Anx-B

Intelligent Navigation and Control of Mobile Robots Using Android Platform

Author

Muhammad Zohaib

2011-NUST-MSPhD-Mts-25

Supervisor

Dr. Kunwar Faraz Ahmed

DEPARTMENT OF MECHATRONICS ENGINEERING

COLLEGE OF ELECTRICAL & MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY

ISLAMABAD

MAY, 2014

Intelligent Navigation and Control of Mobile Robots Using Android Platform

Author

Muhammad Zohaib

2011- NUST-MSPhD-Mts-25

A thesis submitted in partial fulfillment of the requirements for the degree of

MSMechatronics Engineering

Thesis Supervisor:

Dr. Kunwar Faraz Ahmed

Thesis Supervisor's Signature:____

DEPARTMENT OF MECHATRONICS ENGINEERING

COLLEGE OF ELECTRICAL & MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY, ISLAMABAD

MAY, 2014

Declaration

I certify that this research work titled "*Intelligent Navigation and Control of Mobile Robots Using Android Platform"* is my own work. The work has not been presented elsewhere for assessment. The material that has been used from other sources it has been properly acknowledged / referred.

Signature of Student

Muhammad Zohaib

2011-NUST-MSPhD-Mts-25

Language Correctness Certificate

This thesis has been read by an English expert and is free of typing, syntax, semantic, grammatical and spelling mistakes. Thesis is also according to the format given by the university.

Signature of Student

Muhammad Zohaib

2011-NUST-MS PhD-Mts-25

Signature of Supervisor

Copyright Statement

Acknowledgements

Innumerable words of praise and thanks to Allah, the Almighty, and the Creator of the universe for carving the path for me and always helping me out in the best possible way. Without His Will and Mercy, I would not have been able to accomplish this milestone. I am grateful to my parents for their immense love, moral support, encouragement and prayers throughout my academic career.

I am deeply beholden to my supervisor, Dr. Kunwar Faraz Ahmed, for his continuous guidance, inspiration, and patience. His ability of management and foresightedness taught me a lot of things which will be more helpful for me in my practical life.

I would also like to thank my fellow students especially Mr. Hamid and Mr. Arsalan, who at the climax time of my thesis, guided mein the right direction which led to the completion of this thesis on time.

I gratefully acknowledge the help and guidance provided by Guidance and Examination Committee members, Dr. Khurrum Kamal, Dr. Umer Shahbaz, and Dr. Arslan Shauqat, that their valuable suggestions and comments were a great source to improve the research work presented in this thesis.

Dedicated to my exceptional parents and adored siblings whose tremendous support and cooperation led me to this wonderfulaccomplishment.

Abstract

In this thesis a method is proposed for the navigation and control of mobile robot using pure vision based algorithm under real time constraints. This project is dedicated to developing control systems for robots using vision based technology only. In recent years, mobile robot systems and its applications have expanded beyond the basic function. Virtually any task can be adapted and implemented using it. The objective of this project is to develop a system to control a robot and navigate a robot autonomously, the advantages of which include wireless access and easy-to-use GUI.

Key Words:*Smartphones,android, computer vision, opencv, Unmanned Ground Vehicle, UGV, Mobile Robots, Robotics, Control Systems, iRobot, Roomba.*

Table of Contents

List of Figures

CHAPTER 1: INTRODUCTION

This chapter introduces the thesis topic. Research for this thesis with a specific introductionfor the motivation and encouragementof the general field, are presented. Furthermore, the objectives of this thesis are defined within the following chapter.

1.1 Background and Scope

Robots are widely used across many fields now. The use of robots has been developed for the requirement of precision making. As general rule suggests that work done by a robot is much more efficient and precise as compared to a human being. Therefore for the making of robots, certain requirements arises, which lead towards the character building of robotic platform. Including the algorithm involved for most of it, while the design also plays its part. Considering the design does not much greatly effect certain conditions, therefore the algorithm must be of high intelligence. The requirement of thus an intelligent algorithm is the main motivation for this thesis.

The control of a robot also plays a vital role, as if there seems no communication between a robot and its server (the body who controls) then the robot is basically useless. So the control of the robot is also taken into much consideration.The control main mechanism in this project lies basically via Bluetooth connection. All robotic control commands are generated and communicated via Bluetooth connection, by sending the basic serial type data over Bluetooth. It is chosen so as the connection to remain wireless, as wired connections might be found useful in certain parts but mostly wireless connections are preferred due to the nature of being wireless and hassle-free behavior.

1.2 Motivation

Considering a practical point of view, optical flow can be used for many types of vision based models, supposedly the main sensor based and motion based models, in the field of autonomous navigation. Optical flow, whereas, is used here for motion based as for autonomous navigation using single camera model. The need and use of single camera greatly improves in the usefulness of its operation, as in case of being single, first of all, being cheaper and smaller and lighter and the optical flow robustness depends on it, as only vision of one camera is being processed through it. As cameras are also becoming much smaller and lighter, it is becoming easier for it to mount on an autonomous vehicle.

The less need of having more type of sensors mounted on the robot also yields a positive aspect, so as less amount of information is communicated via several times, instead only vision based camera info is only transferred.

Optical flow is already known to be capable of determining rotational quantity of a motion [1,2, 3]. And here we are dealing with the translational motion only, as the rotational motion is not required.

1.3 Objectives

The objectives or steps involved for the completion of this dissertation included,

- Familiarization with image Processing Techniques.
- Interfacing Robot and Windows platform wirelessly. (Bluetooth module used).
- Designing the interface to Robot Control.
- Developing the algorithm for Autonomous (collision free) behavior of the robot.

CHAPTER 2: LITERATURE REVIEW

There seems to exist number of ways to control and navigate a robot using stereo vision and mono vision, a review of the existingmethods are briefly discussed here.

2.1 Related Scientific Techniques:

2.1.1 Blob based Obstacle Avoidance

Blob based obstacle avoidance deals with a simple strategy of thresholding an image and by doing edge detection, obstacle detection is tried on this edge detected image as simply the edges contained are the obstacle points and the result would be to avoid them [4]. The three main conditions required for obstacle detection that were used in the paper [4] were that the object must be not hanging, the obstacles must be entirely different then the floor and the ground is relatively flat.

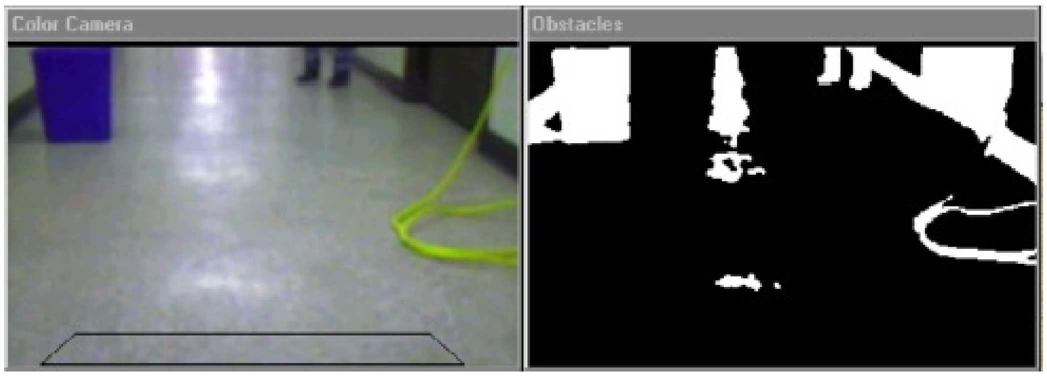


Figure 1. Image from camera (Left), Segmented Image (right), Courtesy [4]

The main aspect to notice here is that, the consideration that the main blob is considered floor and is segmented, what if there are number of obstacles having similar color tone, then instead of avoiding the obstacles the robot will try to hit them.

2.1.2 Single Image Perspective Cues

Single image perspective cues works in such a way that, the robot is considered to move in the direction where the sets of lines align themselves and act as a single perspective. The perfect example used by [5], is the staircase, as the lines being made are by a staircase which will definitely be easily obtained by a simple canny edge detection and then determining the vanishing point, this vanishing point then definitely helps in determining the nature of the environment in which the robot is placed. The far most basic problem with such implementation is that it works in only a few set of environments and is not a wholly solely obstacle avoidance in itself.

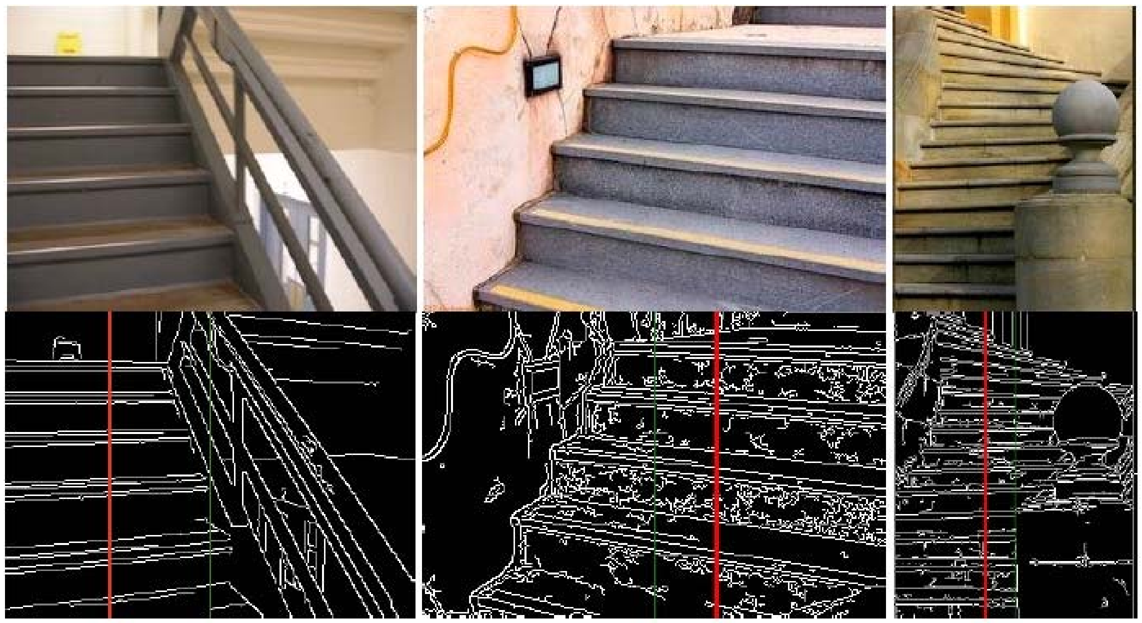


Figure 2. (Top) Original Images of staircases, (Bottom) Images with bold red lining depicting location of staircases, Courtesy [5]

2.1.3 Stereo Vision

Stereo vision, as the name suggests there are two cameras used for the purpose of navigation, which is in fact less robust if we look overall, as computing vision from two cameras and computing using one camera makes the difference less by almost 50%. Whereas it has the ability to achieve vision algorithms used for object recognition without using additional sensors. [6].

In stereo vision, 2 cameras are used to get the same environment using different views. Therefore for getting 3 dimensions from 2 dimensional images, one require more than one 2 dimensional image. In case of more than 2D images, there has to be a relative position of objects to be known and also the relative position of camera be also known, for the finding of the depth inside the images. Also tangential and other disorders and outliers are needed to be removed incase for finding distance to objects. Therefore for such requirements adjustment of angles and distances between the cameras plays a vital role.

The requirement here is still that it requires at least 2 dimensional cameras, as opposed to initial thought to have just one camera used as a sensor. So basically using two cameras means processing all the vision data 2 times as that of one, therefore doubling the amount of processing required.

2.1.4 Monocular Vision

From biological aspect we already know that by just using one camera i.e.in case of bird's eyes being positioned in such a way that binocular depth vision is impossible [7]. There exists number of methods to extract the depth information from single camera, such monocular depth measures include occlusion, texture gradients and optical flow [7]. It is been already discussed that optical based vision depends on two images obtained from one camera with a specific amount of time interval, whereas 2 cameras are talented enough of extracting depth information.

If the environment is static, then by merely using a single camera one can state that two frames are taken at a specific time interval, which will act as they were taken from two separate cameras. The assumption taken here, do limits some criteria's, that any moving object in any of the frame will violate this condition. The basic requirement of using single camera is that one has to find the transformation from one frame to another and for every frame to come. Apart from the other complexity the most important aspect lies with finding one features of a frame in another which has been transformed. Finding these important points/features and using them for vision purposes is known as structure from motion [8]. A common possibility for performing such operation is by using optical flow, therefore by finding the important points and finding those points in the coming consecutive frames. [3,7,9,10,11].

2.2 Image Processing Library:

For the sake of better provisions and improvements and not being required to develop the basic imageprocessing techniques, OpenCV (Open Computer Vision) Library was selected.It was selected as the basic programming could be done in C++ / C language, and due to its improved;

- Speed
- Efficiency
- Optimization
- Cross Platform Support and
- Being Open Source

The Cross Platform support is so much wide that for all popular Operating Systems, there exists a supporting linking library of OpenCV.

The Library is mainly aimed at real time computer vision tasks, its core libraries are written in C++/C, therefore making it further easier for making algorithm run in same language.

2.3 Development Environment:

For the development of the algorithm, there needs to be a suitable platform/environment for the algorithm to work on. As OpenCV core libraries are written under C++/C,therefore for the easy implementation of OpenCV functions, there needed to have a development platform which can directly link to those libraries. Therefore for the case of using an Android platform, Samsung Galaxy Note III was used for the algorithm implementation and development cell. For the sake of programming for this respective cell phone, eclipse development platform was chosen and OpenCV libraries were linked to it.

2.3.1 Eclipse (for Android OS)

Eclipse is an integrated development environment (IDE). It consists of workspace for the use of making an algorithm or for performing respective operations. It is mostly written in java, but some native interfaces can be made for writing programs in C++/C languages and others, as in this case C++ is used, the OpenCV libraries are linked into eclipse and then for writing the program in C++/C language, Java Native Interface (JNI) was used. Java Native Interface was developed for linking the core libraries of OpenCV and the C++/C program, as programming in Java and using the libraries in Java only were not possible, considering the scope of the thesis project.

Using Eclipse number of image processing techniques were implemented, which include

- Image manipulations (grayscale, sepia, etc.)
- Canny edge detection
- Feature detection, etc.

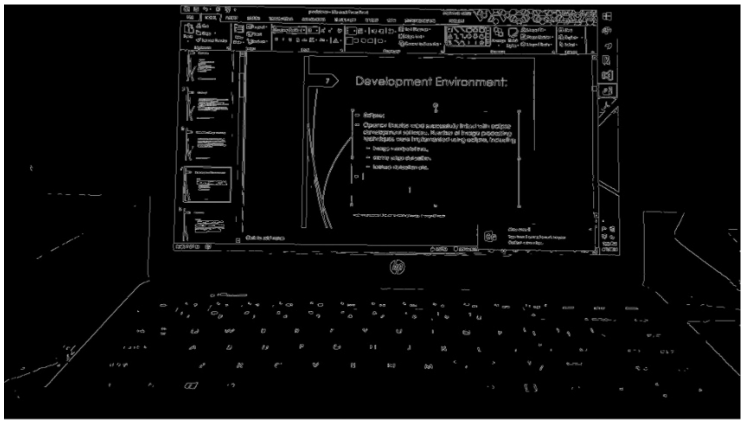


Figure 3. Canny Edge Detection

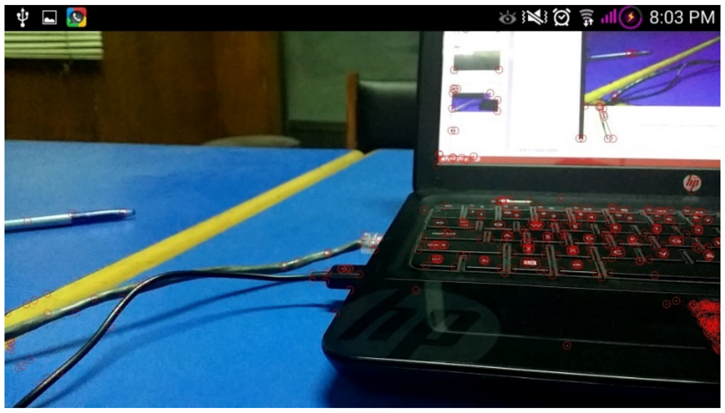


Figure 4. Feature Detection (1)

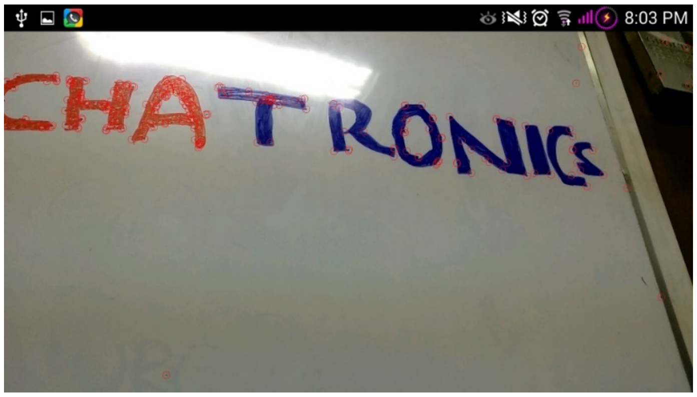


Figure 5. Feature Detection (2)

The Results obtained using eclipse were impuissant and insufficient, and a better approach was needed. As for the utilization of basic Opencv functions, special instructions were required for the java native interface (JNI). For the sake of debugging and understanding the result of every new line of algorithm code, it was way too difficult using Eclipse program, as even if errors are resolved the program "stops responding", and the debugger points towards JNI as a whole. Therefore there was a need for a change in development platform.

2.3.2 Microsoft Visual Studio 2013

Microsoft Visual Studio 2013 is a vast language support platform, for performing multiple tasks. It works on a Microsoft Windows Operating System. It consists of comprehensive amount of tools to perform a wide variety of tasks. The OpenCV libraries at their core language C++/C level were linked and their functions were used successfully. Therefore for the implementation of OpenCV libraries at their core C++/C language level, visual studio was introduced and results were driven out of it.

Therefore the current platform is Windows 8.1 Operating System where libraries of OpenCV 2.4.8 are integrated with Visual Studio 2013.

CHAPTER 3: METHODOLOGY

This chapter clarifies the methodology that has been implemented using the algorithm being made. The main aspect of which is based on optical flow technique. Whereas the methodology can be divided into two categories.

- Algorithm Implementation
- Obstacle Avoidance Strategy

3.1 Flow Charts

The Idea of the implementation of the algorithm can be understood considering the below flow charts.

3.1.1 Main Program

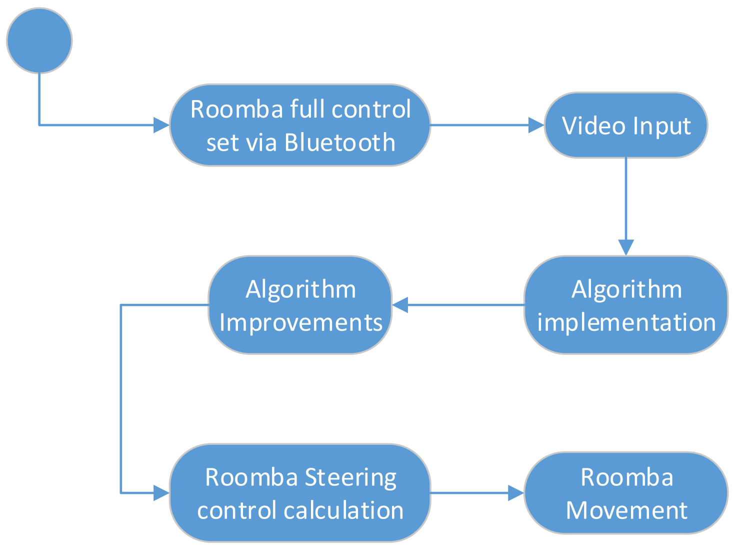

Roomba full control set via Bluetooth
Video Input
Algorithm implementation
Algorithm Improvements
Roomba Steering control calculation
Roomba Movement

Figure 6. Main Program Flow Chart

3.1.2 Algorithm

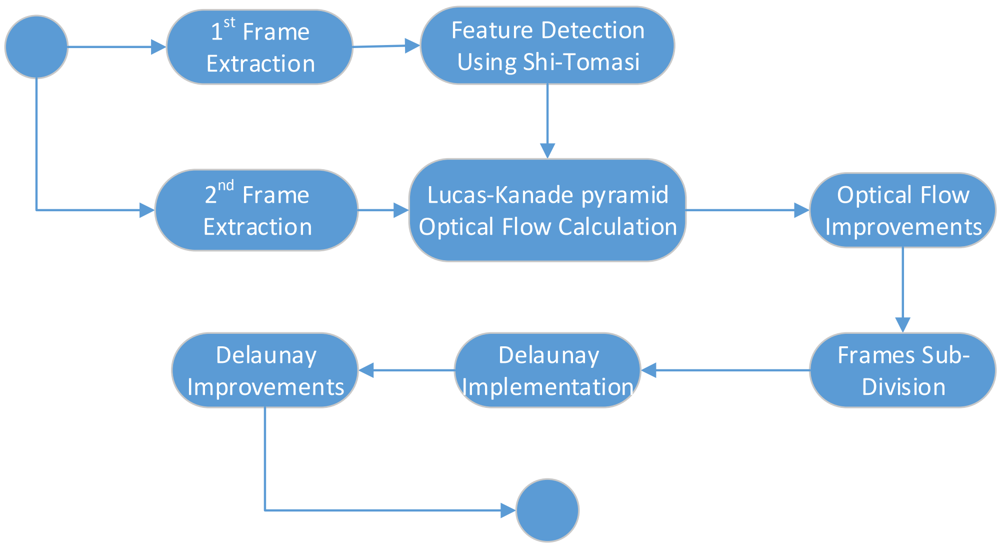

1st Frame Extraction
Feature Detection Using Shi-Tomasi
2nd Frame Extraction
Lucas-Kanade pyramid Optical Flow Calculation
Optical Flow Improvements
Frames Sub-Division
Delaunay Implementation
Delaunay Improvements

Figure 7. Algorithm Flow Chart

3.1.3 Steering Control Conditions

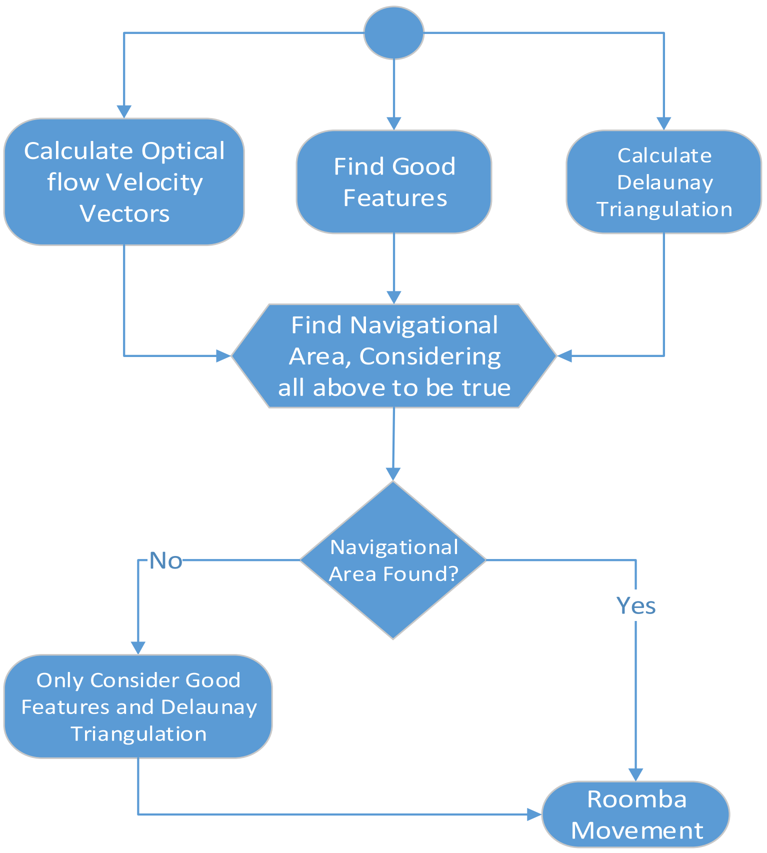

Calculate Optical flow Velocity Vectors
Find Good Features
Calculate Delaunay Triangulation
Find Navigational Area, Considering all above to be true
Navigational Area Found?
No
Yes
Only Consider Good Features and Delaunay Triangulation
Roomba Movement

Figure 8. Steering Control Conditions Flow Chart

3.2 Algorithm Implementation

A moving camera produces a succession of timely ordered images/frames. Each frame whereas consists of projection of 2D pixel array. And for processing these successive frames optical flow is implemented.

3.2.1 Optical Flow

Since optical flow plays a prominent role in biology, it is not surprising that a lot of research [1, 2, 12,13,14,15] has been done considering the computer vision. Optical flow provides many applications in computer vision out of which the main aspect is considered as motion estimation/detection. Optical flow can be also used to detect or track markers to estimate the translational quantity. Another important aspect of it of object segmentation, as when assuming the camera is static and the object is moving inside the environment.

Optical Flow generates flow vectors which purely depends on the features being found between successive frames. Therefore for the velocity vectors to be greater it depicts that the motion is greater and whereas the vectors are smaller depicts that motion is small. In my case the motion of the robot is purely translational therefore the rotational complexity of the optical flow is not taken into account and only the translational components are dealt with.

It is much anticipated that optical flow will work in our case but which type of optical flow has not been discussed. There exists two main types of optical flow.

- Dense optical flow and
- Sparse Optical flow

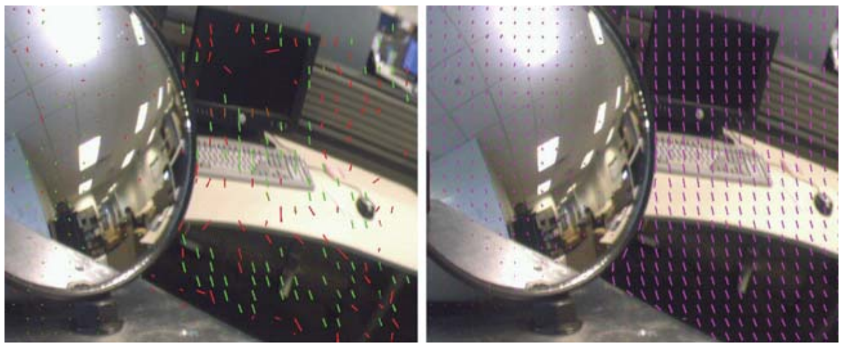


Figure 9. Sparse Optical flow (left) and Dense Optical Flow (right)

Dense optical flow algorithms compute the velocity vectors of optical flow for every pixel of the frame i.e. flow per every pixel whereas sparse optical flow algorithms compute the velocity vectors of optical flow for only good features. These good features are usually the corner/edgy features of the frame and are usually provided to the sparse optical flow using some feature detection technique.

Dense optical flow also needs the assumption that the movement between the images/frames is very less, even up to a single pixel in distance. This leads to problems for the use of robotics [16], since there might happen chances of delay in robotic control and its decision making causing a further distance travel not being supported by dense optical flow. So what is then done is some good features are tracked along the frames which is sparse optical flow, but the problem of distance travel of one pixel assumption is still present, therefore to further improve it pyramidal Lucas-Kanade is used, which solves this problem. [17]

In order to reduce the complexity and computational processing sparse optical flow Lucas-Kanade pyramid implementation is considered.

3.2.2 Lucas-Kanade Pyramidal Optical Flow

There said to be three basic assumptions that are needed to be understood for the implementation of optical flow. They are that the image must has a constant brightness, i.e. the brightness on each every pixel value does not change with time, there has to exist temporal persistence or small movements that the robot must not move abruptly thereby changing the whole environment of its detection frame and the third to be the spatial coherence, which states that for a specific defined neighboring window size take it 3x3 or 10x10 of a pixel remains same throughout the operation of optical flow.

From analytical point of view let's assume that "*I*" is the intensity of a frame which is constant from t to t +δt. Then following constraint equation is obtained

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

Applying tailor series to the above constraint equation gives,

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

The tailor series is applied for only first derivative along time t. Therefore reconsidering the partial variables to new variables, as when distance y is derivated along time t and same case for x, we get

$$I_x u + I_y v + I_t = 0$$

Where u and v are the simple x and y distances of velocity vectors being made from their original point of coordinates x,y to their new position in x,y in time t.

Or $$I_x u + I_y v = - I_t$$

Now for the case when we consider the neighboring pixels inside a definite window for the case of optical flow we have,

$$I_x (q1)u + I_y (q1)v = - I_t (q1)$$

$$I_x (q2)u + I_y (q2)v = - I_t (q2)$$

$$\vdots$$

$$I_x (qn)u + I_y (qn)v = - I_t (qn)$$

Putting the above equation in the form of matrices or simply AV = b, and solving for velocity vector V.

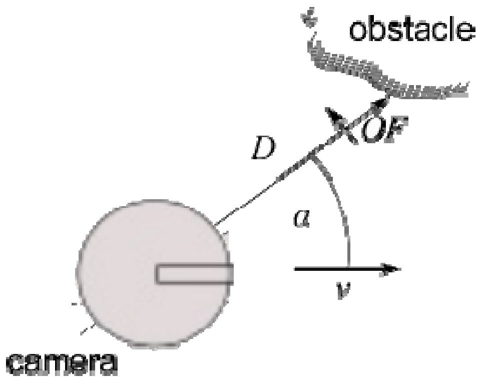
$$A = \begin{bmatrix} I_x(q_1) & I_y(q_1) \\ I_x(q_2) & I_y(q_2) \\ . & . \\ . & . \\ I_x(q_n) & I_y(q_n) \end{bmatrix} \quad b = \begin{bmatrix} -I_t(q_1) \\ -I_t(q_2) \\ . \\ . \\ -I_t(q_n) \end{bmatrix} \quad V = [u \ \ v]^T$$

Considering the above matrices one can tell that it has more equations than unknowns this it is usually over-determined. Thereby Lucas-Kanade obtains a compromise solution by least square principle.

$$V = (A^T A)^{-1} A^T b$$

This same velocity vector "*V*" is found by using the optical flow function used by OpenCV function "calcOpticalFlowPyrLK".

3.2.3 Obstacle to Robot distance Calculation



Considering the figure above, we can deliberately say that the optical flow is function of forward velocity of robot/camera, as

$$OF = \frac{v}{D} \cdot sin\alpha$$

Or rearranging it for calculating the distance, as

$$D = \frac{v}{OF} \cdot sin\alpha$$

Therefore when robot velocity, optical flow velocity vectors and angle of object to obstacle is known then the distance is also calculated.

3.2.4 Feature Detection

As already discussed that the optical flow that is to be implemented is sparse optical flow and to be precise the pyramidal Lucas-Kanade optical flow. Also that the lucas-kanade requires a set of good features as a requirement for its function. Therefore there is a need of a good feature detection technique for the working of lucas-kanade optical flow. For this purpose an optimum feature detection technique was required. And for that purpose the common edge detection techniques, the corner detection techniques like Harris-corner detection etc. and many other were implemented. Out of which the famous Shi-Tomasi technique was selected. It is basically a modified form of Harris- Corner detection technique [18], as both find the same good features but another parameter of minimum threshold is introduced in case of Shi-Tomasi technique, and for its implementation some parameters are to be defined which include the selection of mask size, the number of features to be detected, selection of image quality etc.

3.3 Obstacle Avoidance Strategy

The robot which is being used for the purpose of autonomous navigation is the famous iRobot Create. The vision is being provided to it by means of camera mounted on top of it and the algorithm being run wirelessly via a windows operating system. The algorithm is based on purely optical flow and the vision information provided to it, nothing else, no feedback from the sensors of the iRobot Create or any other sort of ultrasonic sonars/sensors. Therefore for the obstacle avoidance strategy, all the calculations are based on the data obtained from optical flow. The most common strategy for navigation using optical flow is the Balance Strategy [19], which states that,

- Calculate the amount of flow present at the half right side and the half left side of the image plane.
- Navigate towards the direction where flow is minimum, so as to avoid the obstacles.

Initially the Balance Strategy was implemented and results were driven out of it and further for a condition of moving straight, the image plane was divided into three equal frames. And according to their flows the navigation was done. Further improvements are also done, which are discussed in coming chapters.

3.4 Communication Methodology

The robot consists of a Bluetooth module for wireless communication. Therefore Bluetooth technology was used for the communication between the robot (iRobot Create) and the computer. The data sent is interpreted on the robot and the robot moves or acts accordingly, all done autonomously on-the-fly (real-time).

Using Bluetooth type communication, com port identification and selection is done. According to the com port designated as an outgoing port, linked between the robot and the computer. The 8-bit data is sent serially to the com port on which the Bluetooth module is connected to the pc. That data is then transmitted via Bluetooth to the Bluetooth module installed on iRobot create. The Robot interprets the provided data and make various actions accordingly. (Turning radius, Movement direction/speed etc.)

CHAPTER 4: EXPERIMENTATION

Here I would like to introduce the steps involved in implementation of the algorithm that has been developed. The initial steps whereas would be on simple basic images and then later on, it is conducted in real-time video imaging.

To test out the algorithm, images were taken by the camera mounted on iRobot Create at a resolution of 640 by 480 pixels. Initially the experimentation is done on different set of images, which includes;

4.1 Binary Images

The images were converted into binary (i.e. Black and White) and optical flow was implemented on them. The optical flow had a very vast amount of error included as the data provided was in black and white and as the optical flow is based on purely pixels intensity so the reason of implementing it on binary images did not be found useful.

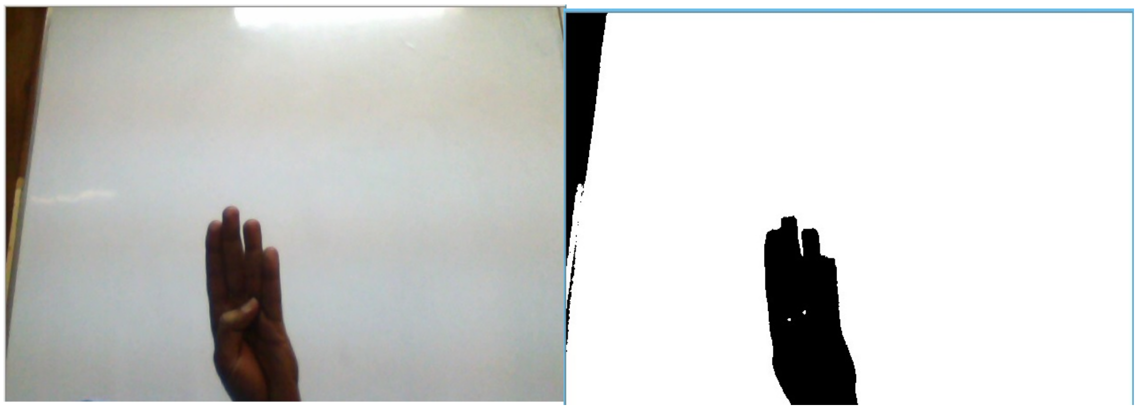


Figure 10. Binary Images (1)

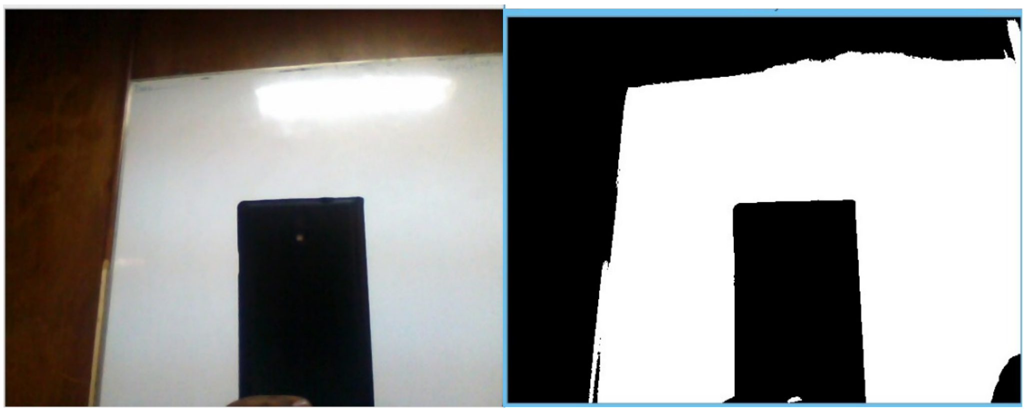


Figure 11. Binary images (2)

4.2 Color Segmentation

Suppose if there are number of obstacles that have a unique color, then by segmenting those obstacles based on their hue intensity, it is possible to avoid them, or they could be only tracked if required.

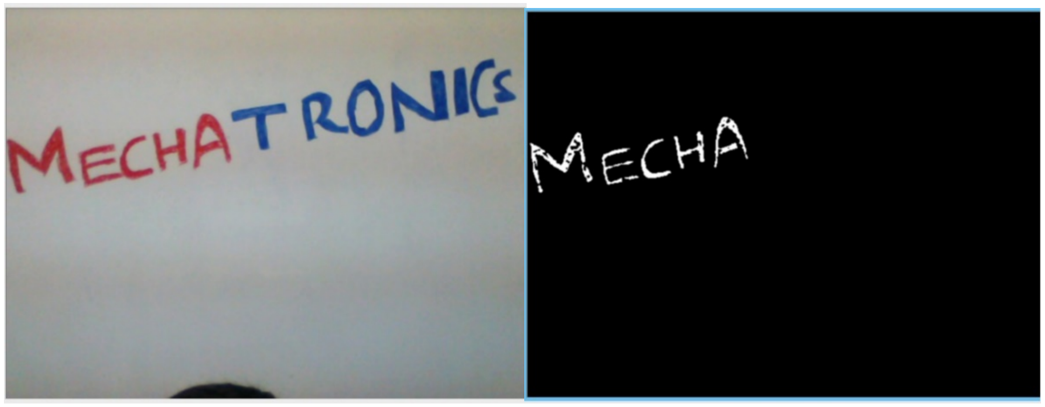


Figure 12. Color Segmented Images

4.3 Initial RGB implementation

Initially the results obtained when optical flow was applied to the RGB images, it looked something like this

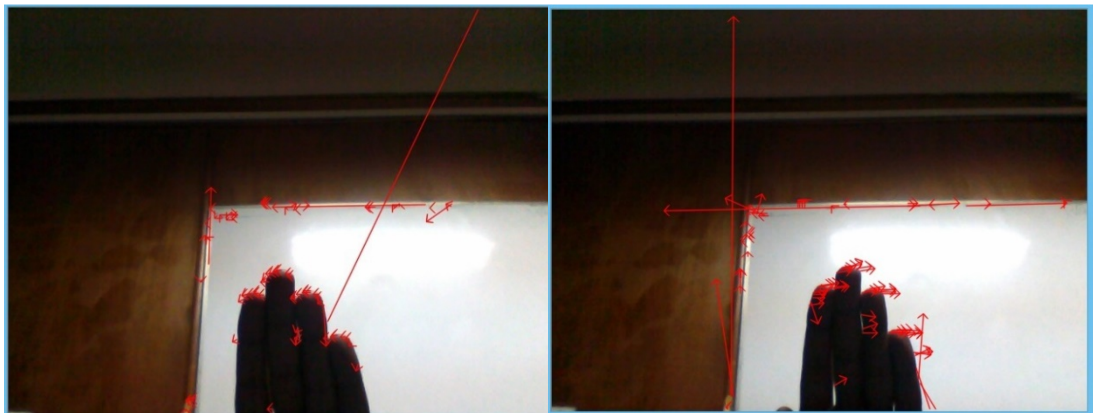


Figure 13. RGB with Optical Flow Images (1)

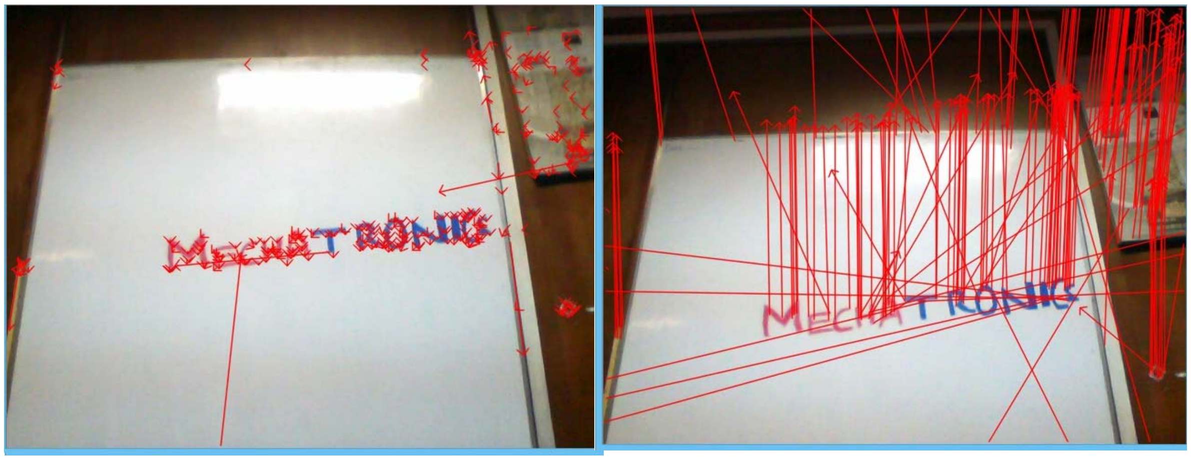


Figure 14. RGB with Optical Flow Images (2)

In the figure 14, the camera of the laptop was first placed at a static position and an image was taken then the camera was moved almost 20 degrees upward, therefore causing the optical flow vectors being generated in the upward direction. The algorithm was controlled in such a way that only after specific key pressed it would track the features therefore the images are produced as such.

CHAPTER 5: EXTENSIVE EXPERIMENTATIONS

Afterwards when optical flow was implemented, it was figured out that there was a quite huge amount of error involved in the data provided by the optical flow vectors. As many of them are produced at wrong angles, many of them had near infinite amount of velocity vector magnitudes, therefore to compensate them many of the improvements were made to the optical flow. The improvements are classified as such;

5.1 Optical Flow Outliers Removal

There was a huge amount of error involved in Lucas kanade pyramid method, as we can see here that there are numerous flow vectors that are made erroneously, to tackle the outliers obtained using the Lucas kanade pyramid method, modification in algorithm was made.

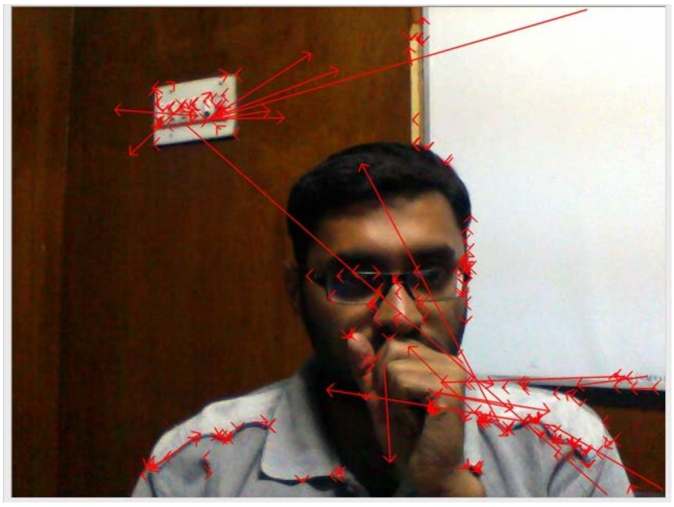


Figure 15. Optical Flow with Error

Let's consider the features from "previous" frames having coordinates as x1,y1 and the features tracked by Lucas kanade pyramid in "present" frame as x2,y2 , therefore for eliminating the ones which are erroneous, that those which show that sudden motion has taken place where as actually there has not, as being shown by the majority of the flow. Therefore for their detection and elimination, magnitude of all the features has been calculated, that is using the previous and present frame. After the calculation of the magnitudes of all the vectors, the

magnitude is summed up of all vectors and it is then divided by the total number of vectors, thereby providing with the average magnitude of all vectors.
The distance between feature flow vectors is calculated as

$$\sqrt{\sum (x2 - x1)^2 + (y2 - y1)^2}$$

After that another loop was formed for finding those features which were outliers, by applying an "if" command and checking out that which features are those that have magnitude greater than 2 times of the average magnitude of all vectors. And thereby discarding those features. Therefore after the implementation, satisfactory results were obtained, as shown below

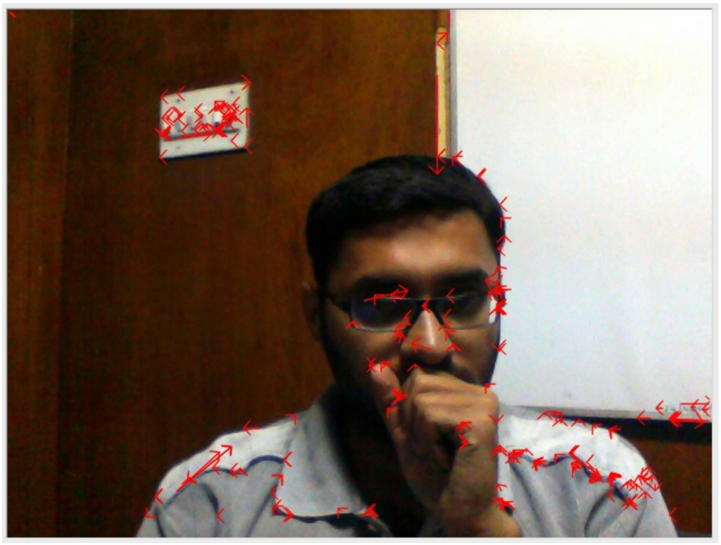


Figure 16. Optical Flow without Error

5.2 Easy Navigation using sub-division

The image plane is split into 3 equal vertically divided frames."Robustness" of each frame of image is summed up individually. The magnitude of all the optical flow velocity vectors are also calculated separately.The obstacles thereby that are far from the robot will have less optical flow as the ones that are near, and the weightage is given accordingly to the right side, middle side and the center one frame of the full plane. According to which the robot move against the flow that is greater.

Certain conditions were also implemented according the detection of the frames and velocity vectors being present in certain sided frames. The lowest weightage was given to the center frame as for the robot to collide with an obstacle the minimum distance it would be to move straight towards it, therefore only after the middle frame is 0.7 times the right frame and the left frame then only the robot was allowed to move forward. This 0.7 decrease of weightage of the middle frame was inserted to make sure that errors could also be compensated. The frames after division looked something like this

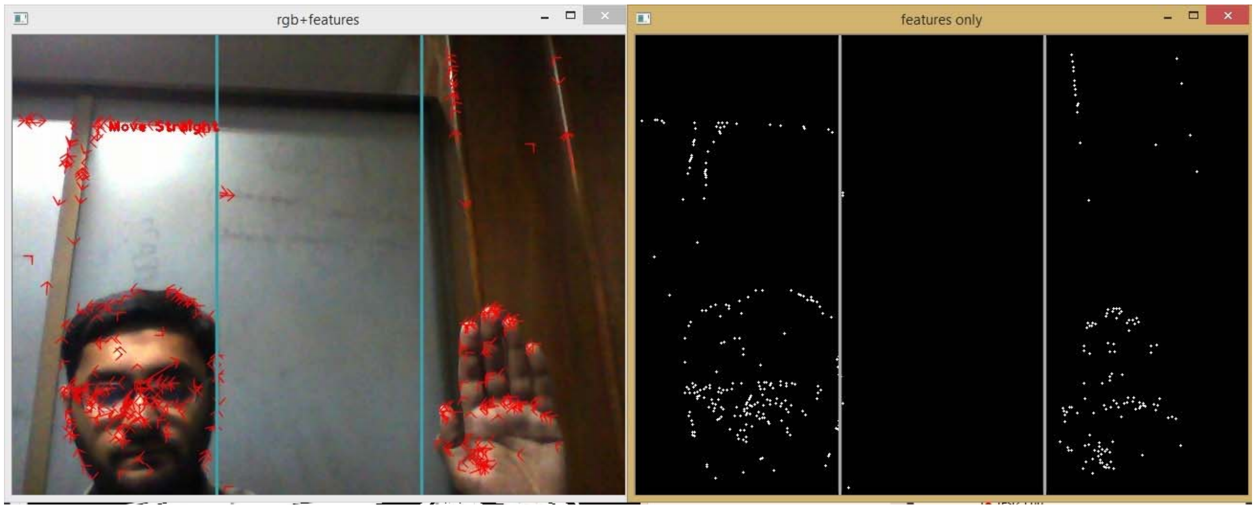


Figure 17. Easy Navigation using Sub-Division

5.3 Delaunay Triangulation Implementation

For the better judgment and estimation of object/obstacle representation/structure using the feature points obtained by Shi-Tomasi technique, Delaunay triangulation method was implemented. The basic need was for the better optimization and calibration of the robot environment. Therefore for all the 300 feature detected points, Delaunay triangulation was applied. Delaunay triangulation also helps with providing the intensity values of an obstacle, as number of vertices and edges are drawn using the feature points of the obstacle. The direct implementation of Delaunay wasn't possible using the available functions of Opencv, therefore rather other possible ways were considered for its implementation.

After the implementation of Delaunay triangulation, the results were achieved as,

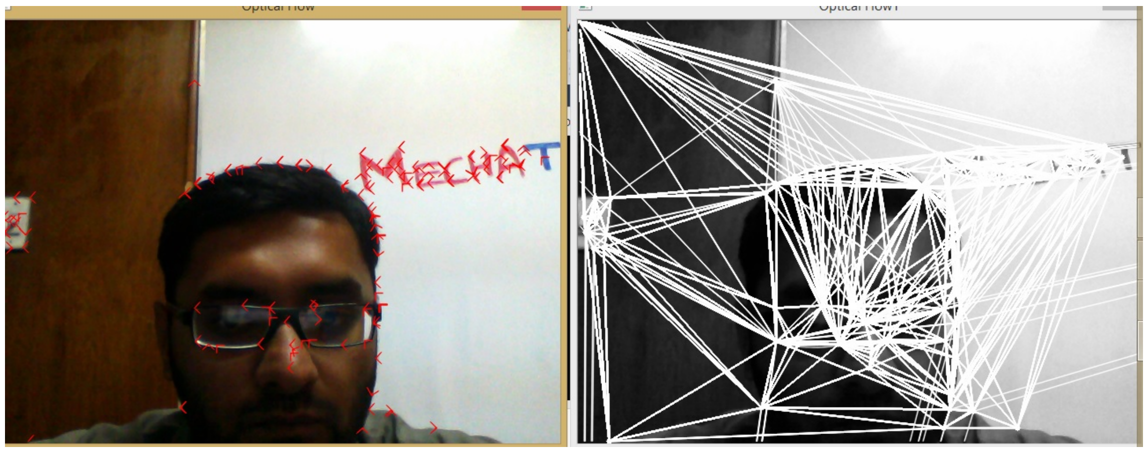


Figure 18. Delaunay Implementation

5.4 Delaunay Triangulation Improvement

After successful Delaunay implementation, there was a need for further implementations of conditions on Delaunay Triangulation for better results, as many edges of the triangles are generated from infinity and also from feature points placed (0,0) (which were those feature points that were considered as erroneous).

Therefore for omitting those erroneous features points from being considered in Delaunay triangulation, conditions were implied, which stated to not include the feature points that are at location (0,0) and which primary connects with points at infinity.After such improvements the results obtained were much efficient and satisfactory.

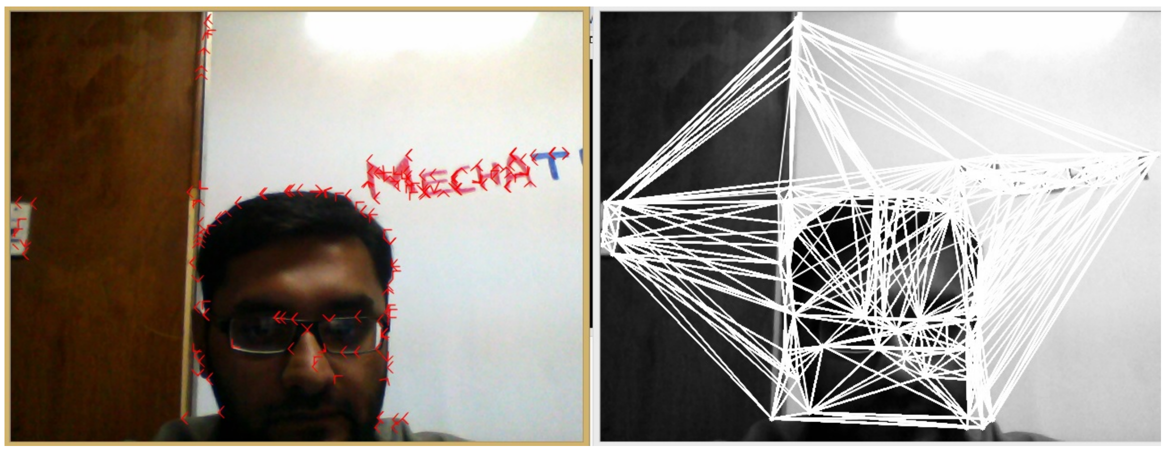


Figure 19. Delaunay Improvements

5.5 Delaunay Triangulation Sub-Division

Delaunay triangulation was further improved by dividing the triangulation made by itself into 3 frames. Considering the case when Delaunay is used for the complete frame at once.

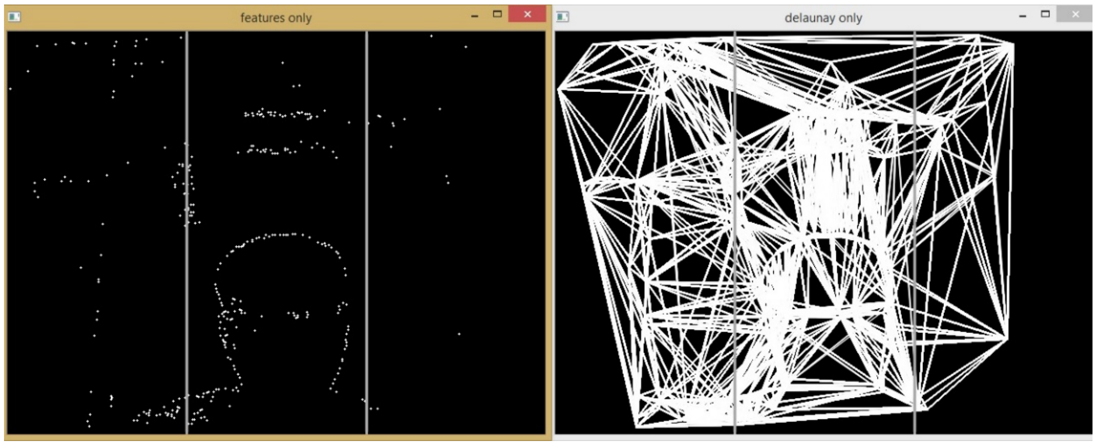


Figure 20. Delaunay Triangulation without Sub-Division

One can see that all the good feature points are used by Delaunay and every point is interlinked with its neighbor, whether it even lies in all 3 frames, example, a feature is present in frame one, and its neighboring feature points to make a triangle are present in frame two and frame three, therefore for making a triangle the Delaunay creates a triangle which is present in all the 3 frames.

Now let’s separate the all 3 frames, so that only the adjacent feature points present in one frame can be used for making a Delaunay triangulation for that respective feature point.After successful segmentation, the results were such as

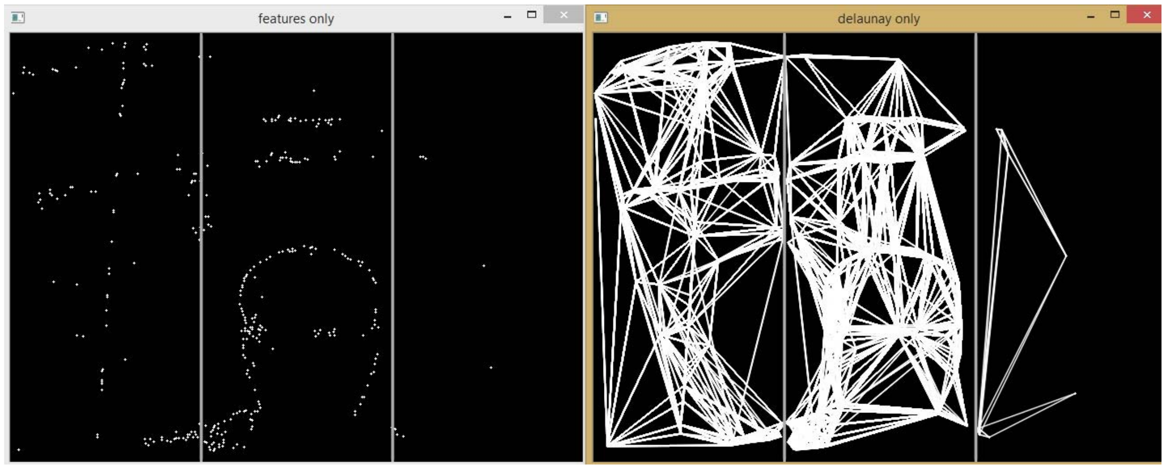


Figure 21. Delaunay Triangulation with Sub-Division

One can clearly see the amount of improvement being obtained by segmenting the frames in Delaunay triangulation.

5.6 Steering Control Conditions

By taking into account the steering control conditions being mentioned in the flow chart, it becomes pretty clear that there are three main conditions to fulfill for the navigation being able to be performed. It means that the three conditions that are, the magnitude of velocity vectors and good features and calculating of Delaunay Triangulation, for them all to point towards one direction will let the robot move in that specific direction but if one of them points towards any other direction, then only the good features and Delaunay calculations are considered, and the robot moves according to the later decision, it has been found that almost 90% of the time the decision is already made by these conditions but if somehow the good features and Delaunay even fail to compromise on single direction, then the robot stops for that respective iteration and then continues to look further for next iteration.

5.7 Practical Experiments

The iRobot create was made to avoid obstacles in the real world in different scenarios, initially the windows operated laptop was placed fixed and the robot was allowed to move according to the actions being performed infront of the laptop as the robot was moving instead of the obstacles.

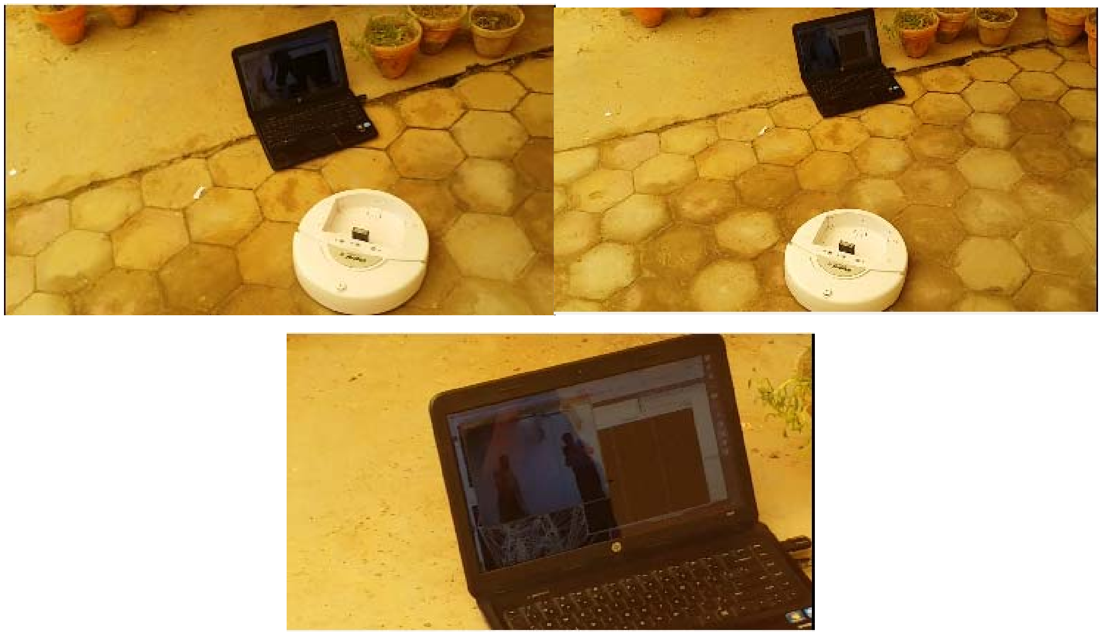


Figure 22. Practical Implementation Examples (1)

Then the laptop was placed on the robot and the robot avoided the single obstacle placed directly in front of him.

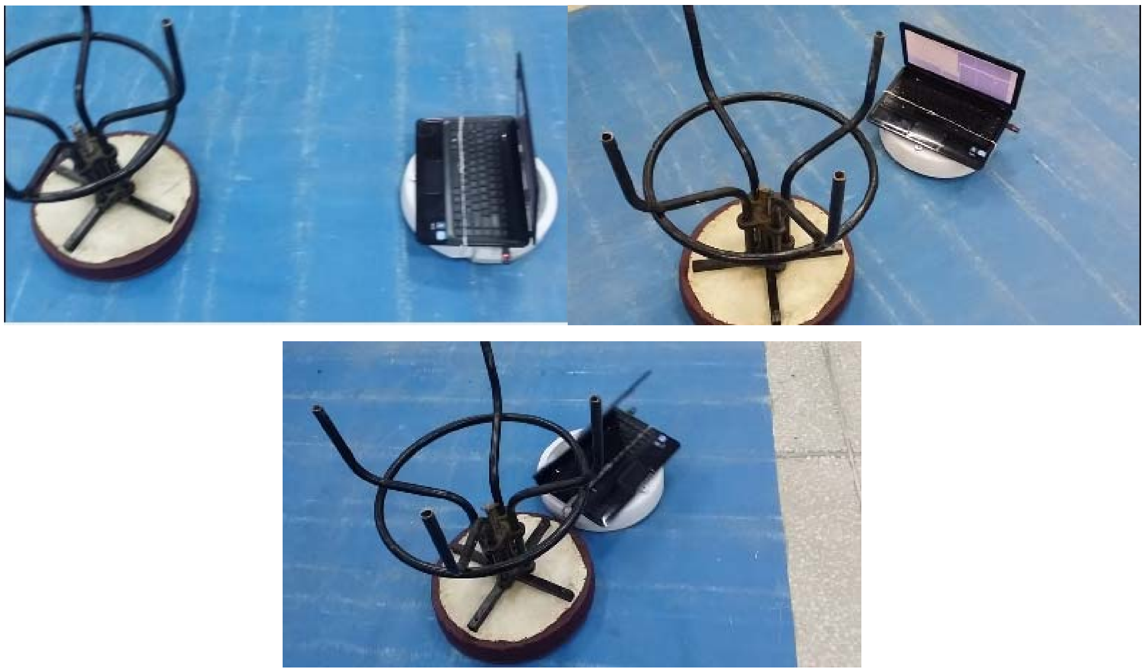


Figure 23. Practical Implementation Examples (2)

Afterwards a camera was mounted on the iRobot Create and two obstacles were placed in front of it, due to their flow being generated equally on both sides of the image plane, and less flow being generated in the middle, the robot navigated towards the front but when it came near enough the middle frame flow became greater and thus the robot navigated away from the obstacles.

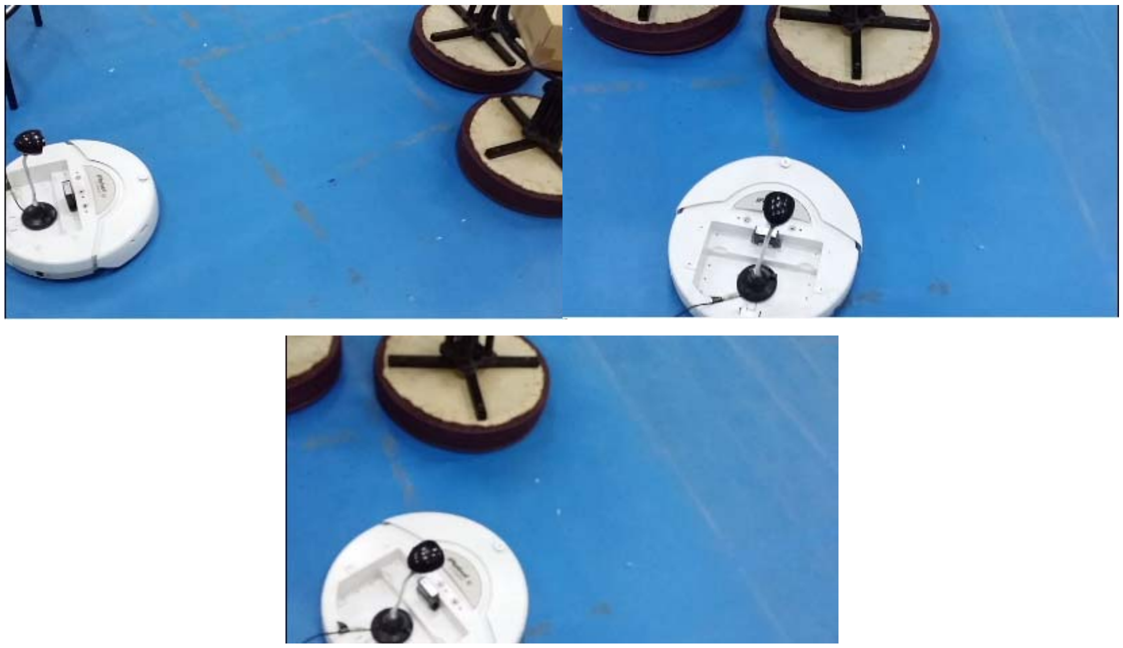


Figure 24. Practical Implementation Examples (3)

CHAPTER 6: CONCLUSION AND FUTURE WORK

An intelligent navigation and control of mobile robot has been successfully developed. The objectives that were involved in reaching this point included mainly the development of an intelligent algorithm for the autonomous movement of the robot and the interfacing of the robot and the windows platform wirelessly, i.e. via the Bluetooth module.

The algorithm was made using the OpenCV library, linked with the Visual Studio 2013 environment. The algorithm was made as a single ".cpp" file, whereas to control the robot there need to introduce two files, rs232.cpp and rs232.h, the C++ and header files respectively. They were needed for the control of the robot, having sent data serially via Bluetooth module.

The robot in return receives the data being transmitted using the algorithm via Bluetooth module, and it acts according to the instruction being provided for its navigation.

6.1 Future Work

The robot was successfully controlled using windows operating system by OpenCV libraries, the same libraries were also linked in the eclipse environment for android platform. But the libraries were not stable and robust enough for the kind of autonomous work that was required out of it. The developers of OpenCV libraries are continuously working on improving the functions being used by android of OpenCV and is therefore being made robust and more efficient. Therefore making small adjustments to the present code written in C language, can be directly used by Android platform, if the libraries are linked successfully and the support of most of the functions being used are made properly.

Now the iRobot Create has been given vision,then number of tasks and work can be done using the iRobot Create. Which could include the improvement of a path planning being introduced where the video feedback is being continuously monitored and a predefined path is being navigated without the collision of the obstacles being found. And even further instead a GPS module could be used and feedback being provided by it could also help in reaching a desired target/goal.

In this dissertation, obstacle avoidance has been done, as mentioned in previous chapters, that obstacle tracking is also possible, as optical flow key feature include tracking behavior.

Therefore a color segmented object tracking or tracking a specific structure based object is also possible.

An introduction of voice controlled robot navigation is also possible. Which can be done using the Microsoft Speech Recognition API for windows or Android Speech Recognition API for android platform.

APPENDIX A

AlgorithmCode

```
/* --Intelligent Navigation and Control of Mobile Robot--
* Written in C Using OpenCV Library
* Written by Muhammad Zohaib
*/
#pragma comment(lib,"Winmm.lib")
#include "opencv2/imgproc/imgproc_c.h"
#include "opencv2/video/tracking.hpp"
#include "opencv2/highgui/highgui.hpp"
#include "rs232.h"
#include <Windows.h>
#include <iostream>
using namespace std;
static const double pi = 3.14159265358979323846;

inline static double square(int a)
{
	return a * a;
}
double NormCalculator(CvPoint p, CvPoint q)
{
	return sqrt(square(p.x - q.x) + square(p.y - q.y));
}


inline static void allocateOnDemand(IplImage **img, CvSize size, int depth, int channels)
{
	if (*img != NULL)	return;

	*img = cvCreateImage(size, depth, channels);
	if (*img == NULL)
	{
		fprintf(stderr, "Error: Couldn't allocate image.  Out of memory?\n");
		exit(-1);
	}
}

void Create_init(int port_numb, int baud)
{
	if (RS232_OpenComport(port_numb, baud))
	{
		printf("Can not open comport\n");
	}
	else
	{
```

```
            printf("Port opened successfully\n\n");
        }
        unsigned char initial[2] = { 128, 132 };

        printf("Setting CREATE to FULL CONTROL mode\n\n");
        int a = RS232_SendBuf(port_numb, initial, 2);
        if (a == -1)
        {
            printf("Error in Setting CREATE to Full Control mode\n\n");
        }

        else
        {
            printf("CREATE set into Full Control mode\n\n");
        }
}


void FwdVelRad(int port_numb, unsigned int speed, unsigned int rad)
{
        unsigned char fwdvel[2];
        unsigned char radm[2];
        unsigned int radmm;
        int a;
        unsigned int spd;
        spd = speed;
        cout << spd << endl;
        fwdvel[0] = spd & 0xFF;
        fwdvel[1] = (spd >> 8) & 0xFF;
        cout << "fwdvel1   " << fwdvel[0] << endl;
        cout << "fwdvel2   " << fwdvel[1] << endl;

        radmm = rad;
        radm[0] = radmm & 0xFF;
        radm[1] = (radmm >> 8) & 0xFF;

        //unsigned char dr[1]={137};
        unsigned char cmd[5] = { 137, fwdvel[0], fwdvel[1], radm[0], radm[1] };  // DRIVE
        int n = RS232_SendBuf(port_numb, cmd, 5);
        if(n ==-1)
        {
            printf("Error in send command for drive\n\n");
            exit(EXIT_FAILURE);
        }
        else
        {
            printf("roomba moving...  ;)\n\n");
        }

}
```

```
int main(void)
{
	int
		cport_nr = 8,		/* /dev/ttyS0 (COM1 on windows) */
		bdrate = 57600;	/* 57600 baud */
	int inf = 1000;

	Create_init(cport_nr, bdrate); //initializing the serial communication.


	CvCapture *input_video = cvCaptureFromCAM(1);
	if (input_video == NULL)
	{
		fprintf(stderr, "Error: Can't open video.\n");
		return -1;
	}

	/* Read the video's frame size out of the AVI. */
	CvSize frame_size;
	frame_size.height =
		(int)cvGetCaptureProperty(input_video, CV_CAP_PROP_FRAME_HEIGHT);
	frame_size.width =
		(int)cvGetCaptureProperty(input_video, CV_CAP_PROP_FRAME_WIDTH);

	int framecheck = 0;
	int leftcount = 0, rightcount = 0, centercount = 0, leftcenter = 0, rightcenter = 0;
	int draw_main = true;

	long current_frame = 0;
	do
	{
		static IplImage *frame = NULL, *frame1 = NULL, *frame1_1C = NULL, *frameD =
NULL, *frameF = NULL, *frame2_1C = NULL, *eig_image = NULL, *temp_image = NULL,
*pyramid1 = NULL, *pyramid2 = NULL;
		framecheck = 0;



		frame = cvQueryFrame(input_video);
		if (frame == NULL)
		{
			fprintf(stderr, "Error: Hmm. The end came sooner than we thought.\n");
			return -1;
		}

		//create a frame for delaunay triangles only with zero filled
		frameD = cvCreateImage(frame_size, IPL_DEPTH_8U, 1);
		cvZero(frameD);
		//create a frame for features only with zero filled
		frameF = cvCreateImage(frame_size, IPL_DEPTH_8U, 1);
		cvZero(frameF);
		frame1_1C = cvCreateImage(frame_size, IPL_DEPTH_8U, 1);
		cvConvertImage(frame, frame1_1C, 0);
		allocateOnDemand(&frame1, frame_size, IPL_DEPTH_8U, 3);
		cvConvertImage(frame, frame1, 0);
		frame = cvQueryFrame(input_video);
		if (frame == NULL)
		{
			fprintf(stderr, "Error: Hmm. The video does not exist.\n");
			return -1;
		}
		allocateOnDemand(&frame2_1C, frame_size, IPL_DEPTH_8U, 1);
		cvConvertImage(frame, frame2_1C, 0);

		/* Shi and Tomasi Feature Tracking! */
		allocateOnDemand(&eig_image, frame_size, IPL_DEPTH_32F, 1);
		allocateOnDemand(&temp_image, frame_size, IPL_DEPTH_32F, 1);
		int number_of_features = 300;
		const int total_features = 300;
		CvPoint2D32f frame1_features[total_features];
		cvGoodFeaturesToTrack(frame1_1C, eig_image, temp_image, frame1_features,
&number_of_features, .01, .01, NULL, 3);
		CvPoint2D32f frame2_features[total_features];
		char optical_flow_found_feature[total_features];
		float optical_flow_feature_error[total_features];
		CvSize optical_flow_window = cvSize(3, 3);
		CvTermCriteria optical_flow_termination_criteria
			= cvTermCriteria(CV_TERMCRIT_ITER | CV_TERMCRIT_EPS, 20, .1);
		allocateOnDemand(&pyramid1, frame_size, IPL_DEPTH_8U, 1);
		allocateOnDemand(&pyramid2, frame_size, IPL_DEPTH_8U, 1);
		cvCalcOpticalFlowPyrLK(frame1_1C, frame2_1C, pyramid1, pyramid2,
frame1_features, frame2_features, number_of_features, optical_flow_window, 5,
optical_flow_found_feature, optical_flow_feature_error, optical_flow_termination_criteria, 0.01);
		double average_features[total_features];
		double average_fea_add = 0;
		cv::Scalar delaunay_color(255, 255, 255);
		cv::Rect rect(0, 0, cvGetCaptureProperty(input_video,
CV_CAP_PROP_FRAME_WIDTH), cvGetCaptureProperty(input_video,
CV_CAP_PROP_FRAME_HEIGHT));
		cv::Subdiv2D subdiv(rect);
		CvPoint p, q;
		for (int i = 0; i < number_of_features; i++)
		{
			p.x = (int)frame1_features[i].x;
			p.y = (int)frame1_features[i].y;
			q.x = (int)frame2_features[i].x;
			q.y = (int)frame2_features[i].y;
			average_features[i] = NormCalculator(p, q);
			average_fea_add = average_fea_add + average_features[i];
		}
```

```
        double average_fea = average_fea_add / total_features;


            for (int i = 0; i < number_of_features; i++)
            {
                draw_main = true;
                if (average_features[i] > (average_fea * 0.5))
                {
                    draw_main = false;
                    frame1_features[i].x = 0.0;
                    frame2_features[i].x = 0.0;
                }

                if (draw_main == false) continue;
                if (frame2_features[i].x > cvGetCaptureProperty(input_video,
CV_CAP_PROP_FRAME_WIDTH) || frame2_features[i].y > cvGetCaptureProperty(input_video,
CV_CAP_PROP_FRAME_HEIGHT) || frame2_features[i].x < 0 || frame2_features[i].y < 0)
                {
                    frame1_features[i].x = 0.0;
                    frame2_features[i].x = 0.0;
                    continue;
                }

                /* If Pyramidal Lucas Kanade didn't really find the feature, skip it. */
                if (optical_flow_found_feature[i] == 0)
                {
                    frame1_features[i].x = 0.0;
                    frame2_features[i].x = 0.0;
                    continue;
                }
                //skip those features which contains error
                if (optical_flow_feature_error[i] == 1)
                {
                    frame1_features[i].x = 0.0;
                    frame2_features[i].x = 0.0;
                    continue;
                }

                int line_thickness;                     line_thickness = 1;
                CvScalar line_color;                line_color = CV_RGB(255, 0,
0);

                p.x = (int)frame1_features[i].x;
                p.y = (int)frame1_features[i].y;
                q.x = (int)frame2_features[i].x;
                q.y = (int)frame2_features[i].y;

                cv::Point2f fp = q;
                cvCircle(frameD, fp, 1, delaunay_color, CV_FILLED, 1, 0);
```

```
					subdiv.insert(fp);
					cv::vector<cv::Vec6f> triangleList;
					subdiv.getTriangleList(triangleList);
					cv::vector<cv::Point> pt(3);
					for (size_t i = 0; i < triangleList.size(); i++)
					{
						cv::Vec6f t = triangleList[i];
						pt[0] = cv::Point(cvRound(t[0]), cvRound(t[1]));
						pt[1] = cv::Point(cvRound(t[2]), cvRound(t[3]));
						pt[2] = cv::Point(cvRound(t[4]), cvRound(t[5]));
						int draw = true;
						for (int i = 0; i<3; i++)
						{
							if (pt[i].x>cvGetCaptureProperty(input_video, 
CV_CAP_PROP_FRAME_WIDTH) || pt[i].y>cvGetCaptureProperty(input_video, 
CV_CAP_PROP_FRAME_HEIGHT) || pt[i].x < 0 || pt[i].y < 0)
								draw = false;
							if ((q.x < 214) && (pt[i].x  > 214))
								draw = false;
							if (((q.x > 213) && (q.x < 428)) && ((pt[i].x > 427) || 
(pt[i].x < 214)))

								draw = false;
							if ((q.x > 427) && (pt[i].x < 428))
								draw = false;
						}
						if (draw)
						{
							cvLine(frameD, pt[0], pt[1], delaunay_color, 1, CV_AA, 
0);
							cvLine(frameD, pt[1], pt[2], delaunay_color, 1, CV_AA, 
0);
							cvLine(frameD, pt[2], pt[0], delaunay_color, 1, CV_AA, 
0);
						}
					}

					cvCircle(frameF, p, 1, delaunay_color, CV_FILLED, 8, 0);
					double angle;		angle = atan2((double)p.y - q.y, (double)p.x - 
q.x);
					double hypotenuse;	hypotenuse = sqrt(square(p.y - q.y) + square(p.x 
- q.x));

					if (hypotenuse > 5)
					{
						if ((p.x < 214 && q.x < 214))
							leftcount = leftcount + hypotenuse;
						if (((p.x > 213) && (p.x < 428)) && ((q.x > 213) && (q.x < 
428)))

							centercount = centercount + hypotenuse;
						if ((p.x > 427) && (q.x > 427))
							rightcount = rightcount + hypotenuse;
```

```
                    if ((p.x < 214) && ((q.x < 427) && (q.x > 213 )))
                        leftcenter = leftcount + hypotenuse;
                    if ((p.x > 427) && ((q.x < 427) && (q.x > 213)))
                        rightcenter = rightcount + hypotenuse;
                    if ((q.x > 427) && ((p.x < 427) && (p.x > 213)))
                        rightcount = rightcount + hypotenuse;
                    if ((q.x < 213) && ((p.x < 427) && (p.x > 213)))
                        leftcount = leftcount + hypotenuse;
                }
                q.x = (int)(p.x - 1 * hypotenuse * cos(angle));
                q.y = (int)(p.y - 1 * hypotenuse * sin(angle));

                cvLine(frame1, p, q, line_color, line_thickness, CV_AA, 0);
                p.x = (int)(q.x + 9 * cos(angle + pi / 4));
                p.y = (int)(q.y + 9 * sin(angle + pi / 4));
                cvLine(frame1, p, q, line_color, line_thickness, CV_AA, 0);
                p.x = (int)(q.x + 9 * cos(angle - pi / 4));
                p.y = (int)(q.y + 9 * sin(angle - pi / 4));
                cvLine(frame1, p, q, line_color, line_thickness, CV_AA, 0);

            }
            cv::Point rectstart(0, 0);
            cv::Point rect1start(213, 0);
            cv::Point rect1end(213, 480);
            cv::Point rect2start(427, 0);
            cv::Point rect2end(427, 480);
            cv::Point rectend(640, 480);

            cvLine(frameD, rect1start, rect1end, cvScalar(170, 160, 60), 2, CV_AA, 0);
            cvLine(frameD, rect2start, rect2end, cvScalar(170, 160, 60), 2, CV_AA, 0);
            cvLine(frameF, rect1start, rect1end, cvScalar(170, 160, 60), 2, CV_AA, 0);
            cvLine(frameF, rect2start, rect2end, cvScalar(170, 160, 60), 2, CV_AA, 0);
            cvLine(frame1, rect1start, rect1end, cvScalar(170, 160, 60), 2, CV_AA, 0);
            cvLine(frame1, rect2start, rect2end, cvScalar(170, 160, 60), 2, CV_AA, 0);

            cv::Mat frame1D(frameD, cv::Rect(rectstart.x, rectstart.y, rect1end.x,
rect1end.y));
            cv::Mat frame2D(frameD, cv::Rect(rect1start.x, rect1start.y, (rect1end.x + 1),
rect1end.y));
            cv::Mat frame3D(frameD, cv::Rect(rect2start.x, rect2start.y, rect1end.x,
rect1end.y));
            float D1 = cv::countNonZero(frame1D == 255);
            float D2 = cv::countNonZero(frame2D == 255);
            float D3 = cv::countNonZero(frame3D == 255);
            float DT = D1 + D2 + D3;

            cv::Mat frame1F(frameF, cv::Rect(rectstart.x, rectstart.y, rect1end.x,
rect1end.y));
            cv::Mat frame2F(frameF, cv::Rect(rect1start.x, rect1start.y, (rect1end.x + 1),
rect1end.y));
```

```
			cv::Mat frame3F(frameF, cv::Rect(rect2start.x, rect2start.y, rect1end.x,
rect1end.y));
			float F1 = cv::countNonZero(frame1F == 255);
			float F2 = cv::countNonZero(frame2F == 255);
			float F3 = cv::countNonZero(frame3F == 255);
			float FT = F1 + F2 + F3;
			fprintf(stderr, "Number of Delaunay pixels,");
			cout << " DT = " << DT << " D1 = " << D1 << " D2 = " << D2 << " D3 = " <<
D3 << endl;
			fprintf(stderr, " \n\n Number of Feature pixels,");
			cout << " FT = " << FT << " F1 = " << F1 << " F2 = " << F2 << " F3 = " << F3
<< endl;

			std::stringstream s;
			float totalleft = (F1 + ((D1 / DT) * 100));
			float totalright = (F3 + ((D3 / DT) * 100));
			float totalcenter = (F2 + ((D2 / DT) * 100));
			float totaldiff = totalleft - totalright;
			cv::norm(totaldiff);

				if (((0.7 * centercount < rightcount) && (0.7 * centercount < leftcount))
&& ((0.7 * totalcenter) < totalright && (0.7 * totalcenter)<totalleft))
				{
					(s << "Move Straight");
					FwdVelRad(cport_nr, 2, 5);
					framecheck = 1;
					Sleep(150);
				}
				else
				{
					if ((leftcount > rightcount) && (totalleft > totalright))
					{
						(s << "Move Straight+Right");
						FwdVelRad(cport_nr, 2, -1); //half radius in clockwise
						framecheck = 1;
						Sleep(60);
					}
					else if ((rightcount > leftcount) && (totalright > totalleft))
					{
						(s << "Move Straight+Left");
						FwdVelRad(cport_nr, 2, 0); //half radius in c.clockwise
						framecheck = 1;
						Sleep(60);
					}

				}

			if (framecheck == 0)

				if ((0.7 * totalcenter) < totalright && (0.7 * totalcenter)<totalleft)
			{
```

```
                        (s << "Move Straight");
                        FwdVelRad(cport_nr, 2, 5);
                        Sleep(150);
                }
                else
                {
                        if (totalleft > totalright)
                        {
                                (s << "turn Right");
                                FwdVelRad(cport_nr, 2, -1);
                                Sleep(60);
                        }
                        else if (totalright > totalleft)
                        {
                                (s << "turn Left");
                                FwdVelRad(cport_nr, 2, 0);
                                Sleep(60);
                        }
                }

                cv::putText((cv::Mat)frame1, s.str(), cv::Point2f(100, 100),
cv::FONT_HERSHEY_PLAIN, 1, cv::Scalar(0, 0, 255, 255), 2);

                cvNamedWindow("delaunay only", CV_WINDOW_AUTOSIZE);
                cvShowImage("delaunay only", frameD);
                cvNamedWindow("features only", CV_WINDOW_AUTOSIZE);
                cvShowImage("features only", frameF);
                cvNamedWindow("rgb+features", CV_WINDOW_AUTOSIZE);
                cvShowImage("rgb+features", frame1);

        int key_pressed = cvWaitKey(1);
        FwdVelRad(cport_nr, 0, 0);
        Sleep(400);
    }while (GetAsyncKeyState(VK_ESCAPE) == 0);
}
```

Bluetooth Control code:

Files for the Bluetooth control codes having been placed on a CD and attached with the dissertation.

REFERENCES

[1] A. Dev. *Visual Navigation* on Optical Flow. PhD thesis, University of Amsterdam, September 1998.

[2] B. Kelly. Structure from stereo vision using optical flow. Master's thesis, University of Canterbury, November 2006.

[3] K. Kanatani. Self-calibration from optical flow and its reliability evaluation.In IAPR Workshop on Machine Vision Applications (MVA2000), pages 443–446, 2000.

[4] I. Ulrich and I. R. Nourbakhsh. Appearance-based obstacle detection with monocular color vision. In *AAAI/IAAI'00*, pages 866–871, 2000.

[5] C. Bills, J. Chen, and A. Saxena. Autonomous mav flight in indoor environments using single image perspective cues. *International Conference on Robotics and Automation (ICRA*), 2011.

[6] F. Blais. Review of 20 years of range sensor development. *Journal of Electronic Imaging*, 13(1), 2004.

[7] S. F. te Pas. *Perception of Structure in Optical Flow Fields*. PhD thesis, University of Utrecht, September 1996.

[8] M. Varga. Practical Image Processing and Computer Vision, chapter 13. John Wiley & Sons, 2009.

[9] B. D. Lucas and T. Kanade. Optical Navigation by the Method of Differences. In International Joint Conference on Artificial Intelligence, pages 981–984.

[10] G. Bleser and G. Hendeby. Using optical flow as lightweight slam alternative. Mixed and Augmented Reality, IEEE / ACM International Symposium on, 0:175–176, 2009. ISBN 978-1-4244-5390-0.

[11] M. Zucchelli, J. Santos Victor, and H. Christensen. Constrained structure and motion estimation from optical flow. Pages I: 339–342, 2002.

[12] D. J. Fleet and Y. Weiss. Mathematical Models in Computer Vision: The Handbook (Optical Flow Estimation), chapter 15, pages 239–258. Springer, 2005.

[13] J. A. Saunders and D. C. Niehorster. A bayesian model for estimating observer translation and rotation from optic flow and extra-retinal input. Journal of Vision, 10(10):1–22, 2010.

[14] D. Kane, P. Bex, and S. Dakin. Quantifying “the aperture problem” for judgments of motion direction in natural scenes. Journal of Vision, 11(3): 1–20, 2011.

[15] S. J. Huston and H. G. Krapp. Visuomotor transformation in the fly gaze stabilization system. PLoS Biol, 2008.

[16] J. Barron, D. Fleet, and S. Beauchemin. Performance of optical flow techniques. IJCV, 12(1):43–77, 1994

[17] J.-Y. Bouguet. Pyramidal implementation of the Lucas kanade feature tracker. Intel Corporation, Microprocessor Research Labs, 2000.

[18] “Vision Based Collision Avoidance System for UAVs”. By Prof K Dana, Nakul N., Arjun K.

[19] Kahlouche Souhila and Achour Karim. Optical flow based robot obstacle avoidance. International Journalof Advanced Robotic Systems, 4(1):13–16, 2007

Completion Certificate

It is to certify that the thesis titled "**Intelligent Navigation and Control of Mobile Robots Using Android Platform**" submitted by Regn. No. **2011-NUST-MS-PHD-Mts-25**, **Muhammad Zohaib** of **MS-70Mechatronics Engineering** is complete in all respects as per the requirements of Main Office, NUST (Exam branch).

Supervisor: ___________

Dr. Kunwar Faraz Ahmed

Date: ____ May, 2014