A MINOR PROJECT REPORT

on

3D VISION BASED MOBILE ROBOT GN&C

Submitted in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

MECHATRONICS

by

R HARISH KUMAR (RA1511018010120)
S GOKULL (RA1511018010211)

Under the guidance of

DR. R. SENTHILNATHAN
Associate Professor
Department of Mechatronics

FACULTY OF ENGINEERING AND TECHNOLOGY
SRM Nagar, Kattankulathur- 603 203

April 2018 – 6th Semester

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this project report titled **"3D VISION BASED MOBILE ROBOT GN&C"** is the bonafide work of **R HARISH KUMAR** and **S GOKULL**, who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr. R. Senthilnathan

PROJECT GUIDE

Associate Professor

Department of Mechatronics Engineering

SIGNATURE

Dr. G. Murali

HEAD OF THE DEPARTMENT

Department of Mechatronics Engineering

Signature of the Internal Examiner

Signature of the External Examiner

TABLE OF CONTENTS

CHAPTER NO.	TITLE	Pg. No.
	ABSTRACT	iv
	ACKNOWLEDGEMENT	v
	LIST OF TABLES	vi
	LIST OF FIGURES	vii
1	**INTRODUCTION**	1
	1.1 Motivation for the Project	1
	1.2 The Initial Idea	1
	1.3 Applications	1
	1.4 Organization of the Report	2
2	**LITERATURE SURVEY**	3
3	**DESIGN AND SPECIFICATIONS**	4
	3.1 System Architecture	4
	3.2 NI LabVIEW Robotics Starter Kit DaNI 1.0	5
	3.3 Arduino UNO	6
4	**3D VISION**	8
	4.1 Image Acquisition	8
	4.2 Image Processing and Analysis	8
	4.3 Occupancy Grid Updation	11
5	**PATH PLANNING**	14
	5.1 Introduction	14
	5.2 Algorithm	14
	5.3 Examples and Results	15
6	**ODOMETRY AND CONTROL**	16

	6.1 Localization of Mobile Robot	16
	6.2 Forward Kinematics Model	16
	6.3 Control of Mobile Robot	18
	6.4 Implementation of Odometry and Control in LabVIEW	19
7	**CONCLUSION AND FUTURE PLAN**	21
	REFERENCES	22
	APPENDIX	
	(a) Code	23

ABSTRACT

Self-driving cars are the future of computer vision and artificial intelligence. They are the most advanced application of this technology. Autonomous mobile robots have their applications in many industries as well as homes, offices, hospitals, etc. The motivation behind undertaking this project is to study, understand and implement autonomous guidance, navigation and control of a mobile robot using 3D Vision. During the course of this project, the work was split into three domains namely, 3D Vision, Path planning and Odometry & Control, and they were integrated later to test the mobile robot. The mobile robot DaNI 1.0 provided by National Instruments was used for this project. Image Processing & Analysis, Path Planning and the main Motion Control loop were performed on a central loop on MATLAB run on a laptop and motion commands were given to sbRIO – 9631 of DaNI 1.0 via Arduino UNO. The Odometry & Control loop was run on LabVIEW and the POSE of the robot are given back as feedback to the main processing loop on MATLAB via Arduino UNO. The autonomous guidance, navigation & control of the mobile robot is implemented in this manner.

ACKNOWLEDGEMENT

We would like to thank the management of SRM Institute of Science and Technology, for providing us with the opportunity to pursue our minor project work in this esteemed institution.

We would wish to express our gratitude to **Professor G. Murali**, HoD (Mechatronics) for encouraging the students of our department to take up minor projects as a part of their curriculum.

We would like to express our heartfelt, sincere thanks to our guide, **Associate Professor, Dr. R. Senthilnathan** for his continuous support and his exceptional ideas and suggestions that helped our project to progress at every stage.

We also express our gratitude to all the faculty members of the Department of Mechatronics, who have been a part of our learning and growing, for mentoring us to be skilled engineers.

We would also like to express our deepest appreciation and gratitude to our Review Committee Members, **Dr. T. Muthuramalingam**, and **Mr. Sanjay Kumar Kar** for their valuable comments and the questions they asked during the reviews which pushed us to work harder on the project.

Finally, we wish to express how grateful we are to our family and friends for their unfailing support and strong encouragement throughout the process of the project.

LIST OF TABLES

TABLE NO.	TITLE	Pg. No.
3.1	Specifications of DaNI 1.0	5
3.2	Laptop and Software Specifications	6
3.3	Arduino UNO Specifications	6
3.4	Kinect Sensor Specifications	7
4.1	Image Acquisition Parameters	8
4.2	Occupancy Grid Parameters	12

LIST OF FIGURES

FIGURE NO.	TITLE	Pg. No.
1.1	Project Work Flow	2
3.1	System Architecture	4
3.2	Analog Input / Output Pinout of sbRIO-9631	5
3.3	Pinout of Arduino UNO	7
4.1	Acquired Depth Images	8
4.2	Range Limited Binary Images	9
4.3	Segmented Images After Floor Removal	11
4.4	(a) Representation of α on Image plane (b) Updation of Obstacles on the Occupancy Grid	12
4.5	Mapped Obstacles on the Occupancy Grid	13
5.1	Path Planning Algorithm	14
5.2	Path Planning Examples	15
6.1	Global Reference Frame and Robot Local Reference Frame	16
6.2	Control of Differential Drive Mobile Robot	18
6.3	Implementation of Odometry and Control Loop in LabVIEW	20

1. INTRODUCTION

1.1 Motivation for the Project

3D-vision based mobile robot navigation has always been a challenge. This is due to the complexity of integrating massive amount of information gathered by robot vision while it undergoes a motion. Visual perception of an environment, is also an important capability for mobile robots, and lots of efforts are being put by mobile robot research community. This is due to its consideration as a crucial part of mobile robot design. The knowledge gain and applications in the field of 3D vision are so vast and intriguing, which is what motivated us to undertake this project.

1.2 The Initial Idea

Our initial objective was to implement obstacle avoidance control for a mobile robot based on 3D Vision and plan a path accordingly to reach the destination. We planned to avoid the obstacles present on the planned path and change its path accordingly. The idea was to make the algorithm robust such that it would work for different types of floors and various obstacles. Generation of a smooth curved path between points and changing the direction during motion was also planned initially.

1.3 Applications

3D vision is highly important in fields such as robotics, to extract information about the relative position of 3D objects in the vicinity of autonomous systems. Other applications for robotics include object recognition, where depth information allows for the system to separate occluding image components, such as one chair in front of another, which the robot may otherwise not be able to distinguish as a separate object by any other criteria. 3D vision incorporated mobile robots are extensively used in hazardous environments and also for military purposes.

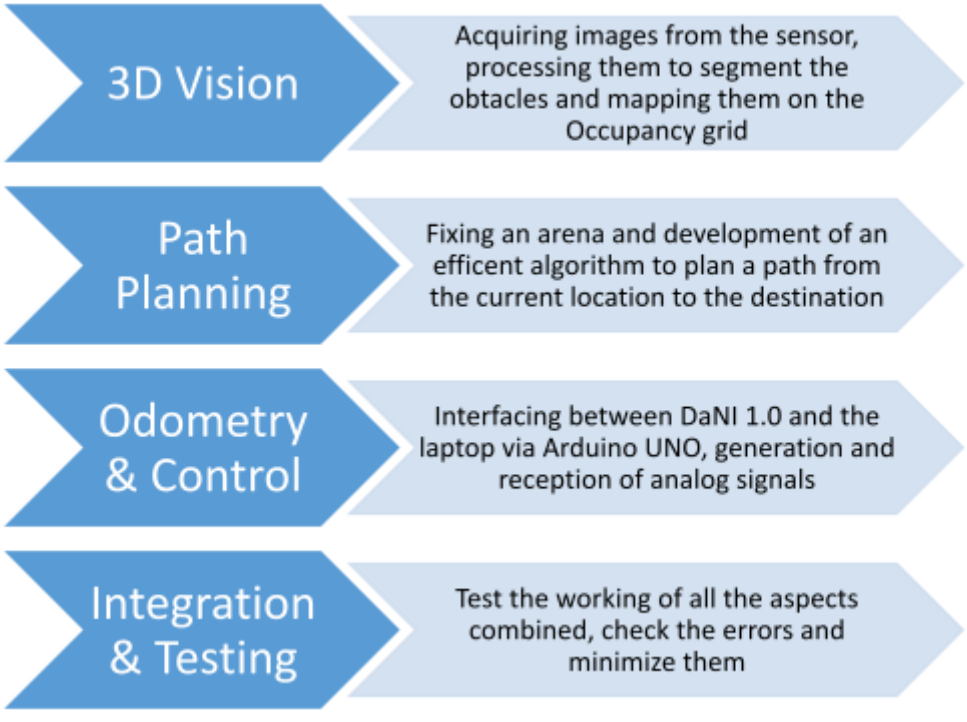


Figure 1.1 Project Work Flow

1.4 Organisation of the Report

The results of this project have been compiled into a report as detailed below:

Chapter 1 details about the motivation behind the project, initial ideas, applications and the work flow of the project.

Chapter 2 details the literature review of the mobile robots that have used 3D vision for their guidance.

Chapter 3 elaborates on the 3D Vision aspect of our algorithm and its implementation on MATLAB.

Chapter 4 elaborates on the Path planning algorithm used to reach the destination implemented on MATLAB.

Chapter 5 elaborates on the Odometry & Control aspect of the algorithm which includes the Odometry & Control loop's implementation on LabVIEW.

Chapter 6 provides the conclusions and the future scope of the project.

2. LITERATURE SURVEY

Rasoul Mojtahedzadeh in his Master thesis titled "*Robot Obstacle Avoidance using the Kinect*" modelled the environment based on the point cloud extracted from the depth image data and implemented obstacle avoidance method using the straight-line segments and circle arcs to find a collision free path from the current position of the robot to the target pose subject to the shortest possible path considering a safeguard.

Gu, JJ and Cao, in the paper titled "*Path planning for mobile robot in a 2.5-dimensional grid-based map*", Industrial Robot-An International Journal 38, 2011, discussed the state-of-the-art in obstacle avoidance and path planning for industrial robots that is practical on the current generation of computer hardware. They have also described practical vehicle planners and summarized that obstacle avoidance and path planning are techniques with differing goals.

3. DESIGN AND SPECIFICATIONS

3.1 System Architecture

The main components of the design architecture are-

- DaNI 1.0 mobile robot
- Microsoft Kinect for Xbox 360
- Optical wheel encoders
- Arduino UNO
- Laptop

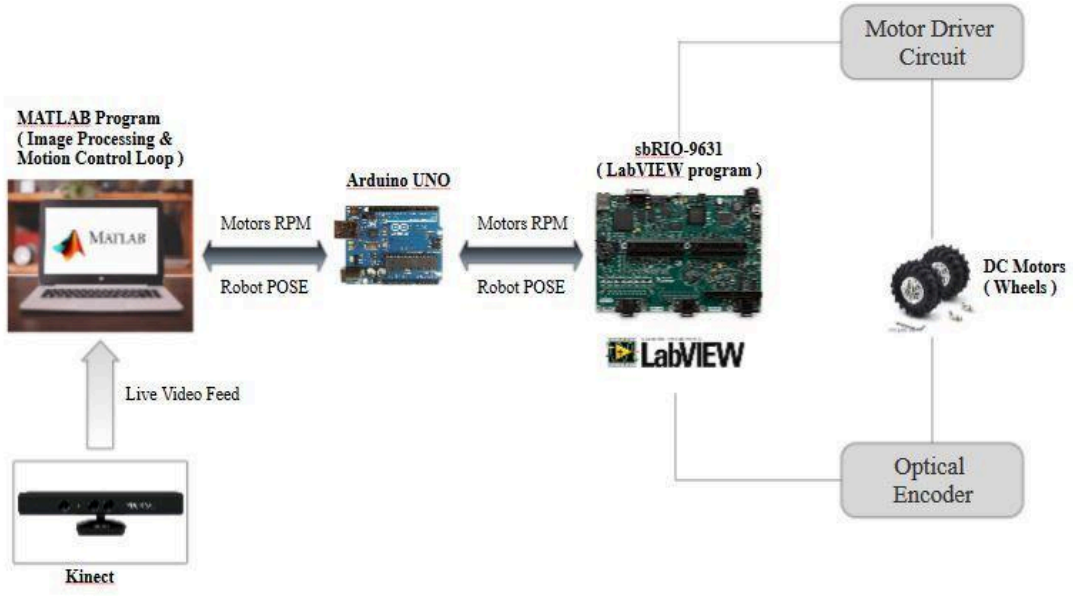


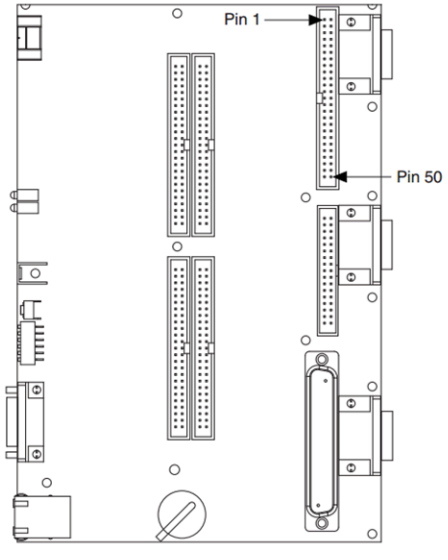
Figure 3.1 System Architecture

3.2 NI LabVIEW Robotics Starter Kit DaNI 1.0

For executing our work processes and algorithms we are using the NI LabVIEW Dani 1.0 mobile robot.

Table 3.1 Specifications of Dani 1.0

DC Motors	
Supply voltage	12V
Torque	300 oz-in
RPM	152
Encoders	
Supply voltage	5V
Pulses per revolution	400 PPR
NI sbRIO-9631	
Network interface	10BaseT and 100BaseTX Ethernet
Communication rates	10 Mbps, 100 Mbps, auto-negotiated
Processor Speed	266 MHz
Non-volatile memory	128 MB
System memory	64 MB
Analog input ranges	±10 V, ±5 V, ±1 V, ±0.2 V
Analog output range	±10 V



AI GND	1	2	AI0
AI8	3	4	AI9
AI1	5	6	AI GND
AI2	7	8	AI10
AI11	9	10	AI3
AI GND	11	12	AI4
AI12	13	14	AI13
AI5	15	16	AI GND
AI6	17	18	AI14
AI15	19	20	AI7
AI GND	21	22	AI16
AI24	23	24	AI25
AI17	25	26	AI GND
AI18	27	28	AI26
AI27	29	30	AI19
AI GND	31	32	AI20
AI28	33	34	AI29
AI21	35	36	AI GND
AI22	37	38	AI30
AI31	39	40	AI23
AI SENSE	41	42	AO GND
AO3	43	44	AO GND
AO2	45	46	AO GND
AO1	47	48	AO GND
AO0	49	50	AI GND

Figure 3.2 Analog Input / Output Pinout of sbRIO-9631

Table 3.2 Laptop and Software Specifications

Laptop	
Company	DELL
Processor	Intel® Core™ i7-5500U
Processor speed	2.4Ghz (turbo boost up to 3Ghz)
Installed RAM	8GB
Software	
MATLAB	2016a
NI LabVIEW	2014

3.3 Arduino UNO

Arduino UNO works as an interface between laptop and the mobile robot. Communication between Arduino and mobile robot is done through generation of analog output and reading it.

Table 3.3 Arduino UNO Specifications

Microcontroller	ATmega328P
Input Voltage	7-12V
Digital I/O Pins	14
PWM Digital I/O Pins	6
Analog Input Pins	6
Clock Speed	16Mhz
Analog output range	0-5V
Analog input range	0-5V
Flash Memory	32KB

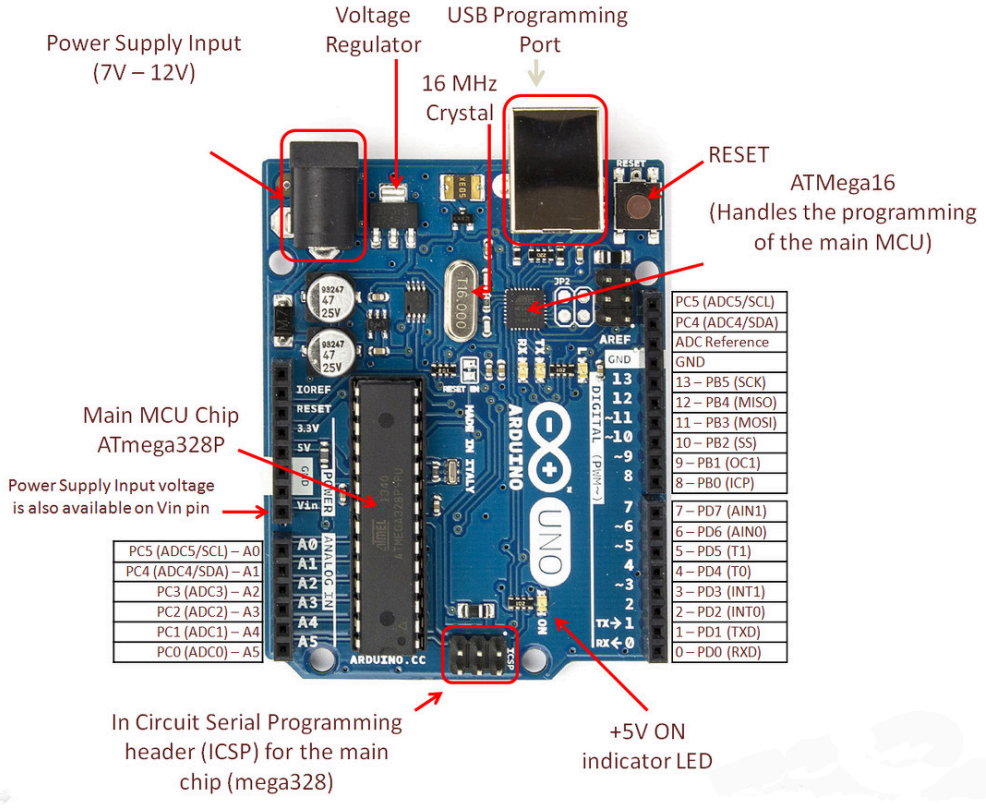


Figure 3.3 Pinout of Arduino UNO

Table 3.4 Kinect Sensor Specifications

Field of View	
Horizontal field of view	57 degrees
Vertical field of view	43 degrees
Depth sensor range	0.8m – 4.0m
Data Streams	
Data path	USB 3.0
640x480 32-bit colour@ 30 frames/sec	
320x240 16-bit depth @ 30 frames/sec	

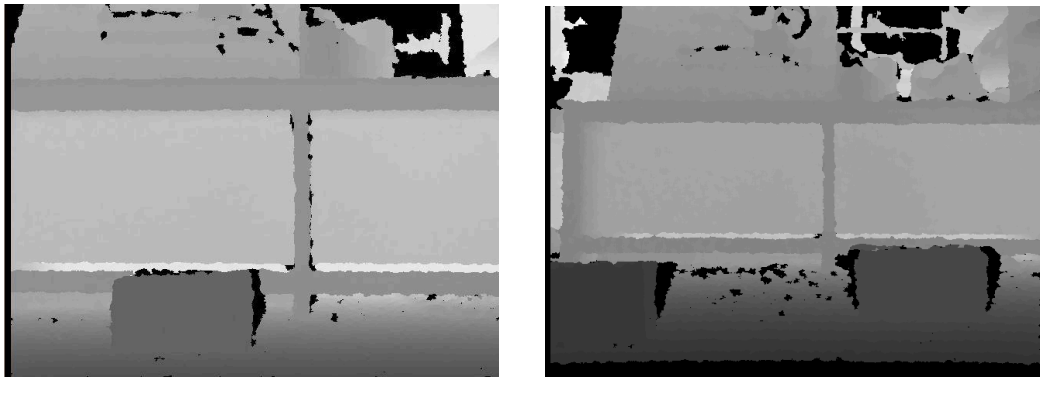
4. 3D Vision

4.1 Image Acquisition

The live video feed is acquired using Microsoft Kinect Xbox 360 which is connected the laptop. The video is acquired as depth video since the area of concern is just limited to ranges of obstacles and colour information is not required for this purpose. As a result, the image does not get affected due to factors such as illumination, etc. However, it can only work in an indoor environment as sunlight contains IR rays and affects the functioning of Kinect. The video is stored as a video object on MATLAB. The resolution of the video is chosen as '320x240' owing to the time and space constraints for processing on the laptop. The video object once triggered, starts acquiring frames until it is stopped by a break condition, i.e. when the robot reaches its destination or if it realizes there is no viable path to the destination. The video object is deleted after the end to prevent memory shortage problem.

Table 4.1 Image Acquisition Parameters

Image Acquisition Parameters	
Modality	Depth video
Resolution	320x240



(a) **(b)**

Figure 4.1 Acquired Depth Image For (a) Single Obstacle, and (b) Multiple Obstacles

4.2 Image Processing and Analysis

A preliminary step in image processing is to flip the acquired frame about the central column of the image to implement lateral inversion since the original image has its sides reversed.

The first step in image processing is to limit the amount of data we process so as to manage with the time and space constraints. This is performed by limiting the range of depth values from the acquired depth image of the scene. The limited range image is converted to binary for further processing.

(a) **(b)**

Figure 4.2 Range Limited Binary Image For (a) Single Obstacle, and (b) Multiple Obstacles

In the above figure, the resulting image is shown after limiting the range from 80 to 120 centimetres and converting it to a binary image where the background is given black and the range of interest is given white.

The floor of the arena is visible in every frame along with the real obstacles. The floor is removed from the image using the method of variances. The pseudo code for the method of variances is given as follows.

Method of variances for columns:

for all_columns

if column_variance > col_threshold

restore same pixel values

else

pixel values in column = 0

end

end

Measured optimal value of column threshold: 1.4e+05

Method of variances for rows:

for all_rows

if row_variance > row_threshold

restore same pixel values

else

pixel values in row = 0

end

end

Measured optimal value of row threshold: 1.7e+05

(a) **(b)**

Figure 4.3 Segmented Image After Floor Removal for (a) Single Obstacle, and (b) Multiple Obstacles

After the removal of the floor, we now have a clear picture of just the obstacles present in the scene. The total number of obstacles is obtained by fitting a bounding box around each obstacle. The size of the bounding box vector gives the number of obstacles.

4.3 Occupancy Grid Updation

Occupancy grids are used to represent a robot workspace as a discrete grid. Information about the environment can be collected from sensors in real time or be loaded from prior knowledge. Laser range finders, bump sensors, cameras, and depth sensors are commonly used to find obstacles in the robot's environment.

2-D occupancy grids have two representations:

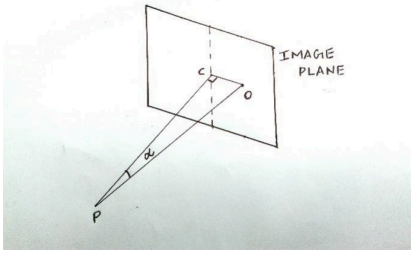
- Binary occupancy grid
- Probability occupancy grid

Binary occupancy grid is used to represent the arena. The entire arena is divided into cells and each cell can have a value of 0 or 1. The obstacles are given a value of 1 (black) and the free space cells are given a value of 0 (white).

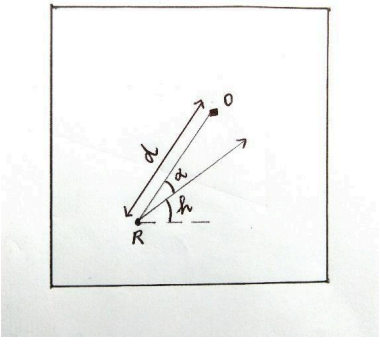
Table 4.2 Occupancy Grid Parameters

Occupancy Grid Parameters	
Length	4 (m)
Breadth	4 (m)
Resolution	40 (cells/m)

After the identification of obstacles, each obstacle is mapped to the occupancy grid based on its position on the image. This mapping is done based on the angle α from the image, where α is the angle between the line joining the obstacle & camera and the center point of image & camera.



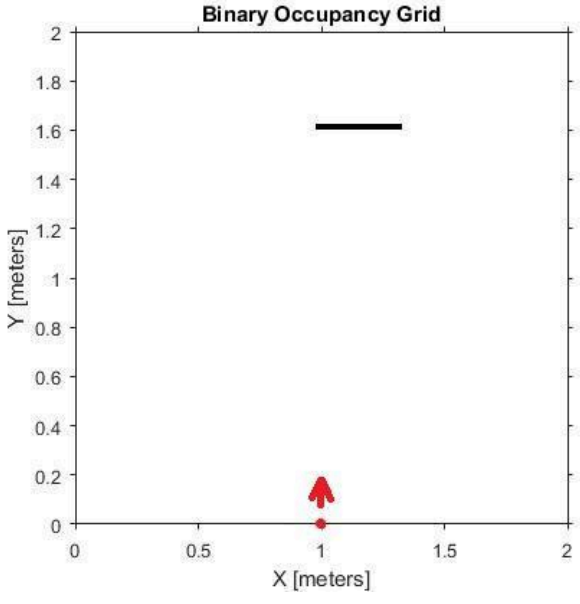
(a)



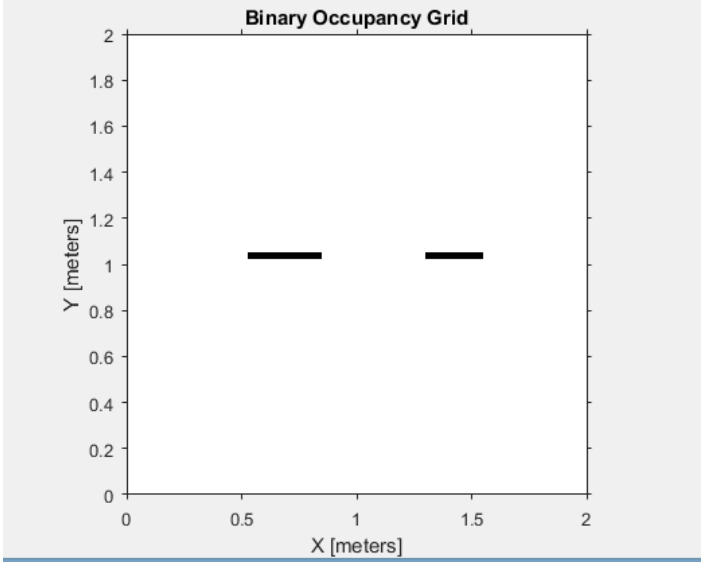
(b)

Figure 4.4 (a) Representation of α on Image Plane (b) Updation of Obstacles on the Occupancy Grid using α, where h – Heading Angle, d – Obstacle Distance, R – Robot Position, O – Obstacle Position

This mapping is done for every sample of every obstacle so that we get the accurate positions of all the points on the obstacles even if the obstacle is not of uniform depth.



(a)



(b)

Figure 4.5 Mapped Obstacles on the Occupancy Grid: (a) Single Obstacle (b) Multiple Obstacles (Note: The size of the Occupancy Grid chosen here is for test purposes. The final size is 4m x 4m.)

After the obstacles have been updates successfully, the flow of control moves on to the path planning and the control sections and the same process is repeated for all the frames until the destination is reached.

5. PATH PLANNING

5.1 Introduction

Path planning refers to the determination of path in configuration space between an initial configuration of the robot and a final configuration such that the robot does not collide with any obstacles in the environment. Path planning can be done either globally or locally. Global path planning is useful when the map of the arena is known a priori and the path is planned based on the pre-loaded map. Local path planning is essential when it is an unknown environment and the robot is required to identify the obstacles and plan its path accordingly. Local path planning is implemented in this project based on the target coordinates acquired from the user.

5.2. Algorithm

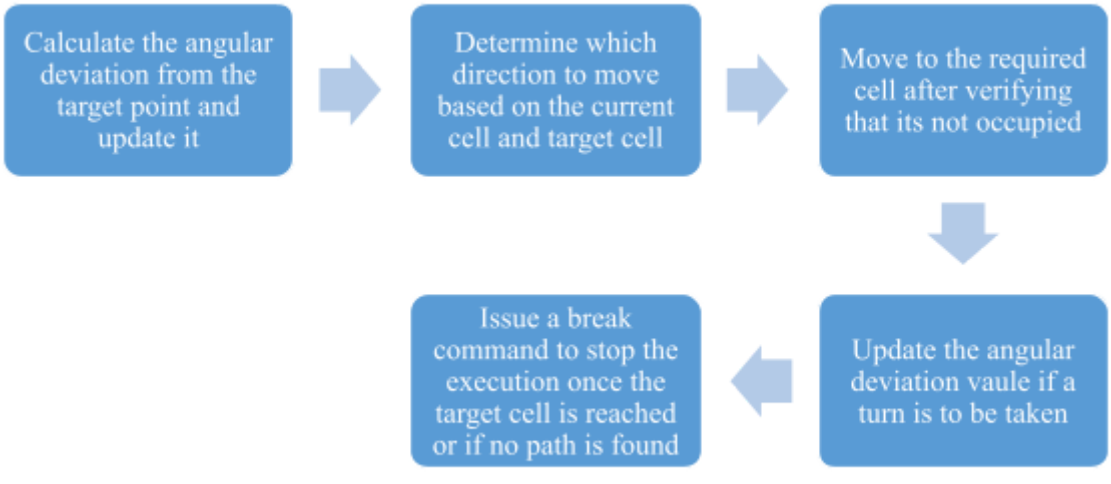


Figure 5.1 Path Planning Algorithm

The path planning starts immediately after the updating of occupancy grid. Based on the latest occupancy grid values, the path is planned using the above algorithm. For every frame, only one cell is updated in the path planning algorithm. This prevents global planning by mistake and is efficient to make changes with every frame. The calculated position and the angular deviation is fed to the Odometry & control loop which acts on this information to send corresponding signals to the DC motors (wheels).

5.3. Examples and Results

Path planning is done on MATLAB and the resulting path can be visualized on the occupancy grid itself. Some results are presented below.

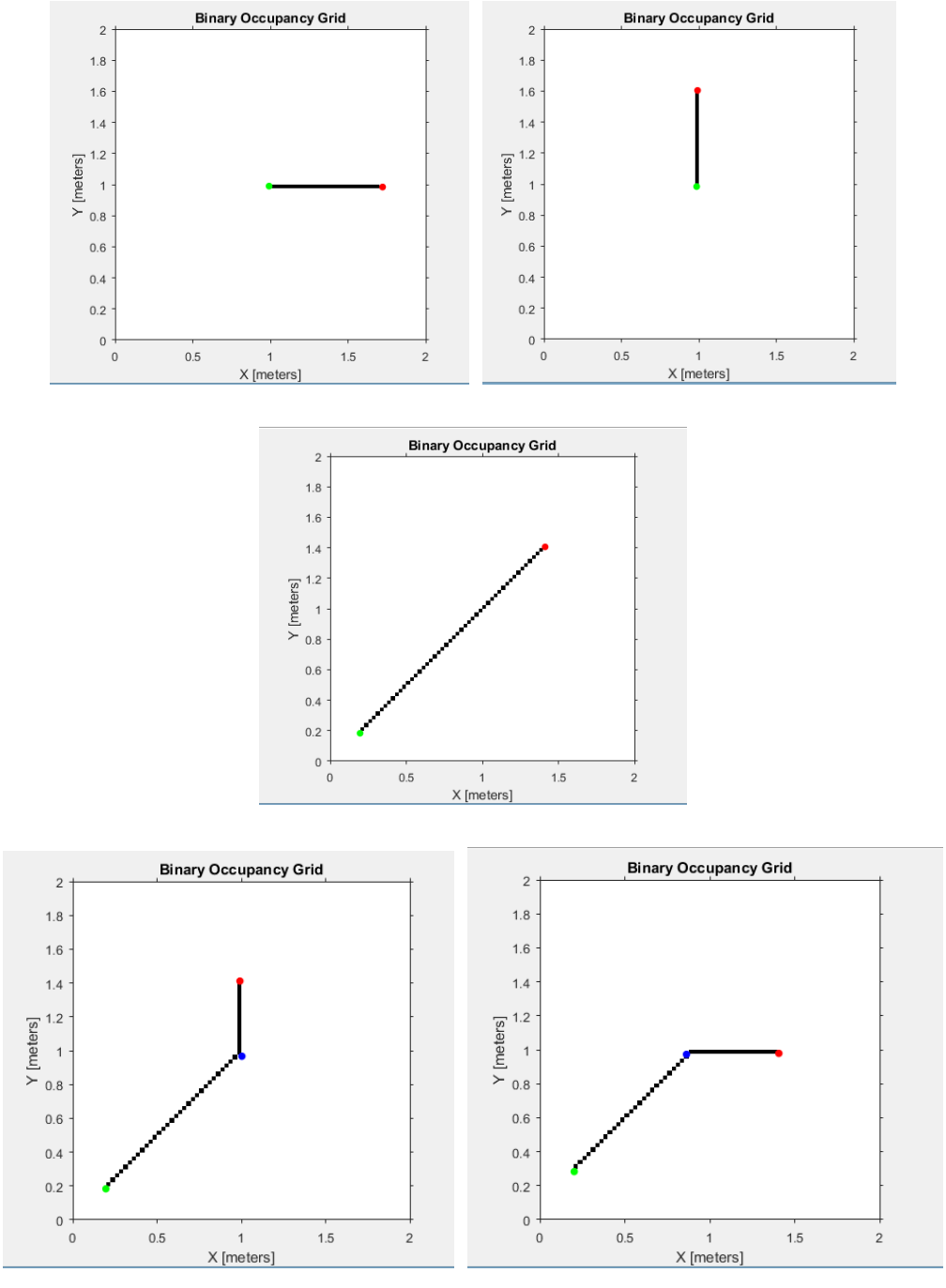


Figure 5.2 Path Planning Examples (green dot – starting point, red dot – destination, blue dot – change of heading angle)

6. ODOMETRY AND CONTROL

6.1 Localization of Mobile Robot

Odometry is the use of data from motion sensors to estimate change in position over time. Each wheel has a role in enabling the whole robot to move. Wheel encoders in the DaNI 1.0 mobile robot gives each wheel's velocity directly in radians per second through LabVIEW. Forward kinematics model is used to compute the POSE of the mobile robot.

6.2 Forward Kinematics Model

This differential drive robot has two wheels, each with diameter **r**. Given a point **P** centred between the two drive wheels, each wheel is a distance **l** from **P**. Let **θ** represent the orientation of the robot local reference plane with the global reference frame. Given **r**, **l**, **θ**, and the spinning speed of each wheel, ϕ_1 and ϕ_2, a forward kinematics model would predict the robot's speed in the global reference frame.

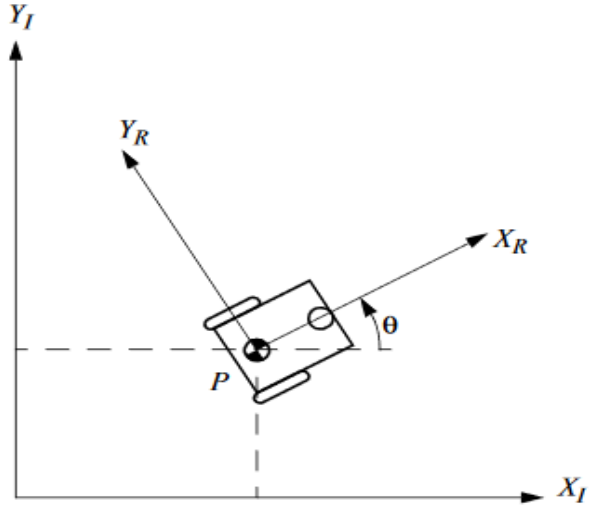


Figure 6.1 Global Reference Frame and Robot Local Reference Frame

Let the contribution of each wheel's spinning speed to the translation speed at **P** in the direction of $+\mathbf{X_R}$. If one wheel spins while the other wheel contributes nothing and is stationary, since **P** is halfway between the two wheels, it will move instantaneously with half the speed: $\bar{\mathbf{X}}_{r1} = \mathbf{(1/2)r\Phi_1}$ and $\bar{\mathbf{X}}_{r2} = \mathbf{(1/2)r\Phi_2}$. In a differential drive robot, these two contributions can simply be added to calculate the $\bar{\mathbf{X}}_r$ component of $\boldsymbol{\xi}\mathbf{R}$. The value of $\bar{\mathbf{Y}}_r$ is even simpler to calculate. Neither wheel can contribute to sideways motion in the robot's reference frame, and so $\bar{\mathbf{Y}}_r$ is always zero. Forward spin of right wheel results in counter-clockwise rotation at point **P**. The rotation velocity $\boldsymbol{\omega}_1$ at **P** can be computed because the wheel is instantaneously moving along the arc of a circle of radius **2l**:

$$\omega_1 = \frac{r\dot{\phi}_1}{2l} \tag{6.1}$$

The same calculation applies to the left wheel, with the exception that forward spin results in clockwise rotation at point **P**:

$$\omega_2 = \frac{-r\dot{\phi}_2}{2l} \tag{6.2}$$

Combining these individual formulas yields a kinematic model for the mobile robot:

$$\dot{\xi}_I = R(\theta)^{-1} \begin{bmatrix} \frac{r\dot{\phi}_1}{2} + \frac{r\dot{\phi}_2}{2} \\ 0 \\ \frac{r\dot{\phi}_1}{2l} + \frac{-r\dot{\phi}_2}{2l} \end{bmatrix} \tag{6.3}$$

where

$$R(\theta)^{-1} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.4}$$

It represents the rotation matrix for the change in orientation of the mobile robot.

On simplification,

$$\dot{\theta}_R = (\Phi_1-\Phi_2)r/2l \quad \textbf{(6.5)}$$

$$\bar{X}_r = \cos\theta\,((\Phi_1+\Phi_2)r/2) \quad \textbf{(6.6)}$$

$$\bar{Y}_r = \sin\theta\,((\Phi_1+\Phi_2)r/2) \quad \textbf{(6.7)}$$

On integrating the above parameters with respect to time we will get x-coordinate, y-coordinate and orientation of the mobile robot with respect to the global frame.

6.3 Control of Mobile Robot

The controlling of differential drive mobile robot is done by varying the velocity and direction of rotation of each wheel. For turning left, right wheel velocity is set at 3 rad/s and left velocity is set at 0 rad/s. For turning right, left wheel velocity is set at 3 rad/s and right velocity is set at 0 rad/s. To move in the straight direction both wheel velocities are set with the same value and the direction of rotation of both wheels are changed to go in the opposite direction. To rotate about a point, the wheel velocities are set to same value but opposite in direction.

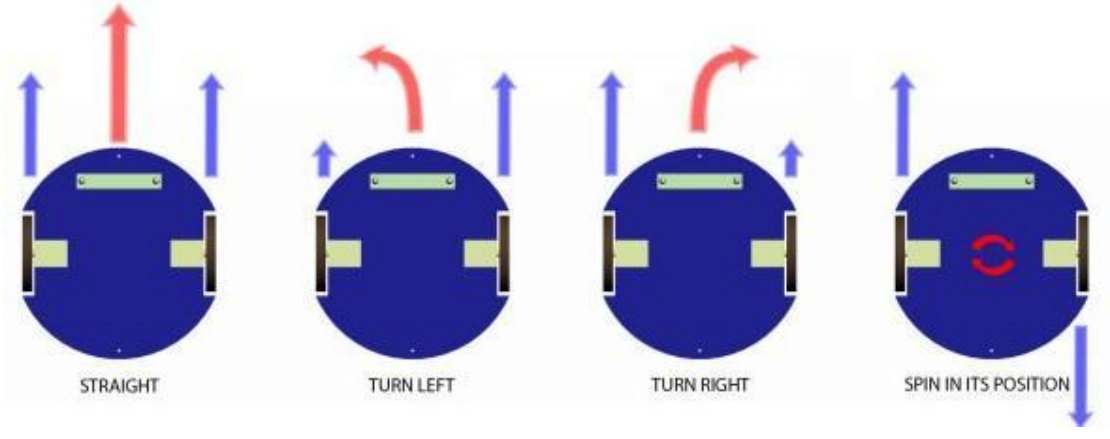


Figure 6.2 Control of Differential Drive Mobile Robot

6.4 Implementation of Odometry and Control in LabVIEW

To implement the algorithm, robotics toolbox in LabVIEW is used along with basic mathematical functions. The important operations that takes place inside the control loop are

- The analog inputs from Arduino (i.e. left and right wheel velocities) are read.
- Velocities are scaled up to their actual values.
- These values are used in the forward kinematics model.
- Mobile robot POSE (i.e. x-coordinate, y-coordinate and heading angle) is then scaled down and converted to analog outputs and these values are read by Arduino and sent to MATLAB for updating the occupancy grid.

The implementation of Odometry & Control loop in LabVIEW is given in the next page.

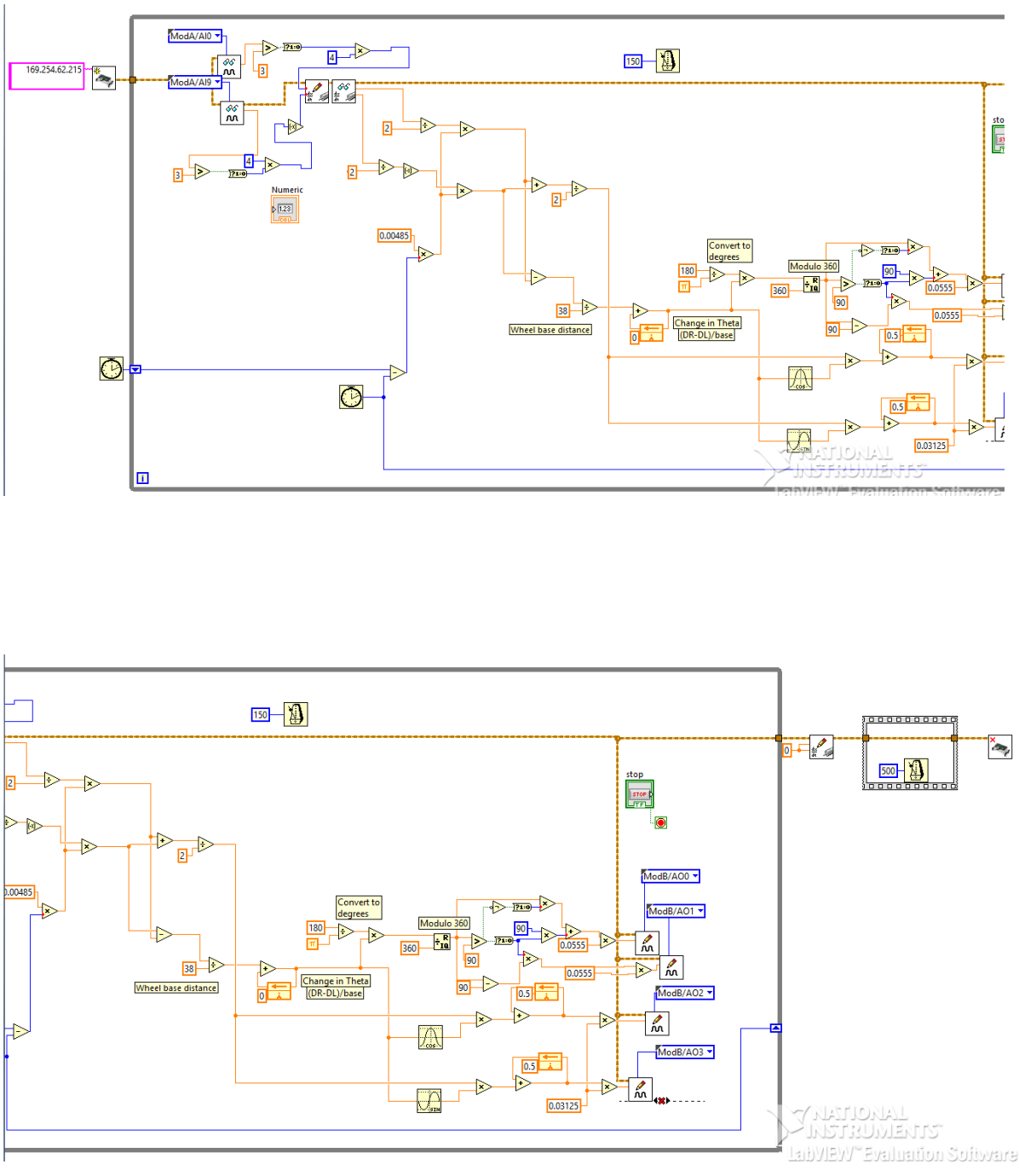


Figure 6.3 Implementation of Odometry and Control Loop in LabVIEW

5. CONCLUSION AND FUTURE PLAN

The initial aim of this project was to plan a path to the destination, identity the obstacles along the way and avoid them to reach the destination. In this project, we successfully demonstrated autonomous navigation and control of the mobile robot while mapping the obstacles around its path. There was a slight offset between the actual destination and the destination reached by the robot due to errors in occupancy grid updation, wheel encoders and analog signal reception and generation. The accumulated errors due to all these factors can be reduced by refining the image processing algorithm, removing the need for signal generation and reception by serial communication and using better sensors for mobile robot localization such as 3-DOF accelerometer. The project can be extended further by implementing obstacle avoidance in path planning. Genetic algorithm can also be implemented to generate optimal paths.

REFERENCES

[1] Roland Siegwart, Illah R. Nourbakhsh - "*Introduction to Autonomous Mobile Robots",* A Bradford Book. The MIT Press.

[2] Gregory Dudek, Michael Jenkin – *"Computational Principles of Mobile Robotics",* Cambridge University Press.

[3] Create occupancy grids with binary values – MATLAB – Mathworks India - https://in.mathworks.com/help/robotics/ref/robotics.binaryoccupancygrid-class

APPENDIX

(a) CODE

```
clc;
clear all;
close all;
%% Initial definition of variables

% Occupancy Grid
length_map=4;
breadth_map=4;
cellspermetre=40;
cmpercell=100/cellspermetre;
map = robotics.BinaryOccupancyGrid(length_map,breadth_map,cellspermetre);

% Robot position ( Initial )
x_pos = 0.5;
y_pos = 0.5;
heading_angle=0;
curr_pose = [ x_pos y_pos ];

% Setting Target
xt=input('enter the x coordinate of the target point : ');
yt=input('enter the y coordinate of the target point : ');
tar_pose = [ xt yt ];

% Motion Control
ard=arduino();
b=5;    % analog voltage of speed
```

```
% Image acquisition
depthVid=videoinput('kinect','2','Depth_320x240');
depthVid.FramesPerTrigger = Inf;

%% Central Loop
% 3D Vision
start(depthVid)
flag = 1;
while (flag)
   initial_img=getdata(depthVid,1);
   size_img = size(initial_img);   % size of image ( row,col )
   flipped_img=flip(initial_img,2);    % to undo lateral inversion by row elements reversing
   range_img=zeros(size_img(1,1),size_img(1,2));  % limit the range to 80-120 cms

for i=1:size_img(1,1)
   for j=1:size_img(1,2)
      if ((flipped_img(i,j)>=806)&&(flipped_img(i,j)<=1228))
         range_img(i,j)=flipped_img(i,j);
      else
         range_img(i,j)=0;
      end
   end
end

% Method of Variances - Floor Removal
col_var = {};
for j = 1:size_img(2)
   col_var = [col_var,var(range_img(1:size_img(1),j))];
end
```

```
for j = 1:320
   if col_var{j} > 1.4e+05
      result_col(1:size_img(1),j) = range_img(1:size_img(1),j);
   else
      result_col(1:size_img(1),j) = 0;
   end
end

row_var = {};
for j = 1:size_img(1)
   row_var = [row_var,var(range_img(j,1:size_img(2)))];
end

for j = 1:240
   if row_var{j} > 1.7e+05
      result_row(j,1:size_img(2)) = range_img(j,1:size_img(2));
   else
      result_row(j,1:size_img(2)) = 0;
   end
end

segmented_img = result_col & result_row;     % after segmentation

stats_centroid = regionprops(segmented_img,'centroid');      % getting centroid value
centroid=cell2mat(struct2cell(stats_centroid));
stats_bb = regionprops(segmented_img,'BoundingBox');
b_box=cell2mat(struct2cell(stats_bb));
size_b_box = (size(b_box,2));

if(size_b_box)
tot_obst = size_b_box/4;
```

```
for a=1:tot_obst
   term=1+((a-1)*4);
   left(a)=round(b_box(1,term));
   right(a)=round(b_box(1,term)+b_box(1,term+2));
   obst_breadth=right(a)-left(a);

   for i=1:240
      for j=1:320
         if(segmented_img(i,j)==0)
         range_img(i,j)=0;
         end
      end
   end
[r,c,v]=find(range_img);
pix_depth=mean(v);          % average value of depth
cm_depth=pix2cmdepth(pix_depth);
pix_breadth=max(c)-min(c);
pix_height=max(r)-min(r);
[cm_breadth,cm_height] = dim_pix2cm(cm_depth,pix_breadth,pix_height);

% Occupancy Grid Updation
m_depth=cm_depth/100;
m_breadth=cm_breadth/100;
m_height=cm_height/100;

inc = obst_breadth/(cm2pix_breadth(cm_depth, cmpercell));
for ogu=left(a):inc:right(a)
   offset=ogu-160;
   cm_offset = bre_pix2cm(cm_depth,offset);
   m_offset=cm_offset/100;
   alpha = -(atand(cm_offset/cm_depth));
```

```
    image_m=x_pos+(m_depth*cos(heading_angle));
    image_n=y_pos+(m_depth*sin(heading_angle));
    obst_dist=(((m_depth)^2)+((m_offset)^2))^0.5;
    x_add=obst_dist*cosd(heading_angle+alpha);
    y_add=obst_dist*sind(heading_angle+alpha);
    obst_x=x_pos+x_add;
    obst_y=y_pos+y_add;
    if ((obst_x >= 0)&&(obst_x <= 4))
        if ((obst_y >= 0)&&(obst_y <= 4))
            mn = [ obst_x obst_y ];
            setOccupancy(map,mn,1);
        end
    end
end

end         % end of occupancy grid updation for 1 obstacle loop
end         % end of processing obstacles loop
% End of 3D Vision

% Path Planning
dec_turn = min(xt,yt);

if ( x_pos == dec_turn )
    if ( y_pos < dec_turn )
        theta = 90;
    else
        theta = 270;
    end
elseif ( y_pos == dec_turn )
    if ( x_pos < dec_turn )
        theta = 0;
```

```
    else
        theta = 180;
    end
elseif (( x_pos < dec_turn )&&( y_pos <= dec_turn ))
    theta = atand((dec_turn-y_pos)/(dec_turn-x_pos));

elseif (( x_pos < dec_turn )&&( y_pos > dec_turn ))
    theta = 360 + atand((dec_turn-y_pos)/(dec_turn-x_pos));

elseif (( x_pos > dec_turn )&&( y_pos <= dec_turn ))
    theta = 180 + atand((dec_turn-y_pos)/(dec_turn-x_pos));

elseif (( x_pos > dec_turn )&&( y_pos > dec_turn ))
    theta = 180 + atand((dec_turn-y_pos)/(dec_turn-x_pos));
end

curr_loc_grid = world2grid(map,curr_pose(:,1:2));       % POSE updated from
odometry
tar_loc_grid = world2grid(map,tar_pose(:,1:2));
% x in world = col in grid, y in world = row in grid
% x min to max = col min to max, y min to max = row max to min
curr_row = curr_loc_grid(:,1);
curr_col = curr_loc_grid(:,2);

tar_row = tar_loc_grid(:,1);
tar_col = tar_loc_grid(:,2);

flag=1;

    if(curr_col<tar_col)
        curr_col=curr_col+1;
```

```
elseif(curr_col>tar_col)
   curr_col=curr_col-1;
end
if(curr_row<tar_row)
   curr_row=curr_row+1;
elseif(curr_row>tar_row)
   curr_row=curr_row-1;
end

curr_loc_grid=[ curr_row curr_col ];
occval = getOccupancy(map,curr_loc_grid,'grid');

if(occval==1)
   break;          % stop if encountering an obstacle
end

setOccupancy(map,curr_loc_grid,1,'grid');
curr_pose = grid2world(map,curr_loc_grid);
flag1 = mean(curr_pose==[ dec_turn-0.0125 dec_turn-0.0125 ]);
% 0.0125 = workspace constant

if(flag1)
   if( curr_loc_grid(:,2) < tar_col )
      theta = 0;
   elseif ( curr_loc_grid(:,2) > tar_col )
      theta = 180;
   elseif ( curr_loc_grid(:,1) < tar_row )
      theta = 270;
   elseif ( curr_loc_grid(:,1) > tar_row )
      theta = 90;
   end
```

```
    end

    flag = mean(curr_loc_grid~=tar_loc_grid) ;
% End of path planning

% Control Loop
heading_ang_vol=readVoltage(ard, 'A0');
pause(0.02);
heading_ang_vol_2=readVoltage(ard, 'A1');
pause(0.02);
heading_angle=(heading_ang_vol+heading_ang_vol_2)/0.055;
fnl_heading_angle=0;
if((theta-heading_angle)>4)
    heading_ang2=mod(heading_angle+theta,360);
    while(fnl_heading_angle-heading_ang2<0)
        writePWMVoltage(ard, 'D5', b);
        writePWMVoltage(ard, 'D6', 0);
        fnl_heading_vol=readVoltage(ard, 'A0');
        pause(0.02);
        fnl_heading_vol_2=readVoltage(ard, 'A1');
        pause(0.02);
        fnl_heading_angle=(fnl_heading_vol+fnl_heading_vol_2)/0.055;
    end

    x_vol=readVoltage(ard,'A3');
    pause(0.02);
    x_pos=x_vol/3.125;
    y_vol=readVoltage(ard,'A4');
    pause(0.02);
    y_pos=y_vol/3.125;
```

```
 elseif((theta-heading_angle)<-4)
     heading_ang2=mod(heading_angle+theta,360);
    while(abs(fnl_heading_angle-heading_ang2)>2)
        writePWMVoltage(ard, 'D5', 0);
        writePWMVoltage(ard, 'D6', b);
        fnl_heading_vol=readVoltage(ard, 'A0');
        pause(0.02);
        fnl_heading_vol_2=readVoltage(ard, 'A1');
        pause(0.02);
        fnl_heading_angle=(fnl_heading_vol+fnl_heading_vol_2)/0.0555;
    end


    x_vol=readVoltage(ard,'A3');
    pause(0.02);
    x_pos=x_vol/3.125;
    y_vol=readVoltage(ard,'A4');
    pause(0.02);
    y_pos=y_vol/3.125;
elseif(flag==0)
    writePWMVoltage(ard, 'D5', 0);
    writePWMVoltage(ard, 'D6', 0);
    x_vol=readVoltage(ard,'A3');
    pause(0.02);
    x_pos=x_vol/3.125;
    y_vol=readVoltage(ard,'A4');
    pause(0.02);
    y_pos=y_vol/3.125;
else
    writePWMVoltage(ard, 'D5', b);
    writePWMVoltage(ard, 'D6', b);
    x_vol=readVoltage(ard,'A3');
```

```
    pause(0.02);
    x_pos=x_vol/3.125;
    y_vol=readVoltage(ard,'A4');
    pause(0.02);
    y_pos=y_vol/3.125;
end

x_pos=x_pos*0.025;
y_pos=y_pos*0.025;
curr_pose=[x_pos y_pos];

end          % End of central loop

stop(depthVid);
flushdata(depthVid);
```