CS 101 - PROJECT SPRING 2015
FINAL REPORT

OPTIMISED BALL COLLECTOR

GROUP - 418

Team Members

***Sudeep Salgia,* 14D070011**

***Aviral Kumar,* 140070031**

***Sohum Dhar,* 140070001**

***Muthamizh Selvan*,140110091**

TABLE OF CONTENTS

PROBLEM STATEMENT

It is often necessary to go to a set of certain points in a network and minimization of effort/energy is always a matter of concern. Thus we aim to address this requirement in a prototype scenario.

Thus, using a bot, we intend to collect balls spread over an area ensuring that the distance that the bot travels is minimized.

In order to achieve this, a top view of the arena is provided to the code which connects with the bot and sends the path to the bot which it has to traverse in order to achieve the total path length minimization.

REQUIREMENTS

A) Hardware Requirements:

1. FIREBIRD V : 1 FIREBIRD V robot manufactured by NEX- ROBOTICS
2. Camera : Mobile Camera / any Camera would suffice
3. Balls : Any number of identical balls
4. X-Bee Pair and adapter : To maintain communication system between the bot and the laptop
5. Sharp sensors : A pair of sharp sensors (already attached on the bot)

B) Software Requirements:

1. OpenCV : For finding out the balls' coordinates in the image sent by the camera
2. Atmel Studio : For bot Embedded C programming
3. Code::Blocks 13.12 : Code::Blocks IDE for writing and executing C++ programs.
4. GTK+2.24 : For running the GUI

FUNCTIONALITY

a) Determining path and generating file containing set of instructions:

Using the image processing, on the image provided by the user, the program finds the centre of the balls and generates a file containing the set of instructions containing distance to be traversed and angle to be rotated, to make the bot traverse on the shortest path to collect all the balls.

b) Making the bot moves using serial communication:

Using the XBee Module, the bot is sent instructions in real time to traverse in the shortest path to collect all the balls.

c) Interfacing with the user:

The program opens a window asking only for the path of the image to be used. All the codes, for image processing, shortest path algorithm, serial communication, are run with a simple press of enter key!

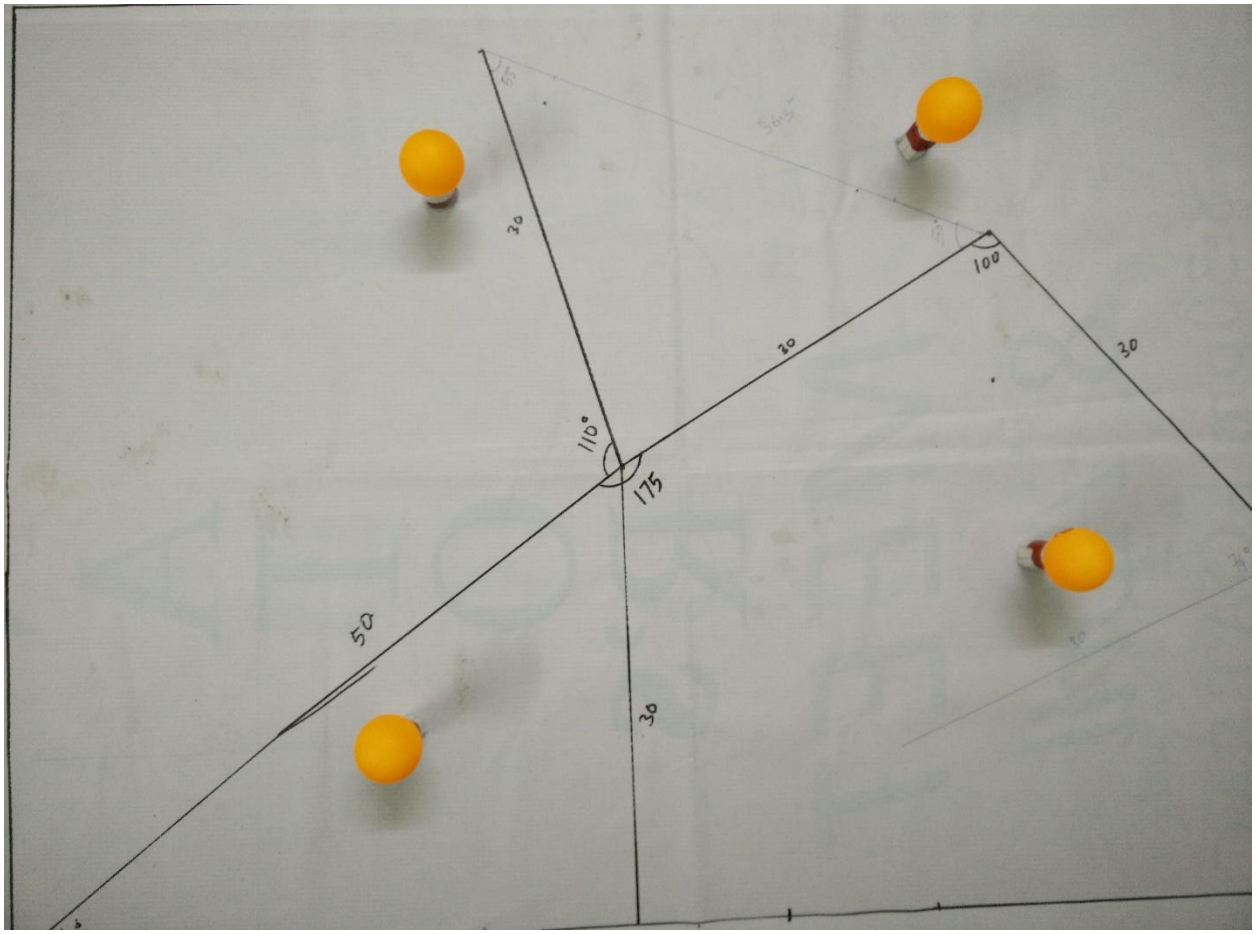
TESTING STRATEGY

Module 1: Image Processing:

The Image Processing code was tested with images of up to 6 balls in a given arena with different orientations.

They were

1.

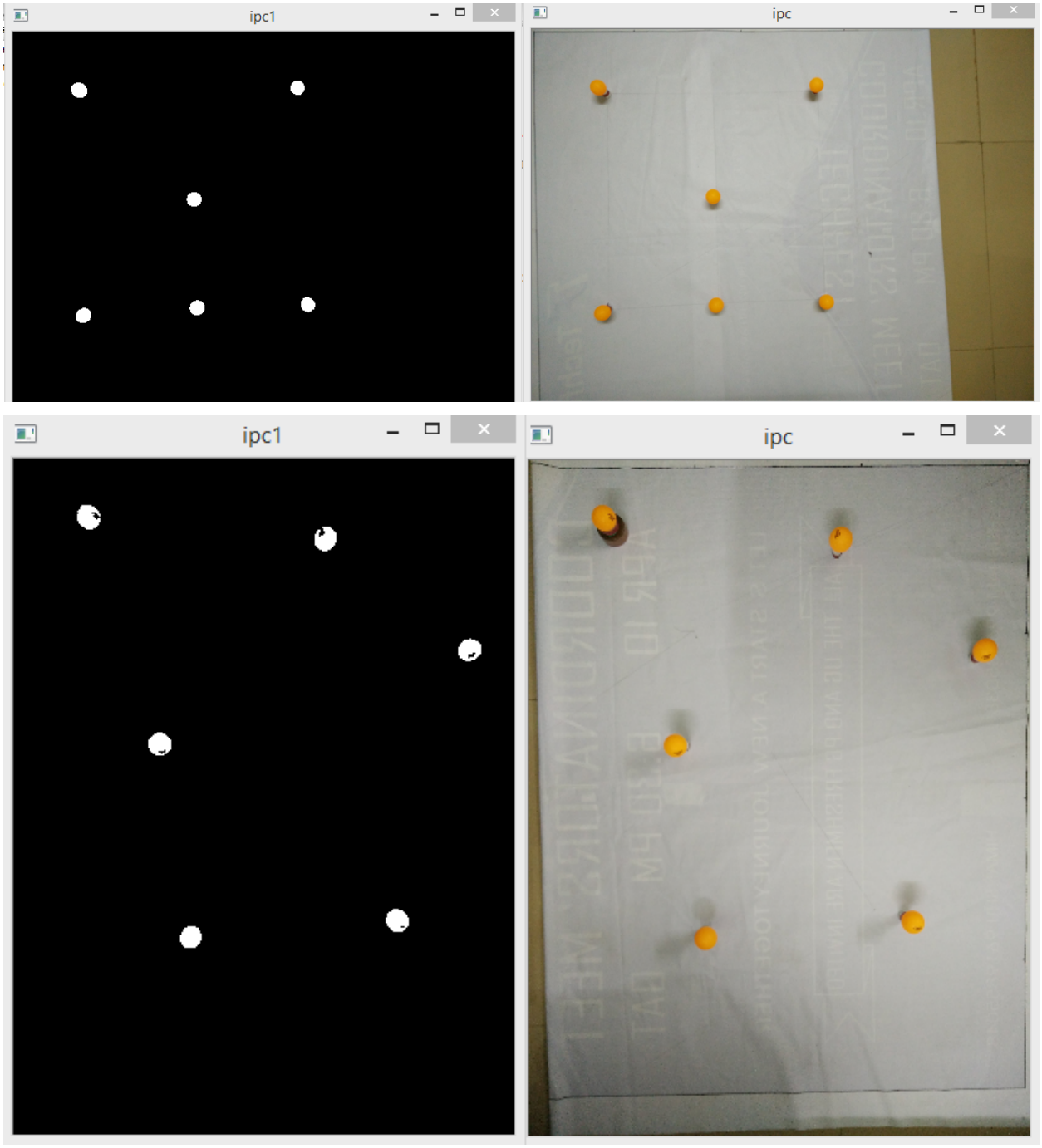


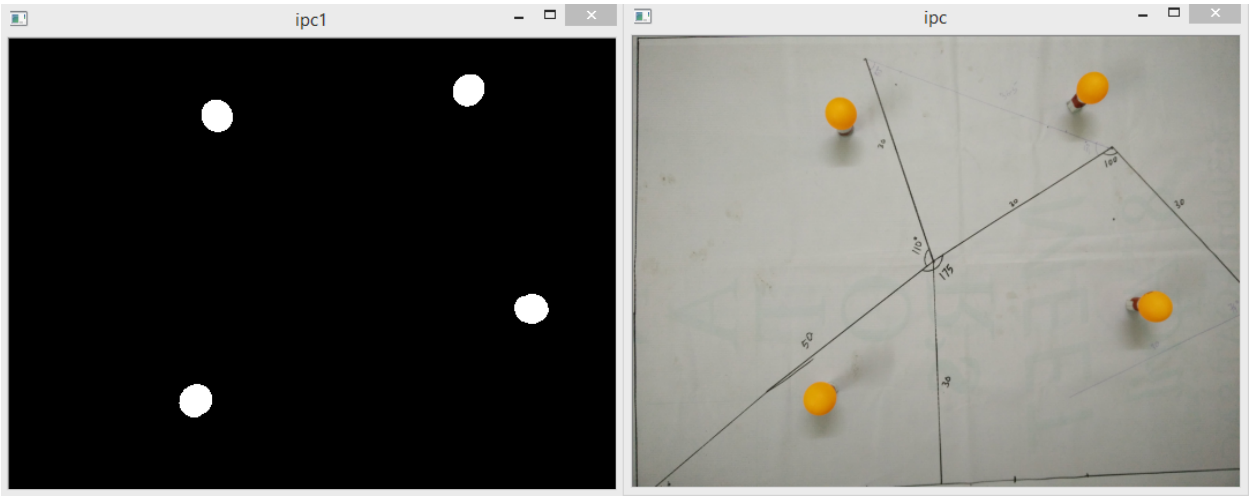
3.

Process:

The images are read and the balls are isolated in a binary image by applying proper thresholding conditions.

The images after this process -





The White pixels of the edges of the balls are stored and their average is found out to get the center of the balls.

Errors:

The errors generated during this processing were mostly due to hardware constraints.

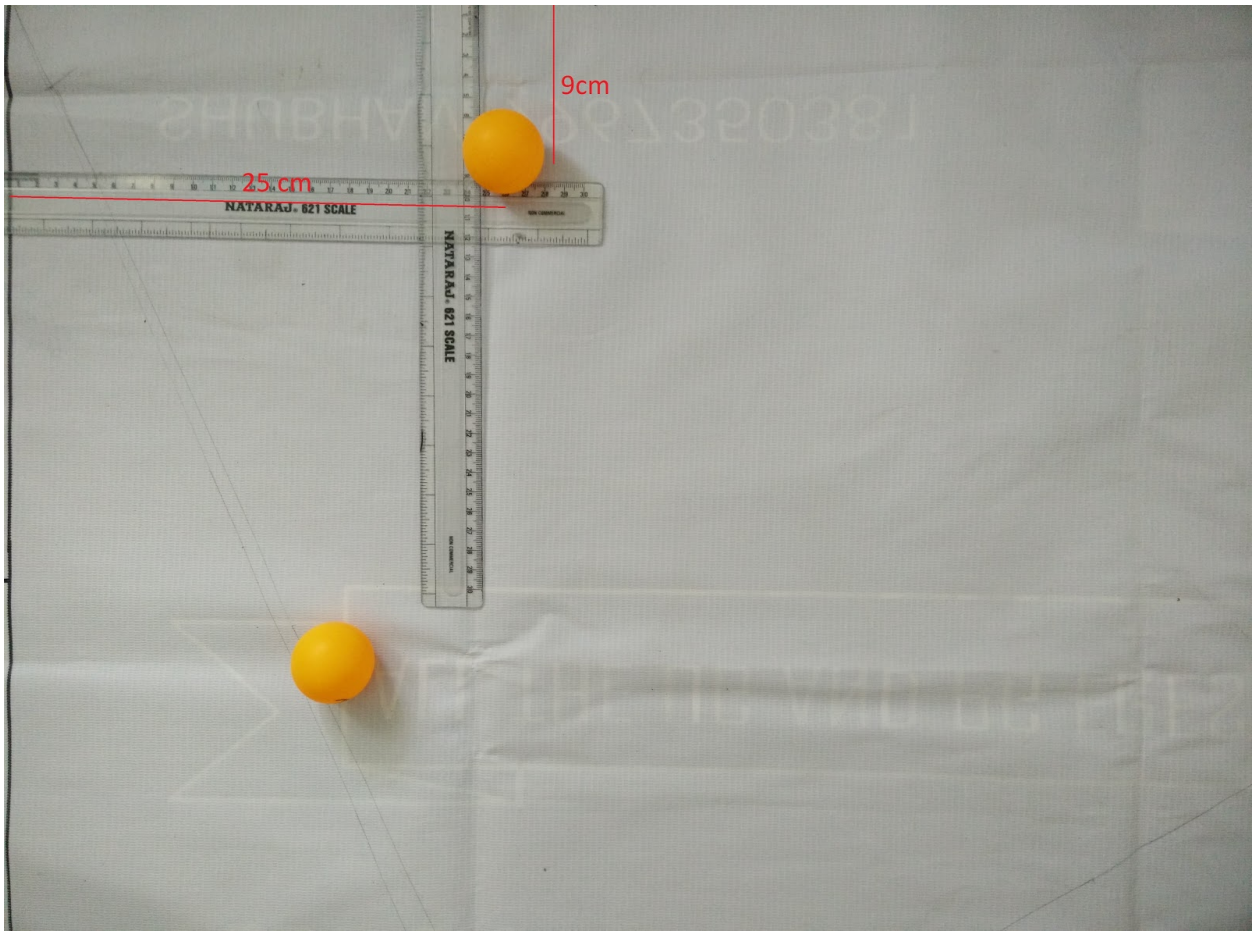
The image taken from the camera itself is not fully parallel to the view ,so the balls at the end of the image are shifted virtually in the image and the code would consider this position in the image as the position of the ball.

This problem , on an average, can bring about 2-3 cm error in the calculations.

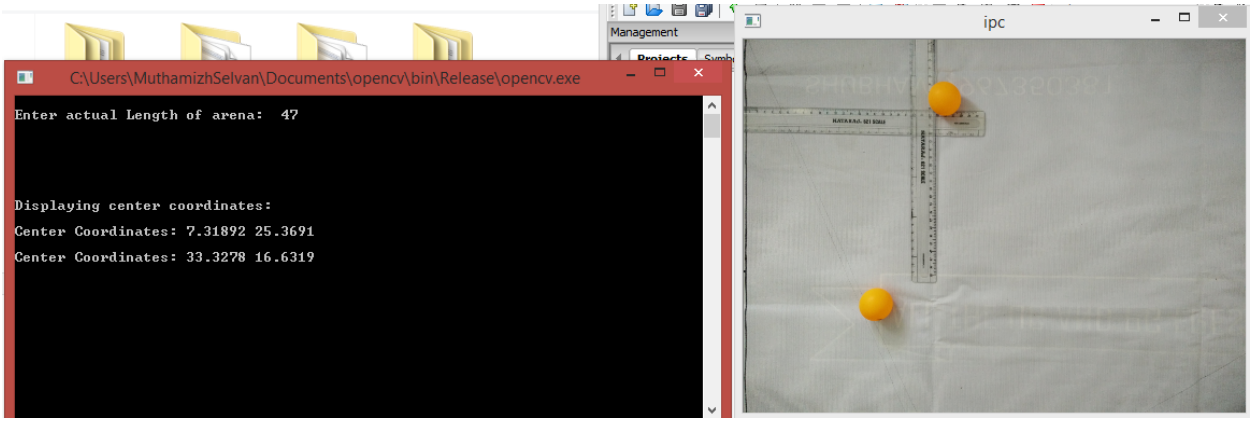
This error can be considerably reduced by taking the photos from highest possible point so that every object is as parallel to the camera as possible.But this produces problem in resolution of the balls in the image.

So the image must be taken from an optimum position to reduce the error.

The Image as taken from camera-



The code gives the coordinates as 25.3cm , 7.3cm.



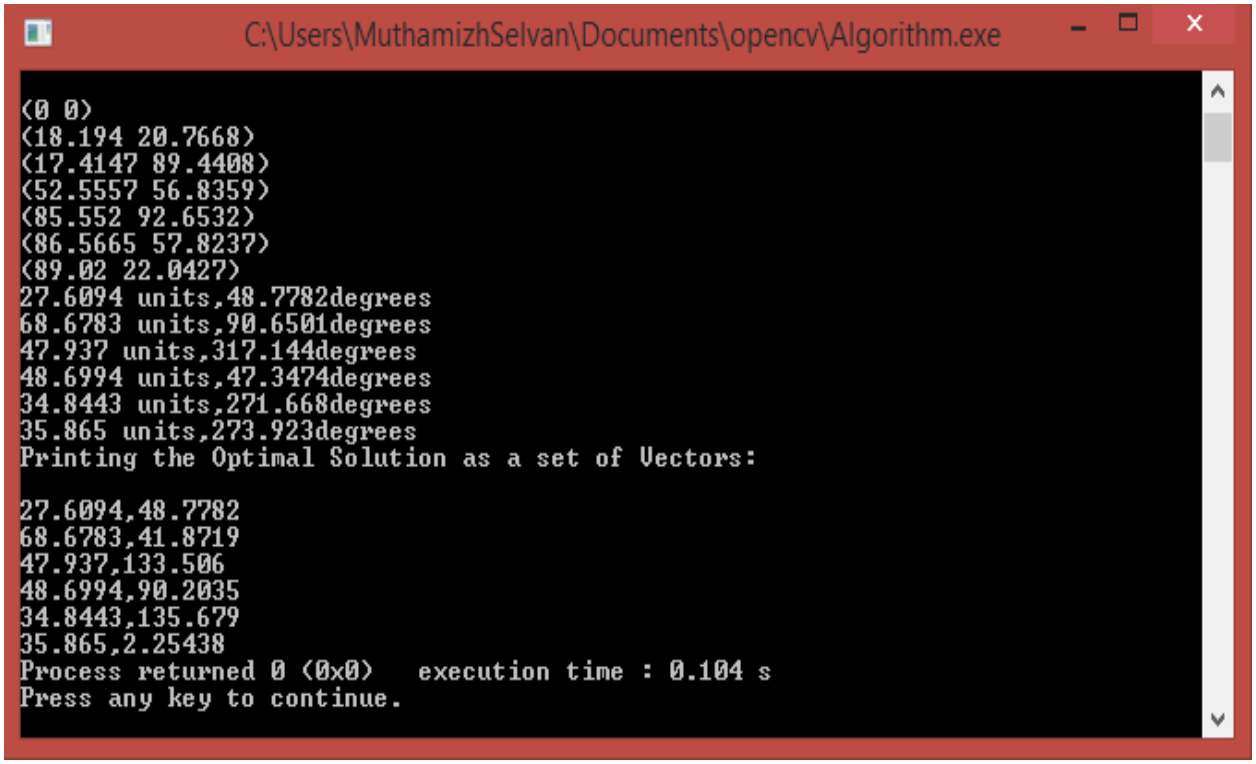
Module 2: Shortest Path Algorithm:

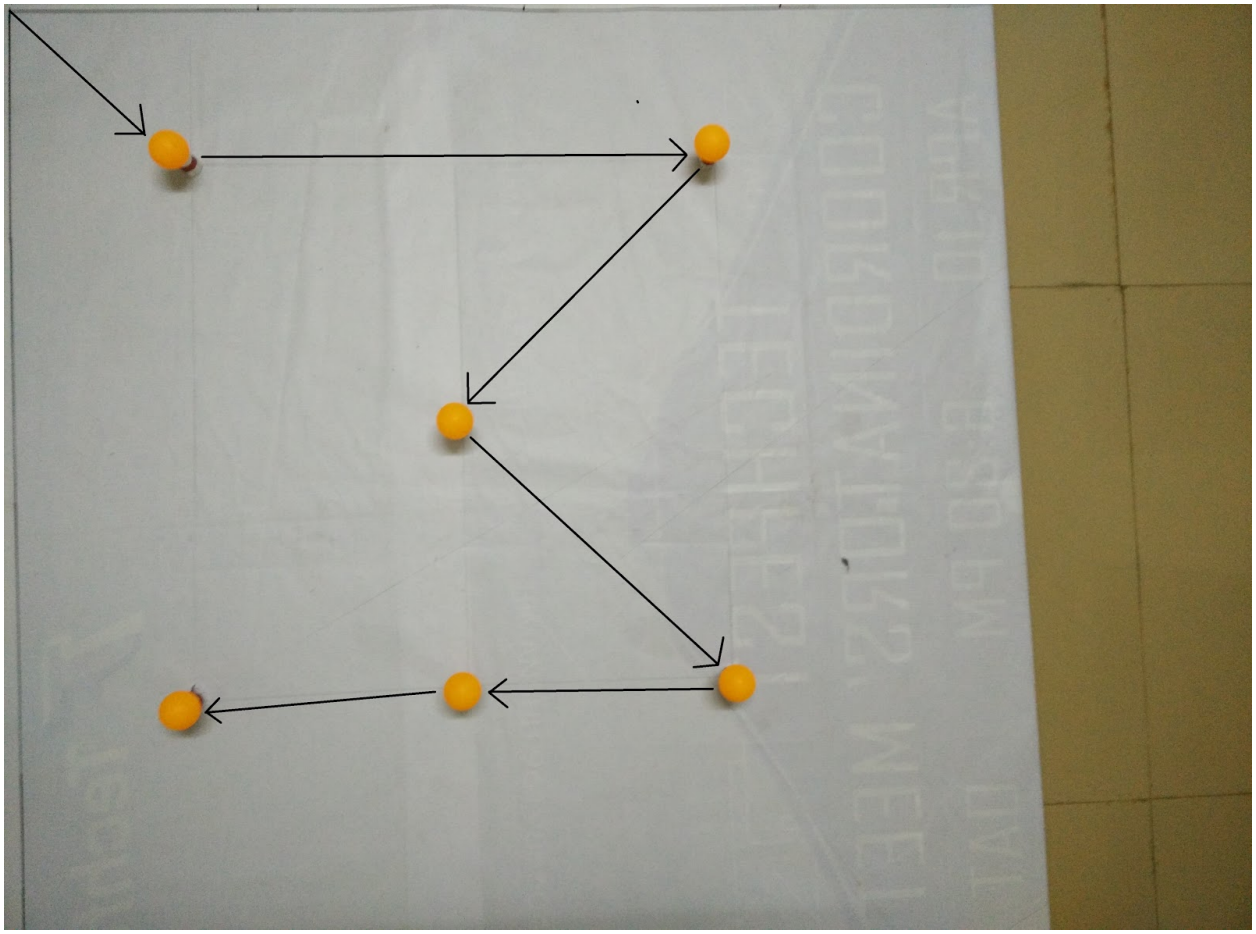
The shortest path algorithm was tested for random set of test points separately as well as in correspondence with the previously explained image processing code.

We had run the code for our test case and verified it by manually calculating the distances and the shortest path for the given set of points. The code works well as long as the given set of points does not have large number of balls/points.

The code take input from the previously created “Points.dat” file and creates a file “Instr.dat” containing sat of instructions in (‘distance to be moved’, ‘angle to be rotated’) format.

The following shows a snapshot of the console screen, tested for an arrangement of balls in the arena(shown in picture). The console screen prints the set of points in the optimal solution in (x,y) Cartesian coordinates, then in polar coordinates (radius,theta) and finally as the set of instruction to be stored in the file “Instr.dat.”





The above image (containing black arrows) shows the optimal path.

The code gives accurate processed data. The round off errors are within the scope of the resolution of the bot, and produces correct solution to the travel optimization problem, in general.

Errors:

The code works well in general, but the path might deviate from the optimal path when the number of balls becomes huge. This can be corrected by increasing the number of iterations in the code, by increasing the global variable 'NoOfCycles' (Details are explained in the documented code and documentation).

Also, the error in the coordinates of centres of the balls is carried from the image processing code. However, this error is in general negligible and within the resolution of the robot's position encoder.

Module 3: Bot Code

The bot code was tested 127 times.

The youtube link for videos of the motion of our bot are :

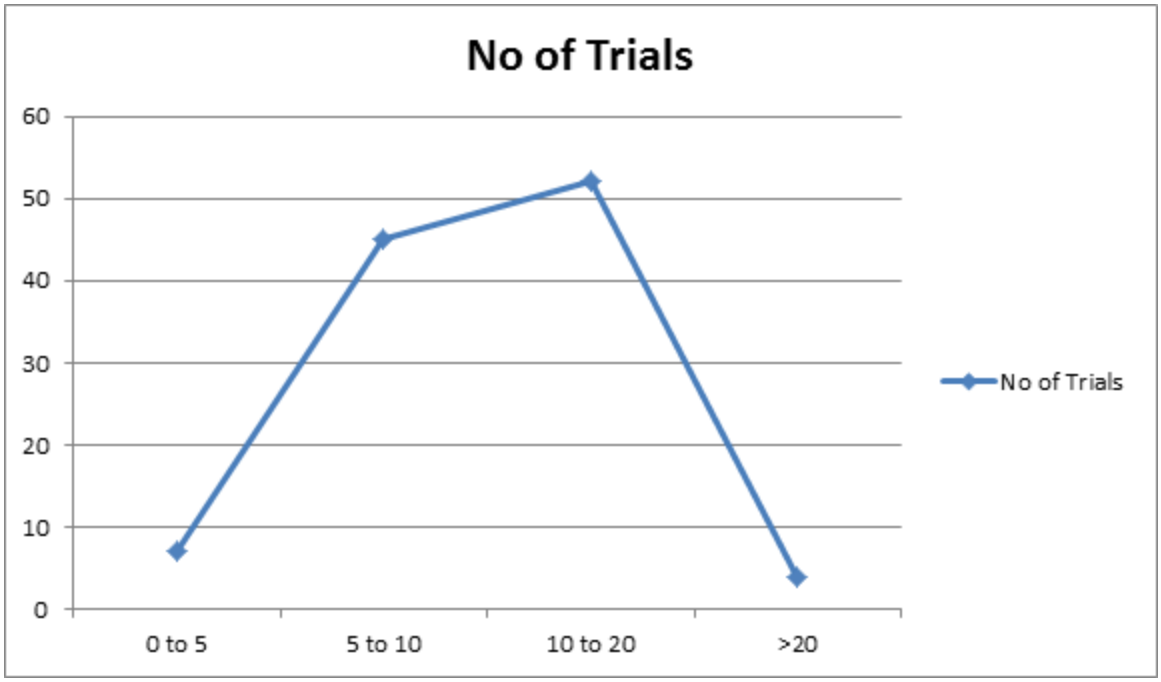
The approximate average error in the position of the bot in each case was found and recorded in the table below-

Error in position (cm)	No. of times
0-5	7
5-10	45
10-20	52
>20	4
Random errors in the bot or X-bee	19

The error in position is the algebraic sum of the deviations of the bot from the ideal position i.e., the coordinates of the center of the balls in one complete run. The error is always positive mostly for the reason that the bot tended to rotate more in a specific direction which we had taken to be positive thereby only giving positive errors.

The random errors include **random** failure of wireless communication and the bot failing to respond. Such errors do not have a definite cause and hence have no explanation or solution and are often eliminated in the next run.

This experimental data can be depicted by the graph shown below:



The bot uses its position encoder to measure distance. Both of its motions: translation and rotation are dependent on the position encoder values. As per the user manual of the FIREBIRD V bot, this gives an error of around 6-8 degrees while rotation. So, the error was mostly due to hardware limitations of the bot.

In order to reduce this error, we used sharp sensors. The bot was made to turn to the required angle, and then made to check the presence of the ball within ± 8 degrees around this angle. The position where the maxima is detected by the sharp sensor corresponds to the direction of the ball, and the bot was made to move along that. This was the version of the **PID Algorithm** we used for error reduction.

Module 4: X-Bee

We used the C++ header file <windows.h>, which is used to access , program and manipulate the windows ports, icons, etc. We accessed the COM port on which X-Bee was connected and sent data byte by byte through the serial port. In order to check whether the wireless transmission was happening properly, we echoed back the data to the laptop and verified whether it was the correct response.

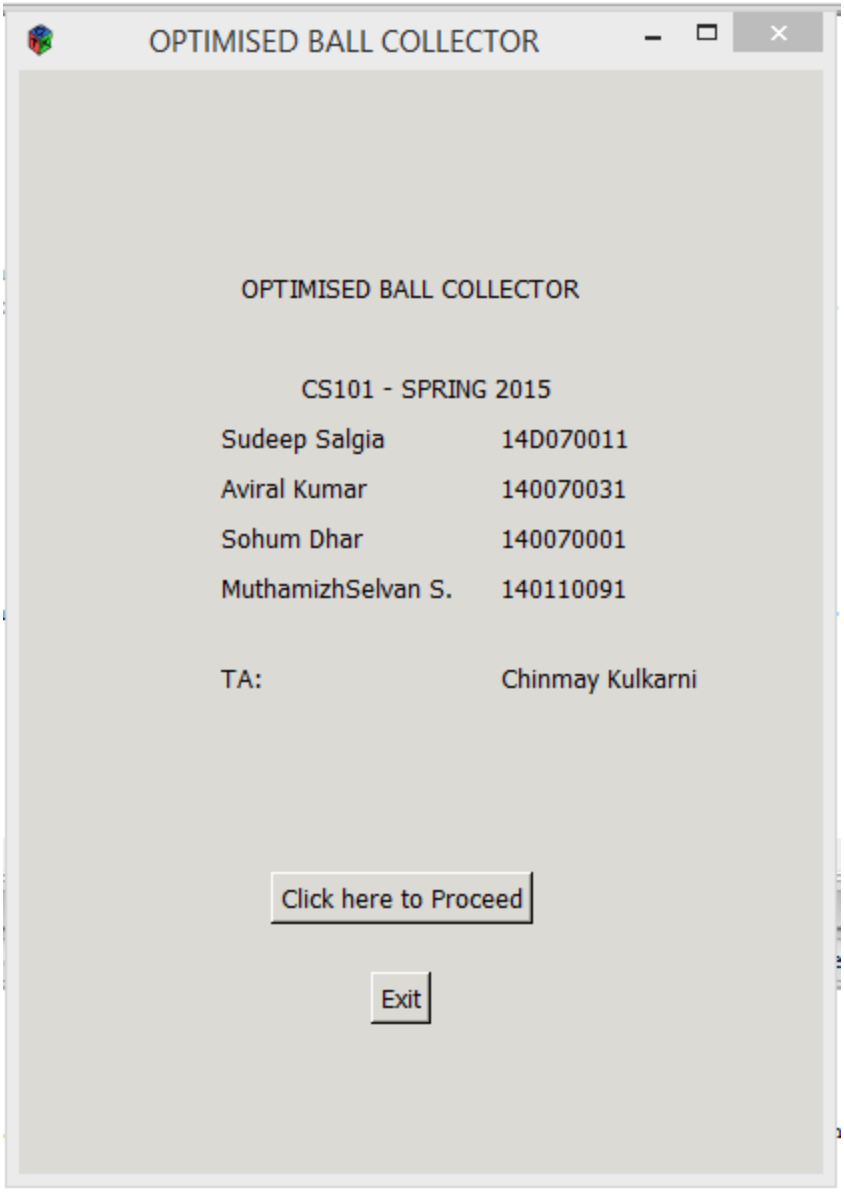
We defined our own encoding and decoding styles for optimal data transfer, i.e. we just used 2 bytes in order to transfer a 4 byte integer.

Errors:

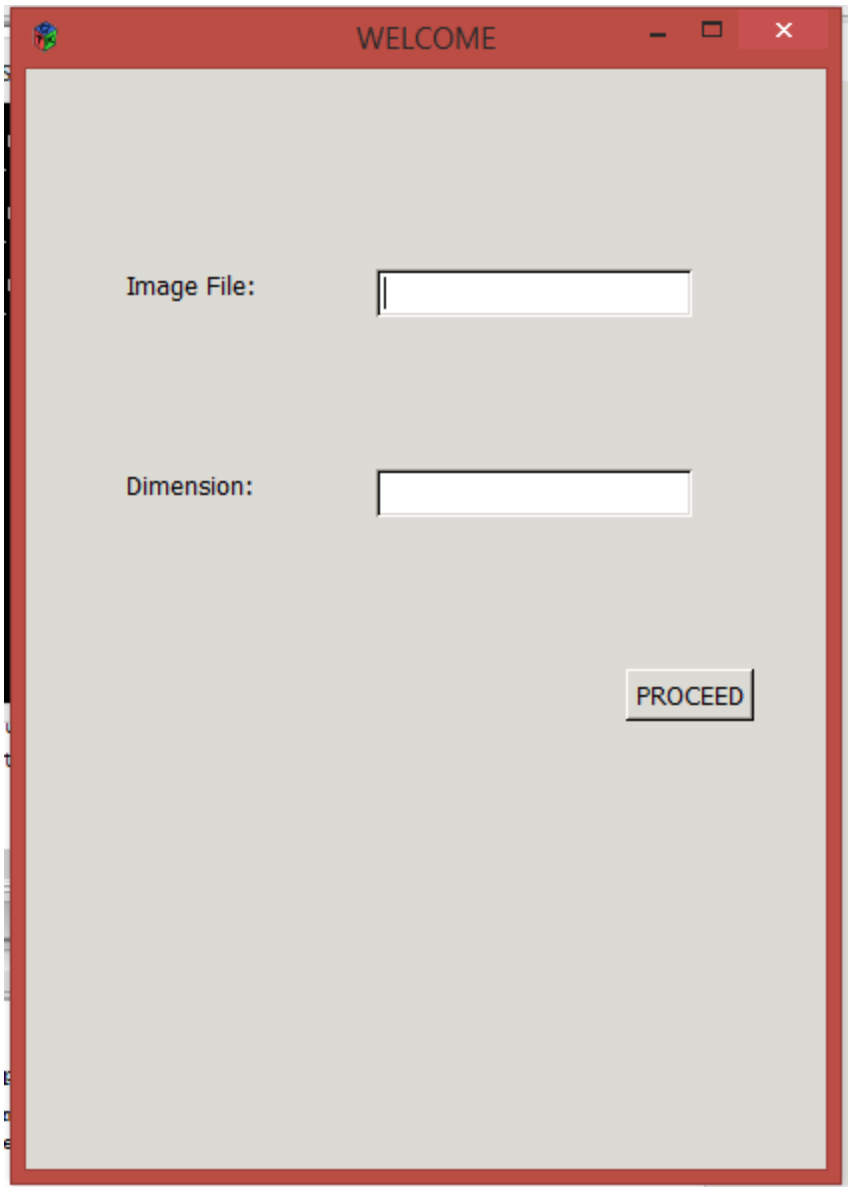
Sometimes there was a failure in the data transmission from the bot to the X-bee and vice versa. This was a random error and thus, cannot be explained.

Module 5: GUI Development

Initial Window which opens when the program is executed:



Window created after the user clicks “PROCEED” in the previous window



DISCUSSION OF THE SYSTEM

A. What have we completed ?

Our project has 4 modules -

1. Image Processing - A picture of the top view of the arena is taken from a camera and given to the program. The image is processed and the centres of the balls are found.

2. Shortest Path Algorithm - This part of our code finds the path (order in which the bot should traverse these points) such that the net path length is minimized.

3. Bot Code - This code, written in Embedded C++ on Atmel Studio 6.0 is fed into the bot. It makes the bot move on the path, implements PID and communicates with the X-Bee.

4. X-Bee Code - The X-bee code is used for data transmission from the C++ code to the bot. The (r, θ) values corresponding to the balls are read and transmitted one by one to the bot.

For more details of the code and the implementation please refer to the documentation.

B. What have we done beyond the SRS?

We have also made a GUI using GTK+2.24 in which the user can input the path of the image and the actual dimension of the arena instead of taking the input through the console while running the program.

C. What worked as per plan?

We had planned to initially implement the bot using the standard PID algorithm on the LF (Line follower) method. We were able to successfully finish that. After that we had planned to remove all the line follower and instead implement a “real-time PID” where the errors are determined and reduced with respect to the balls spread in the arena. We have been able to successfully implement this. Also, the image processing code and the shortest path algorithm have now been integrated with x-bee and running a single program makes the bot perform its actions.

The youtube link for our project video is:

https://youtu.be/on3euKLPWc0

FUTURE WORK

RE-USABILITY FEATURES:

1. The Whole Code of the project is written following the coding standards of C++ programming language.

2.The Xbee.h file contains xbee code which is completely reusable for every other Firebird V bot if properly configured.

3. The Shortest Path Code can be reused in any likewise problems.

4. The Image Processing code can be reused in isolating different objects other than balls and also could be used for different colours.

SOME PRACTICAL APPLICATIONS OF THE PROJECT:

Tennis- The algorithm that we used can be implemented as a ball collector in a tennis court. Thus using a specialized bot for the purpose and image being fed from a camera at a reasonable height and our algorithm, it can be achieved to collect balls spread across the arena.

Garbage Collection- On a relatively larger scale, our code can be used to design a mechanized energy-efficient system for garbage collection from several points along with some necessary equipment in a specific area .

As far as the small scale implementation is concerned, our project does not have the capability to collect the balls by itself (due to time constraints) .So in future proper grippers could be added to the bot.

CONCLUSIONS

We had a very great learning experience working with OpenCV and operating Firebird V. The project gave us a good hands on experience with embedded systems. The development of algorithm for finding the shortest path helped us learn various aspects in this dimension.

In general this idea has various application in real world, both at present and in future. The algorithm addresses a variety of problem and hence has great reusability features.

Overall, it was a very enriching experience for us.

REFERENCES

1. OpenCV Documentation :

   http://docs.opencv.org/

2. ATMega2560 Datasheet Published by Atmel , Link:

   http://www.atmel.com/images/doc2549.pdf

3. e-Yantra projects :

   http://www.e-yantra.org

4. Travelling Salesman Problem:
   http://en.wikipedia.org/wiki/Travelling_salesman_problem

   http://mathworld.wolfram.com/TravelingSalesmanProblem.html

5. Gtk+ Documentation:

   http://www.gtk.org/documentation.php

Link to Github repository -
https://github.com/sudeepsalgia/14D070011_418_Optimised-Ball-Collector

Link to VIdeo of working Project -
https://www.youtube.com/watch?v=on3euKLPWc0

Link to Video tutorial - https://www.youtube.com/watch?v=XAYdx7oH660