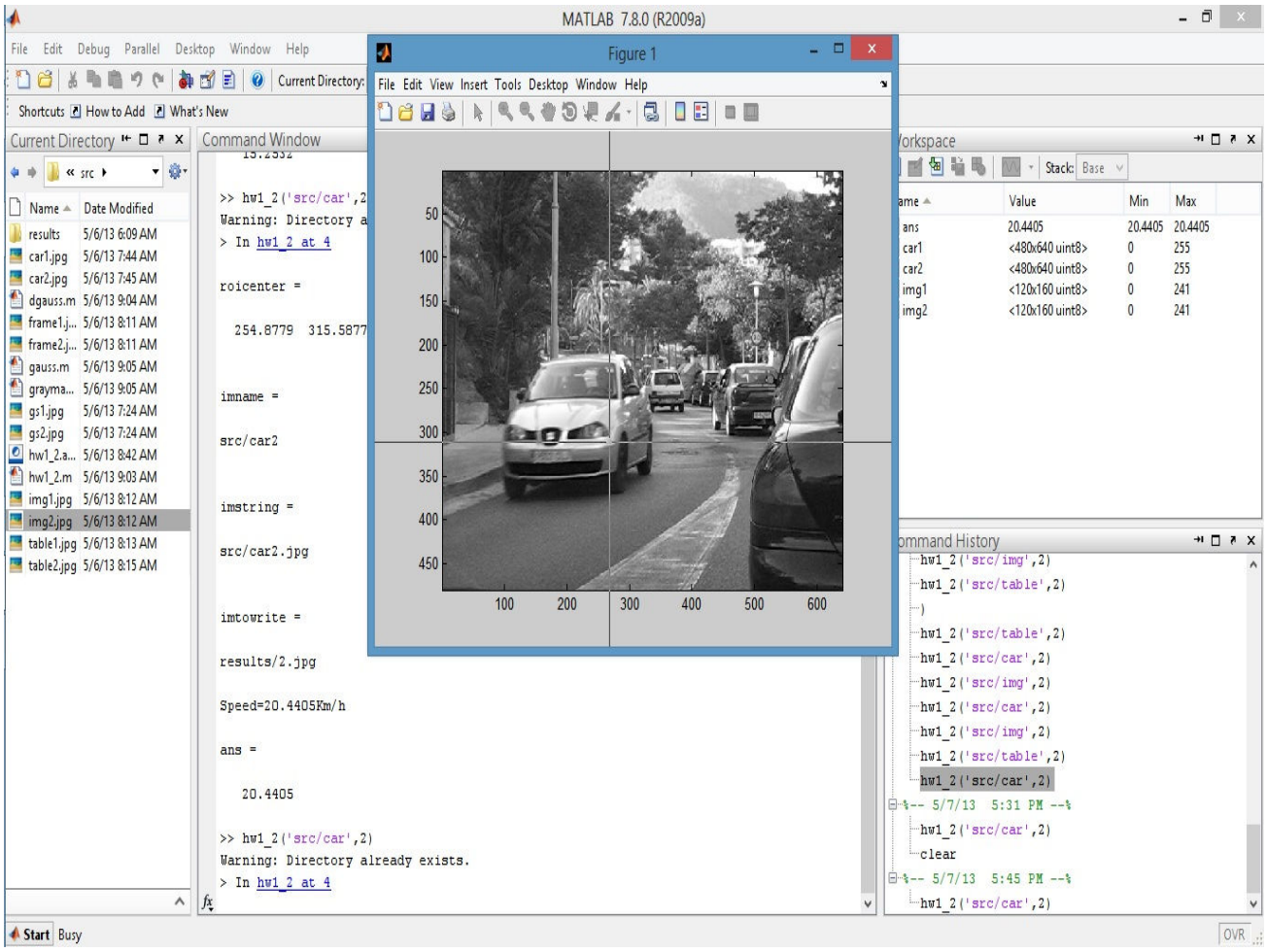
```
  xi = 1:1/(2^ntimes):ncols; yi = (1:1/(2^ntimes):nrows)';
  x = 1:ncols; y = (1:nrows);
  [msg,x,y,z,xi,yi] = xyzchk(x,y,varargin{1},xi,yi);

elseif narg==3, % interp2(z,xi,yi)
  [nrows,ncols] = size(varargin{1});
  x = 1:ncols; y = (1:nrows);
  [msg,x,y,z,xi,yi] = xyzchk(x,y,varargin{1:3});

elseif narg==4,
  error('Wrong number of input arguments.');

elseif narg==5, % linear(x,y,z,xi,yi)
  [msg,x,y,z,xi,yi] = xyzchk(varargin{1:5});

end

if ~isempty(msg), error(msg); end

%
% Check for plaid data.
%
xx = x(1,:); yy = y(:,1);
if (size(x,2)>1 & ~isequal(repmat(xx,size(x,1),1),x)) | ...
   (size(y,1)>1 & ~isequal(repmat(yy,1,size(y,2)),y)),
  error(sprintf(['X and Y must be matrices produced by MESHGRID. Use' ...
    ' GRIDDATA instead \nof INTERP2 for scattered data.']));
end

%
% Check for non-equally spaced data.  If so, map (x,y) and
% (xi,yi) to matrix (row,col) coordinate system.
%
if ~bypass,
  xx = xx.'; % Make sure it's a column.
  dx = diff(xx); dy = diff(yy);
  xdiff = max(abs(diff(dx))); if isempty(xdiff), xdiff = 0; end
  ydiff = max(abs(diff(dy))); if isempty(ydiff), ydiff = 0; end
  if (xdiff > eps*max(abs(xx))) | (ydiff > eps*max(abs(yy))),
    if any(dx < 0), % Flip orientation of data so x is increasing.
      x = fliplr(x); y = fliplr(y); z = fliplr(z);
      xx = flipud(xx); dx = -flipud(dx);
    end
    if any(dy < 0), % Flip orientation of data so y is increasing.
      x = flipud(x); y = flipud(y); z = flipud(z);
      yy = flipud(yy); dy = -flipud(dy);
    end

    if any(dx<=0) | any(dy<=0),
      error('X and Y must be monotonic vectors or matrices produced by MESHGRID.');
```

```
    end

    % Bypass mapping code for cubic
    if method(1)~='c',
      % Determine the nearest location of xi in x
      [xxi,j] = sort(xi(:));
      [dum,i] = sort([xx;xxi]);
      ui(i) = (1:length(i));
      ui = (ui(length(xx)+1:end)-(1:length(xxi)))';
      ui(j) = ui;

      % Map values in xi to index offset (ui) via linear interpolation
      ui(ui<1) = 1;
      ui(ui>length(xx)-1) = length(xx)-1;
      ui = ui + (xi(:)-xx(ui))./(xx(ui+1)-xx(ui));

      % Determine the nearest location of yi in y
      [yyi,j] = sort(yi(:));
      [dum,i] = sort([yy;yyi(:)]);
      vi(i) = (1:length(i));
      vi = (vi(length(yy)+1:end)-(1:length(yyi)))';
      vi(j) = vi;

      % Map values in yi to index offset (vi) via linear interpolation
      vi(vi<1) = 1;
      vi(vi>length(yy)-1) = length(yy)-1;
      vi = vi + (yi(:)-yy(vi))./(yy(vi+1)-yy(vi));

      [x,y] = meshgrid(1:size(x,2),1:size(y,1));
      xi(:) = ui; yi(:) = vi;
    else
      uniform = 0;
    end
  end
end

% Now do the interpolation based on method.
method = [lower(method),'   ']; % Protect against short string

if method(1)=='l' | all(method(1:3)=='bil'), % bilinear interpolation.
  zi = linear(x,y,z,xi,yi);

elseif method(1)=='c' | all(method(1:3)=='bic'), % bicubic interpolation
  if uniform
    zi = cubic(x,y,z,xi,yi);
  else
    d = find(xi < min(x(:)) | xi > max(x(:)) | ...
             yi < min(y(:)) | yi > max(y(:)));
    zi = spline2(x,y,z,xi,yi);
    zi(d) = NaN;
```

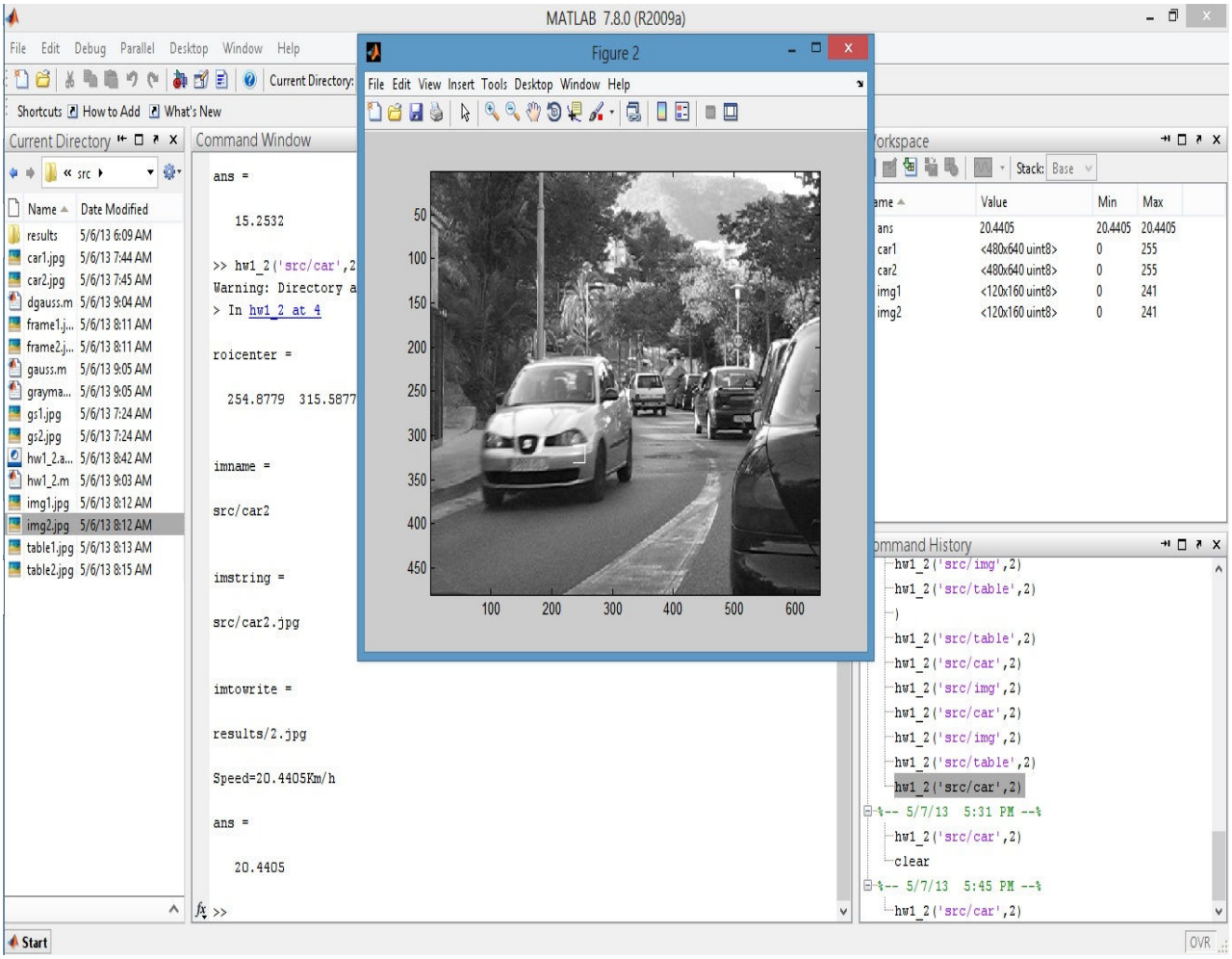
```
  end

elseif method(1)=='n', % Nearest neighbor interpolation
  zi = nearest(x,y,z,xi,yi);

elseif method(1)=='s', % Spline interpolation
  zi = spline2(x,y,z,xi,yi);

else
  error([deblank(method),' is an invalid method.']);

end
```

6. RESULTS AND DISCUSSION:

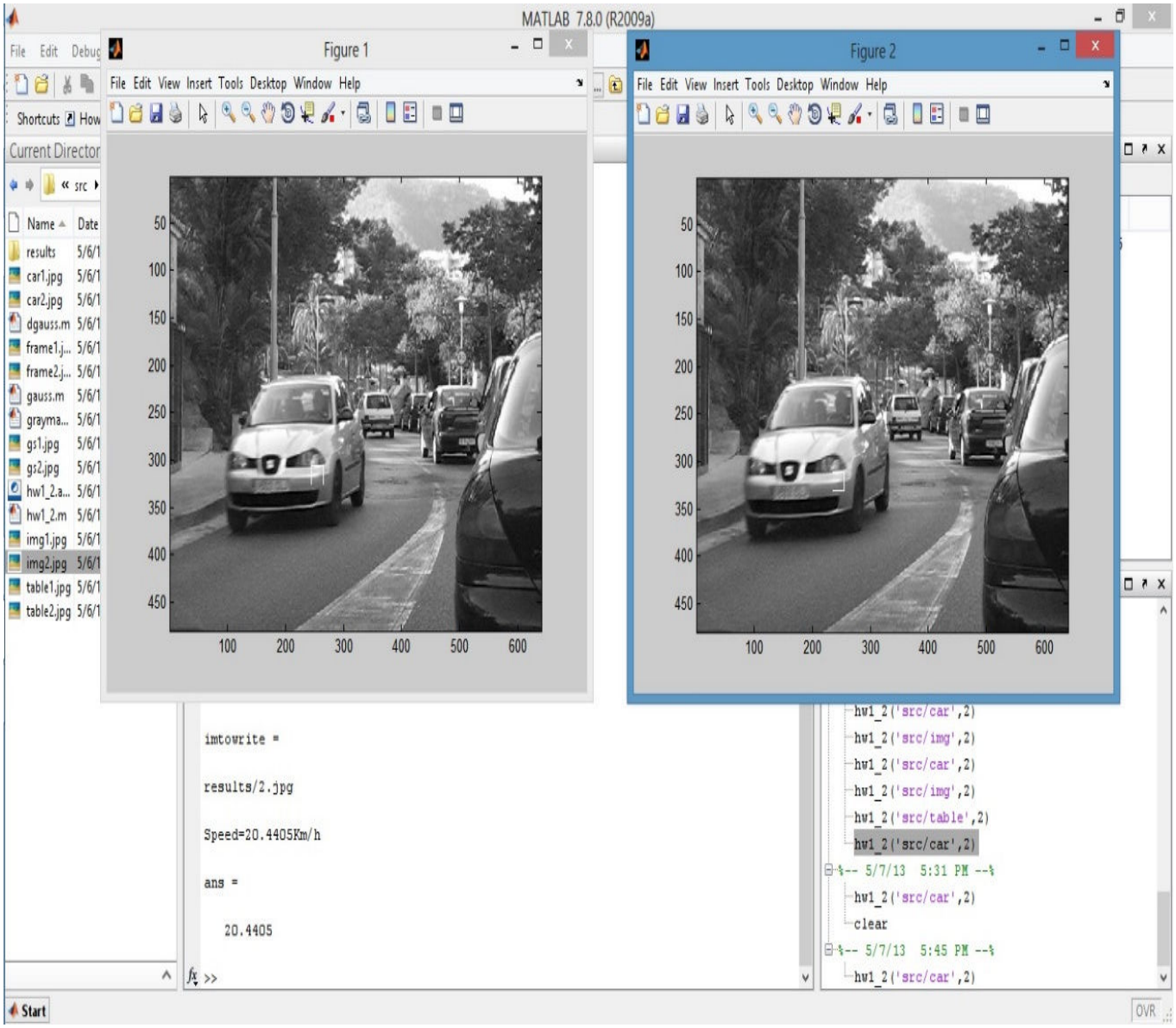
6.1. Screenshots:



6.1.1 Object to be tracked is selected in the image i.e Figure 1

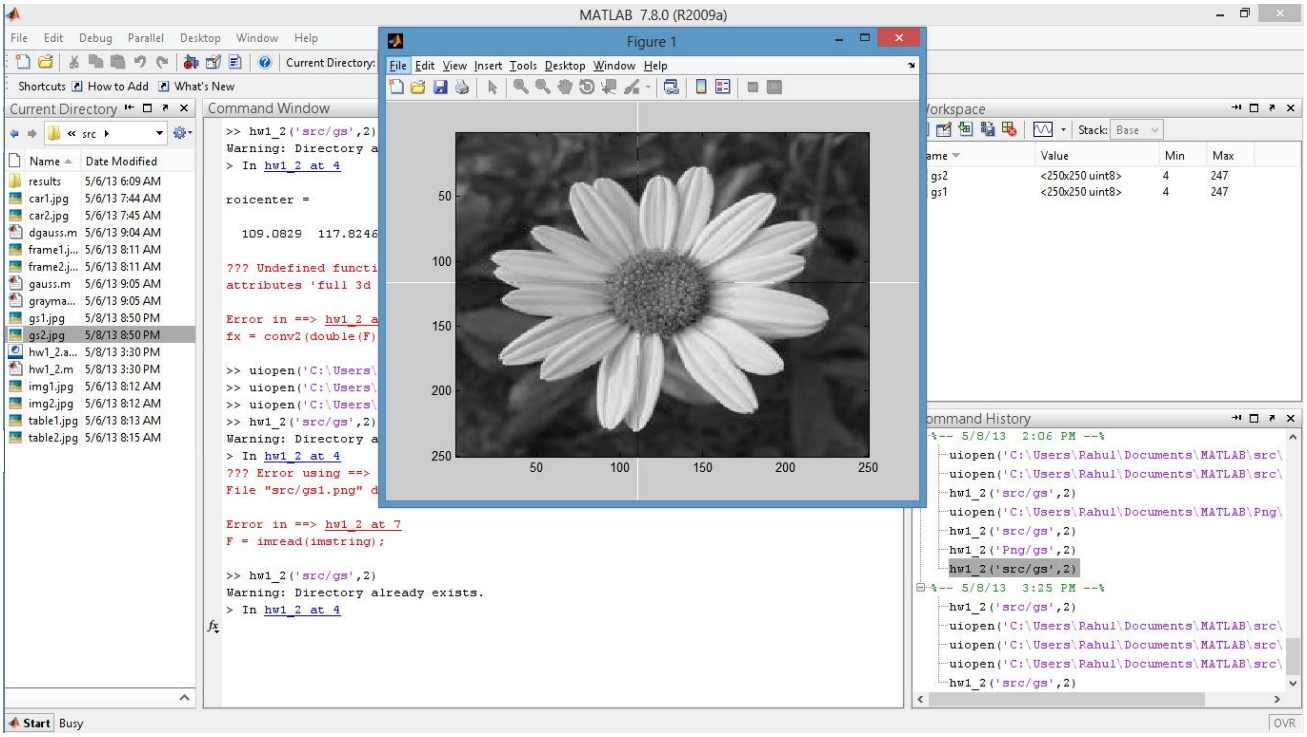


6.1.2 Object is tracked in the second image Figure 2

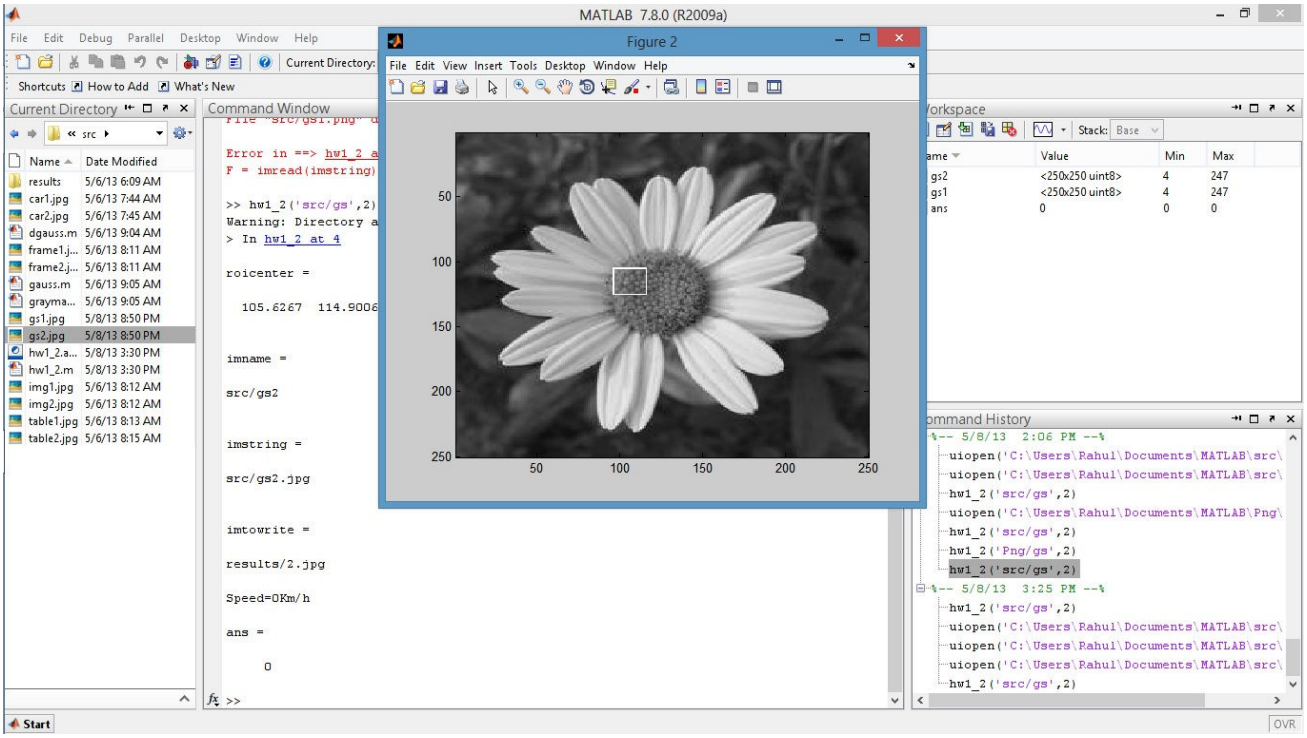


6.1.3 Speed is calculated and displayed

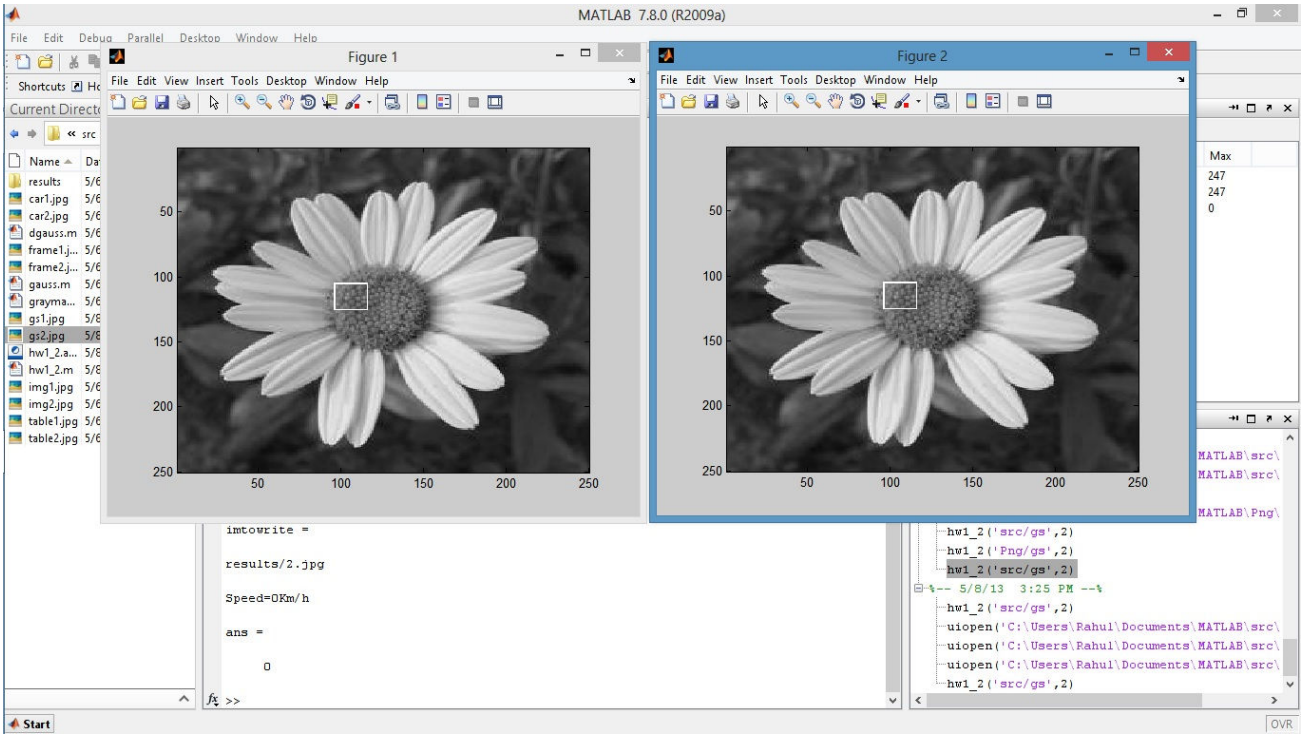
Speed of the car calculated: 20.4405 Km/h



6.1.4 Object to be tracked is selected in the image i.e Figure 1.



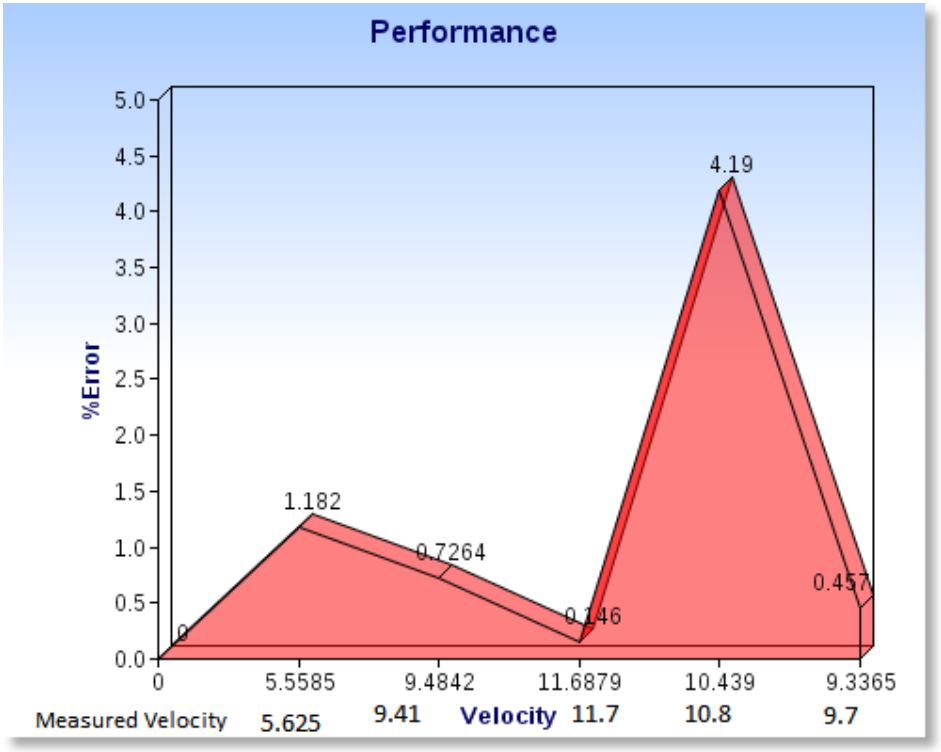
6.1.5 Object is tracked in the second image i.e Figure 2



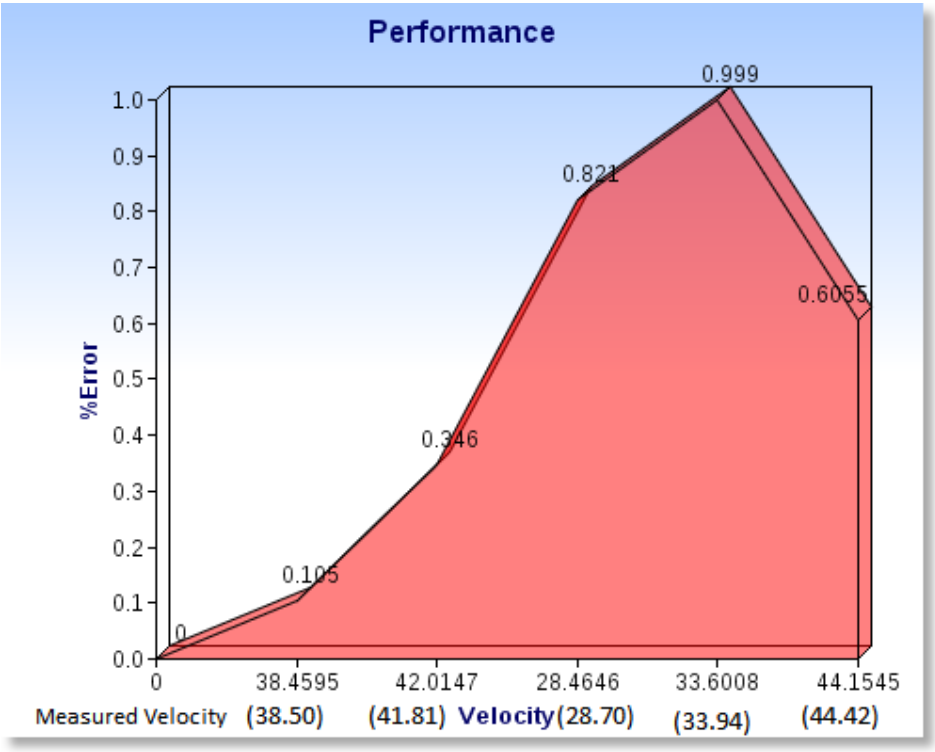
6.1.6 The speed of the object is calculated and displayed.

Speed of the flower is: 0.000 Km/h

6.2. Performance Graphs



6.2.1 Performance analysis of Vehicle 1



6.2.2. Performance Analysis of vehicle 2

6.2.3 Table: Comparison of Measured and Calculated Velocity (Two Vehicles)

Measured Velocity (cm/s)		**Computed Velocity (cm/s)**		**Error | v-v′ | (cm/s)**		**% Error**	
V1	V2	V1’	V2’				
5.625	38.50	5.5585	38.4595	0.0665	0.0405	1.182	0.105
9.4158	41.81	9.4842	42.0147	0.0683	0.2047	0.7264	0.346
11.705	28.70	11.6879	28.4646	0.017	0.2358	0.146	0.821
10.896	33.94	10.439	33.6008	0.4572	0.339	4.19	0.999
9.7839	44.424	9.3365	44.1545	0.0447	0.269	0.457	0.6055

7. TESTING

7.1. Introduction

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and code generation. Testing technique provides systematic guidance for designing it mainly helps in exercise the internal logic of software components Exercise the input and output domains of the program to uncover errors in program function, behavior and performance.

7.1.1. Unit Test

This is the starting point of testing. In this a module is tested separately and is often performed by the coder himself simultaneously along with the coding of the module. The purpose is to exercise the different parts of the module code to detect coding errors. The project has been into different modules namely login, search, payment and shipping, admin module. Thus these modules are tested separately.

7.1.2. Integrated Test

In this many unit-tested modules are combined into subsystems which are then tested. The goal here is to see if the modules can be integrated properly. Hence the emphasis is on testing interfaces between modules. Thus all the above mentioned modules are integrated together and tested.

7.1.3. System Testing

After the system is put together system testing is performed. Here the system is tested against requirements to see if the requirements are met and if the system performs a specified by the requirements. Here, the connectivity with the database should be tested in order to store and retrieve information.

7.1.4. Acceptance testing

Finally acceptance testing is performed to demonstrate to the client on the real life data of the client the operation of the system. It is sometimes performed with realistic data of the client to demonstrate that the software is working satisfactory. For this, a multiple client server connection is made and checked if multiple users could send commands to control the virtual machine. Thus one could make sure that the project was made according to the requirements specified

7.2. Test Plans

The proposed project has undergone the formal process of implementation in the same manner as every other system would undergo. The test plan was designed to test the quality of the system.

7.2.1. Unit Testing

Various test cases are prepared. For each module these test cases are implemented and it is checked whether the module is executed as per the requirements and desired output is obtained.

7.2.2. Integration Testing

The modules are tested separately for accuracy and modules are integrated together using bottom up integration i.e. by integrating from moving from bottom to top. The system is checked and errors found during integration are rectified.

7.2.3. Validation Testing

Entering incorrect values does the validation testing and it is checked whether the errors are being considered. Incorrect values are to be discarded. The errors are rectified.

7.2.4. System Testing

System testing was performed to verify that all system elements have been properly integrated and perform allocated function. Performance testing was done to test the runtime performance of the software. For user acceptance testing, the system was given to the end user to use. The errors found are rectified.

7.3. Test Cases & Test Results

7.3.1 Test Plan for vehicle tracking:

Test Case ID	Input	Description	Expected Result
01	Image name blank	Image name should be given	Error
02	No of images blank	No of images should be given	Error
03	One image	Minimum two images Should be given	Error
04	3D images	Images should be 2D	Error
05	No point selected for integration Window.	A point is selected for integration Window.	Error
06	Valid Image name and no of images	Valid Image name and no of images	Vehicle tracking successful

7.3.2 Test Report for Vehicle tracking:

Test Case ID	Input	Description	Expected Result	Pass / Fail
01	Image name blank	Image name should be given	Error	Pass
02	No of images blank	No of images should be given	Error	pass
03	One image	Minimum two images Should be given	Error	Pass
04	3D images	Images should be 2D	Error	Pass
05	No point selected for integration Window.	A point is selected for integration Window.	Error	Pass
06	Valid Image name and no of images	Valid Image name and no of images	Vehicle tracking sucessfull	Pass

8. CONCLUSION AND FUTURE ENHANCEMENTS

The project was successfully completed and executed. The speed of a moving vehicle was found with good accuracy.

8.1 Future Scope

The project can be expanded to various applications that are given below:

The project can be implemented in a variety of scenarios, as it gives the exact displacement of a moving vehicle at a given amount of time. This enables us to find the accurate speed. The project can be refined and can replace the speedometers that are currently in use. The technology used today is used to find only the approximate speed, while the speed tracker calculates the exact speed. A real time speedometer can be implemented by video using a high resolution camera and by finding the displacement in each of its frame.

The project can also be implemented on a stationary object and speed of vehicles moving by the object can be estimated. Speed tracker uses Lucas-Kanade algorithm which uses effective object tracking. Objects of any size and shape can be tracked.

The project can auto track objects with good features that can automatically track good features that are most likely to be found in the consecutive frames. There is also an option to track features in low light.

A system can also be developed with which can the vehicles route by using first identifying the vehicle using an identification number recording the location, time, speed of the vehicle into the database, which can later be retrieved accordingly.

BIBILIOGRAPHY

[1] Jae Kyu Suhr. Kanade-Lucas-Tomasi (KLT) Feature Tracker. In Computer Vision (EEE6503)., pages 9-18. Fall 2009, Yonsei Univ

[2] R. Cutler and M. Turk. View-based Interpretation of Real-time Optical Flow for Gesture Recognition. In Proc. IEEE Intl. Conference on Automatic Face and Gesture Recognition, pages 416-421, April 1998.

[3] P. Viola and M. Jones. Robust Real-time Object Detection. Int. Journal of Computer Vision, 2002.

[4] Lazaros Grammatikopoulos, George Karras, Elli Petsa, "Automatic Estimation of Vehicle Speed from Uncalibrated Video Sequences" , International Symposium on Modern Technologies, Education and Professional Practice in Geodesy and related fields, Sofia, 03 – 04 November, 2005.

[5] Gibson J.J., "The Perception of the Visual World" , Riverside Press,Cambridge 1950.

[6] J.L. Barron, N.A. Thacker, 'Tutorial: Computing 2D and 3D Optical Flow' , Tina Memo No. 2004-012, 2005.

[7] Chi-Cheng Cheng, and Hui-Ting Li .'Feature-Based Optical Flow Computation'

.International Journal of Information Technology Vol.12 No.7 2006.

[8] Savan Chhaniyara, Pished Bunnun, Lakmal D. Seneviratne and Kaspar Althoefer.

'Optical Flow Algorithm for Velocity Estimation of Ground Vehicles: A Feasibility Study' . International Journal on smart sensing and intelligent systems, VOL. 1, NO. 1, MARCH 2008.

[9] C. Braillon, C. Pradalier, J. Crowley, C. Laugier, 'Real-time moving obstacle

detection using optical flow models' , Proc. of the IEEE Intelligent Vehicle Symp., 2006, pp.466-471.

[10] J.L. Barron, D.J. Fleet, S.S. Beauchemin, T.A. Burkitt,'Performance of Optical Flow Techniques' , Computer Society Conference on Computer Vision and Pattern Recognition, 1992, pp. 236-242.