KINECT SENSOR ON MOBILE ROBOT
AUTONOMOUS INDOOR MOBILE ROBOT NAVIGATION

PUANG EN YEN

A project report submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering (Hons.) Mechatronics Engineering

Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

April 2013

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature : _____

Name : _____

ID No. : _____

Date : _____

APPROVAL FOR SUBMISSION

I certify that this project report entitled **"KINECT SENSOR ON MOBILE ROBOT- AUTONOMOUS INDOOR MOBILE ROBOT NAVIGATION"** was prepared by **PUANG EN YEN** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Engineering (Hons.) Mechatronics Engineering at Universiti Tunku Abdul Rahman.

Approved by,

Signature : _____

Supervisor : Dr. Ng Oon-Ee

Date : _____

Specially dedicated to
my beloved grandmother, mother and father

ACKNOWLEDGEMENTS

I would like to thank everyone who had contributed to the successful completion of this project. I would like to express my gratitude to my research supervisor, Dr. Ng Oon-Ee for his invaluable advice, guidance and his enormous patience throughout the development of the research.

In addition, I would also like to express my gratitude to my loving parent and soul mate who had helped and given me encouragement. Precious advises and suggestions are also given by them as public point of view on this project.

The highest gratitude of mine is granted to Udacity, an online free university which provides many courses in various field of knowledge. Some ideas of this project were inspired by a class called ‘Artificial Intelligent of Robotics’.

KINECT SENSOR ON MOBILE ROBOT
AUTONOMOUS INDOOR MOBILE ROBOT NAVIGATIO

ABSTRACT

Vehicle had been founded thousands years ago. It was evolved from human powered to engine propelled and its development showing no sign of stopping. The criteria of recent automobile development are focusing on safety and environment friendly. Autonomous vehicle is the current trend of research which emphasizing on improves traffic efficiency and reduces accident caused by human error. Autonomous mobile robot even developed into indoor applications to carry out numerous logistic tasks. Autonomous robotics in indoor environments is traditionally done using a combination of laser range finders, time of flight cameras, and/or stereo vision. This work proposes the use of the Microsoft® Kinect sensor as an alternative lower-cost modality for low-cost mobile robotics. The indoor environment is scanned by Kinect sensor and processed to produce a global map which contains all robots' navigable region within the indoor environment. By using the global map, robot path planner is able to search through the environment and generates path to reach its destination from a starting point. For better motion performance, path is being smoothened afterward to reduce the number of turning point in path. During its navigations, it might face new obstacle that never recorded in global map. Obstacle detection and avoidance module are implemented on the robot to detect new static and dynamic obstacle during robot's forward motion. A modification on global map will carry-out after a static obstacle is found to facilitate subsequence path planning routine. As the result, the mobile robot is shown to be capable of navigating a previously captured global map autonomously with obstacle detection and avoidance functions.

TABLE OF CONTENTS

DECLARATION **ii**
APPROVAL FOR SUBMISSION **iii**
ACKNOWLEDGEMENTS **vi**
ABSTRACT **vii**
TABLE OF CONTENTS **viii**
LIST OF TABLES **xi**
LIST OF FIGURES **xii**
LIST OF SYMBOLS / ABBREVIATIONS **xvi**
LIST OF APPENDICES **xvii**

CHAPTER

1 INTRODUCTION **1**
1.1 Background 1
1.2 Problem Statement 2
1.3 Aims and Objectives 3

2 LITERATURE REVIEW **5**
2.1 Range Sensor 5
2.1.1 Passive Stereo Vision 6
2.1.2 Laser Range Finder 10
2.1.3 Microsoft Kinect 12
2.1.4 Sensors Merging 14
2.2 Navigation and Algorithm 16
2.2.1 Line Following 16

	2.2.2	Localization	17
	2.2.3	Path Planning	21
	2.2.4	Obstacle Handling	23
	2.2.5	PID Controller	24
3	**METHODOLOGY**		**26**
3.1	Robot Hardware		26
	3.1.1	Robot Driving System	26
	3.1.2	Embedded System	28
	3.1.3	Robot Sensors	29
3.2	Robot Software		32
	3.2.1	Kinect Software	32
	3.2.2	Programming Tools	36
3.3	Project Job Scope		38
	3.3.1	Hardware design and Fabrication	38
	3.3.2	Hardware Control	39
	3.3.3	Kinect Programming	41
	3.3.4	Robot’s Intelligent	42
3.4	Project Management		43
4	**RESULTS AND DISCUSSIONS**		**45**
4.1	Global Map		45
	4.1.1	Global Map Acquisition	45
	4.1.2	Global Map Processes	46
4.2	Particle Filter		53
	4.2.1	Filter’s Phases	53
	4.2.2	Filter Performance	56
4.3	Path Planner		56
	4.3.1	Path Searching	57
	4.3.2	Path Smoother	58
	4.3.3	Path Planner Performance	60
4.4	Obstacle Handler		61
	4.4.1	Obstacle Detection	61

4.4.2 Obstacle Avoidance 62
4.4.3 Obstacle Handler Performance 64
4.5 Motor Control 64
4.6 Visual Odometry 65
4.6.1 Visual Odometry Performance 66
4.7 Hardware Architecture 67
4.7.1 Robot Hardware Structure 68
4.7.2 Encoder Odometry 69
4.7.3 DC Motor and Driver 72
4.7.4 Robot Embedded System 73
4.7.5 Communication Protocol 74
4.7.6 Motor Control System 76
4.8 Overall Performance 77

5 CONCLUSION AND RECOMMENDATIONS 80

REFERENCES 82

APPENDICES 86

LIST OF TABLES

TABLE	TITLE	PAGE
2-1	Ziegler-Nichols Tuning Equations	25
4-1	Particle Filter Experimental Result	56
4-2	Result of Path Length against Processing Time	60
4-3	Robot's Respond Time to Obstacle	64
4-4	Motion Data Series	75

LIST OF FIGURES

FIGURE	TITLE	PAGE
1-1	Advance Range Sensor	2
1-2	Xbox Gaming with Kinect	3
2-1	Ultrasonic and Infra-red Range Sensor	6
2-2	Ultrasonic Sensor's Sensing Ranges	6
2-3	Mars Exploration Rover	7
2-4	Obstacle Detection and Avoidance	8
2-5	Flow Chart of Simple Obstacle Avoidance Robot	8
2-6	Omnidirectional Vision System	9
2-7	Original and Processed Images of Terramax	9
2-8	Rotational Laser Range Finder	10
2-9	Stanley's Laser Range Finder	11
2-10	Stanley's Local Map	11
2-11	Kinect's Naked View	12
2-12	Segmented Depth Map & ANN Recognized Paths	13
2-13	Kinect Sensor on Quadrotor Helicopter	14
2-14	Multiple Sensors Merged on Indoor Robot	15
2-15	Junior The Self-driving Car	15
2-16	Transcar and its Path Network	16
2-17	Differential Drive Robot	17

2-18	Graphical Illustration on Odometry System	17
2-19	Particle Filter in Action	20
2-20	A* Search and Dynamic Programming Result	22
2-21:	Before and After Path Smoother	23
2-22:	PID Block Diagram	24
3-1	Mecanum Wheel and its Drive Method	27
3-2	Omni-Wheel and its Drive Method	27
3-3	Arduino™ Logo	28
3-4	Microchip® Logo	29
3-5	ESB Rotary Encoder and its Output Pulses	31
3-6	Simple Infra-red Emitter and Receiver Pair	31
3-7	OpenKinect Logo	33
3-8	Kinect for Windows Logo	34
3-9	Role of OpenNI between Device and Application	35
3-10	Point Clout Library Logo	36
3-11	Cmake Graphical User Interface	38
3-12	Autodesk Inventor User Interface	39
3-13	Eagle Circuit and Layout Design Interface	40
3-14	MPLAB-X Programming IDE	40
3-15	Text Editor and Command Line Interface Programming in Ubuntu Platform	41
3-16	Project Gantt Chart	44
4-1	Skanect and Scenect 3D Reconstruction Software Interface	46
4-2	Global Map Process Flows	46
4-3	Result of Voxel Grid Filter	47

4-4	Result of Radius Outliner Remover	48
4-5	Result of Plane Segmentation, 3D Transformation and Plane Indices Extraction	49
4-6	Result of Pass Through Filter in Z Axis	50
4-7	Result of Up-Sampling	51
4-8	Result of Plane-Obstacle Overlap Remover	51
4-9	Result of Euclidean Distance Clustering	52
4-10	Comparison between Raw Scanned and Processed Global Map	52
4-11	Particle filter in Action	55
4-12	Particle Filter Flows	55
4-13	Non-Optimized Path Search and Optimized Path Smoother	58
4-14	Comparison of Different Planned Path	60
4-15	Transformation of Kinect Field of View Through-out Entire Path	62
4-16	Global Map Modification and Path Re-planning	63
4-17	Overall Software Architecture with Visual Odometry	65
4-18	Week Performed Region (a) and Planned Compared to Actual Travelled Path (b)	66
4-19	Hardware Architecture	67
4-20	The Robot Complete Hardware	68
4-21	Structure Isometric View	68
4-22	Placement of Kinect Sensor on Robot’s Structure	69
4-23	Kinect Sensor Field of View	69
4-24	Software Architecture with Encoder Odometry	70
4-25	IR Sensor and Encoder Disk	71

4-26	Schematic and Board of IR Encoder Board	71
4-27	DC Motor and Shaft-Wheel Coupling	72
4-28	Schematic of Microcontroller Board	74
4-29	Motor Driver, Microcontroller Board and USB-UART Converter	74
4-30	Motion Command's Communication Protocol	75
4-31	Motor Control System Diagram	77
4-32	Graph of Speed against Travelled Distance	77
4-33	Result of Starting Point's Coordinate and Heading Error	78
4-34	Overall Robot's Functionality Flow Chart	79

LIST OF SYMBOLS / ABBREVIATIONS

dsPIC	Digital signal controller
UART	Universal Asynchronous Receiver Transmitter
TTL	Transistor-Transistor Logic
PCL	Point Cloud Library
RNDF	Route Network Definition File
ROS	Robot Operating System
RGB	Red Green Blue
IR	Infra-red
ANN	Artificial Neural Network
CCD	Charge-Coupled Device
LED	Light Emitting Diode
SDK	Software Development Kit
API	Application Programming Interface
OOP	Object-oriented Programming
DC	Direct Current
3D	3 Dimensions
OS	Operating System

LIST OF APPENDICES

APPENDIX	TITLE	PAGE
A	Robot Hardware Structural Drawing	86
B	Project's Conference Paper	87
C	Conference Paper Acceptance Letter and Review Form	93

CHAPTER 1

INTRODUCTION

1.1 Background

As technology evolves, autonomous robots have become more and more popular in the manufacturing, health care, security, military and similar sectors. The primary benefit of this technology is that it can function without human intervention and yet perform better under certain circumstances. In other words, autonomous robots can replace human in tasks which are dangerous, repetitive and require high levels of concentration over a long time.

Unlike manual-control robot, autonomous robot does not rely on live input commands. Autonomous robot makes decisions by itself based on its sensor inputs and pre-defined behaviours. Therefore, it can operate with minimum of human supervision and correction. Hence, it behaves as an autonomous individual in its environment.

Autonomous indoor mobile robots are required to navigate through an indoor environment. In order for an indoor mobile robot to be considered autonomous, it must at least have the following capabilities. First, it should be able to locate itself within the indoor environment. Second, it should be able to plan its path to reach it destination. Third, it should be able to correct itself if it deviates from the planned path. Fourth, it should be able to detect and avoid new obstacles. There are also many other capabilities an autonomous mobile robot needs, depending on its function and how the word ‘autonomous’ being defined by the designer.

1.2 Problem Statement

Obstacles for outdoor mobile robot are diversified, for instance trees, slopes, traffic lights, road lanes, vehicles, pedestrians, and other environmental factors common in outdoor environments. On the other hand, indoor mobile robot only needs to process data that concerning the navigable floor areas and possible indoor obstacles such as furniture, human and wall. Although the environment classification's complexity is slightly lower, indoor mobile robot still having strict criterion on sensor requirement.

There are many types of sensors which can be used by autonomous indoor mobile robots. Popular choices include laser range finders, time-of-flight cameras and stereo camera as shown in figure 1-1. However, these sensors can be either too expensive or too computationally complex (or both) for this application. Laser and time-of-flight sensors are one of the costly devices which price can goes from a thousand to tens thousands USD. Stereo camera can be built economically but it requires additional calibration and complex data processing if compared to laser sensor or time-of-flight camera.

Figure 1-1: Advance Range Sensor

(Ptgrey 2013, Sick 2013, Mesa-imaging 2013)

Kinect was introduced by Microsoft® at year 2010.It immediately created a huge wave that hit the world of video gaming. The first version of Kinect appears as an Xbox360 video game console and function as a player's motion sensing input device and its microphone also allow user to interact using voice. Instead of having a controller in player's hand, Kinect allow player to interact with the game by using

their body. In other word, Kinect changes ordinary gaming style (finger interact) into a brand new method (whole body interact).

Figure 1-2: Xbox Gaming with Kinect

(John 2010)

Besides, Microsoft® Kinect™ drew interest from robotics hobbyists as well. This sensor is a low cost device which able to produces RGB image, depth image and sound recordings. Although its depth accuracy and sensing range are not as good as advance range sensors, Kinect sensor still provides a good trade-off between price and data quality for indoor mobile robotics.

One other benefit of the Kinect sensor is the numerous open source Kinect™ libraries available for developers, easing implementation of the Kinect sensor on other applications besides gaming. Some examples of these libraries are OpenKinect, OpenNI, Kinect for Windows, Point Cloud Library (PCL), and the Robotic Operation System (ROS).

1.3 Aims and Objectives

The main aim of this project is to produce a robot which able to navigate through-out an indoor environment autonomously, freely, and accurately without collision with obstacles. To achieve this, Kinect sensor is used as the vision sensor together with

other sensors if requires. Besides, this robot which equipped with Kinect sensor should have similar standard or small deviation in term of level of performance with robot which equipped with advance system and sensors.

In this project, there are few objectives to be achieved:

- To apply new technology, Kinect sensor on robotic application.
- To apply 3D computer vision technology on mobile robot navigation.
- To provide mobile robot the ability to generate path to its destination.
- To provide mobile robot the ability to localize itself in indoor environment.

CHAPTER 2

LITERATURE REVIEW

Advance autonomous mobile robots are mostly equipped with advance sensors to acquire different type of data, for instance robot's and obstacles' position, heading, moving speed through sensors such as laser range finder, radar sensor, global positioning system, digital compass, inertia measurement unit, time of flight and stereo camera. Together with navigation intelligent, these robots performed numerous impressive tasks that purveying conveniences into human's life.

These reviews are going to explore the sensors of most autonomous mobile robot deployed on acquiring self and environments' situations as while as the newly emerging sensor technology that applicable in mobile robotic. Besides, there are also discussions on the algorithms implemented in providing robot artificial intelligent based on their assigned task respectively.

2.1 Range Sensor

Autonomous mobile robots are not a new development and many types of range sensors have been implemented. Start from the most basic level, it can be using cheap ultrasonic and infra-red distance sensors. Mobile robots are able to find out the one dimension distance between itself and the obstacle in front.

Figure 2-1: Ultrasonic and Infra-red Range Sensor

(Maxbotic 2013, Sparkfun 2013)

These sensors are very simple and easy to build but having poor performance in certain aspect. Infra-red range sensor has low sensing area while ultrasonic range sensor has large sensing are but low space resolution. Both of them are suitable for beginner in educational purpose and not suitable in precise space measuring for advance autonomous mobile robot.

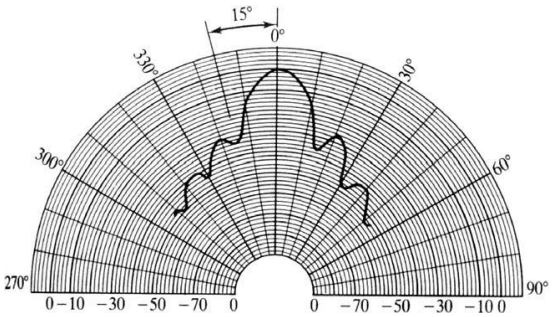


Figure 2-2: Ultrasonic Sensor's Sensing Ranges

(karsten and Ewald 2009)

2.1.1 Passive Stereo Vision

Stereo vision does not actively transmit pulses and observe reflections. Instead it passively captures light from different positions with overlap. By using multiple images from 2 or more cameras placed at a defined offset, stereo matching algorithm able to calculate the distance of the object captured in the images. The array of distances calculated is often called depth or disparity map. Stereo matching faced problems such as lack of texture, occlusion, discontinuity and noise. Several

techniques have been developed to improve the correspondence problem such as intensity, feature and phase based (Hile and Zheng 2004).

Besides, algorithms which generate depth maps from stereo cameras for mobile robotics are also a popular research topic. This technology has been widely used in a lot of robotic application. NASA's Mars Exploration Rover (MER) missions was applied autonomous passive stereo vision (Goldberg, Mark and Larry 2002) as a terrain observer while the rover navigating on Mars. In this application, stereo vision is a better choice of range sensor because the environment is an unknown terrain, it might have laser and infra-red disturbances that can interfere other type of sensor which expect to receive reflected pulses.

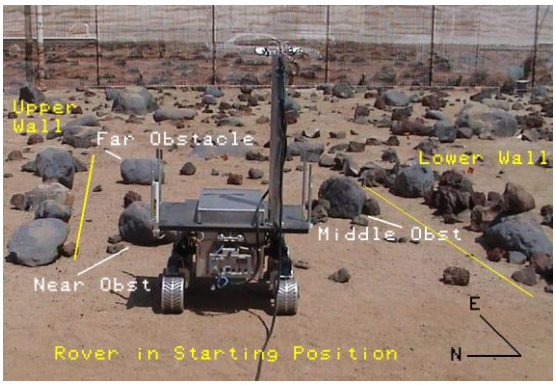


Figure 2-3: Mars Exploration Rover

(Goldberg, Mark and Larry 2002)

In unmanned robotic ground vehicle development field, stereo vision technology provides useful disparity map for navigation. Few researchers have been applied the disparity map on mobile robot mainly performing obstacle location, detection and avoidance task (Hasan, Hamzah and Johar 2009) (Mingxiang and Yunde, 2006). The basic obstacle avoidance rules are robot will turn to the other side if there is an obstacle in front of it and so on. However, these were not adequate anymore for robot nowadays because they are not only expected obstacle avoidance ability but also navigation within places.

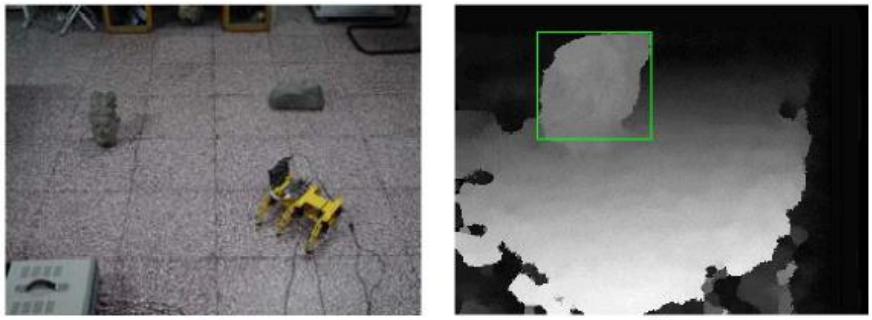


Figure 2-4: Obstacle Detection and Avoidance

(Mingxiang and Yunde, 2006)

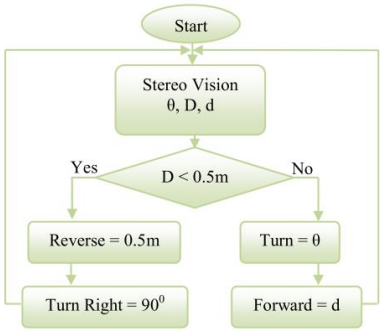


Figure 2-5: Flow Chart of Simple Obstacle Avoidance Robot

(Hasan, Hamzah and Johar 2009)

A stereo vision based driver assistance project called Enpeda Project (Liu 2007) from university of Auckland applied stereo and motion Analysis towards Analysis and Understanding of Traffic. However, the result was influenced by outdoor environment such as snow, rain, light and road condition etc. Unlike MER who moving in much slower speed, the system was not responds fast enough if the vehicle moving in high speed on road. In other word, the quality of image is extremely important for stereo algorithm because the algorithm is very sensitive to noise and take time to process.

Ordinary stereo vision system use 2 camera to capture 2 different images with offset. However, conventional camera is limited by their field of view. It can only capture image around 70 degree horizontally. Su, Luo and Zhu (2006) were implementing an effective way to increase field of view, which is constructing an

omnidirectional vision system by using mirrors in conjunction with stereo cameras to generate disparity map. Although the field of view increased, the signal to noise ratio decreased compared with ordinary stereo image. Besides, defocusing effect becomes obvious due to the different of focal length of mirrors and cameras.

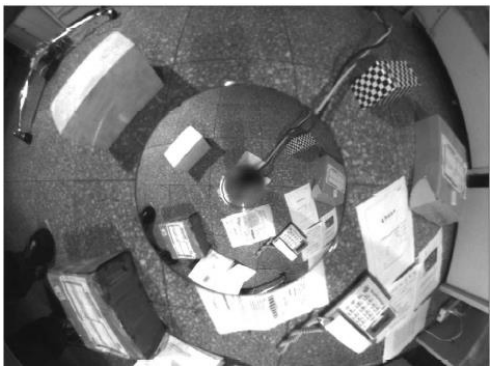


Figure 2-6: Omnidirectional Vision System

(Su, Luo and Zhu 2006)

In DARPA grand challenge 2004, a team called Terramax implemented stereo vision system on their desert challenger (Broggi et al., 2006). Their vision system provides V-disparity representation for its path planner. Since the system is designed to work in unstructured environment, ground correlation line had been computed to differentiate vehicle pitch angle. With the useful dynamic pitch information, region of interest can be accurately located. This allows system to allocate ground and region to search for obstacle. Thereafter, a global map is matched up with the local map generated by the stereo disparity map to estimates the vehicle actual coordinate on the earth.

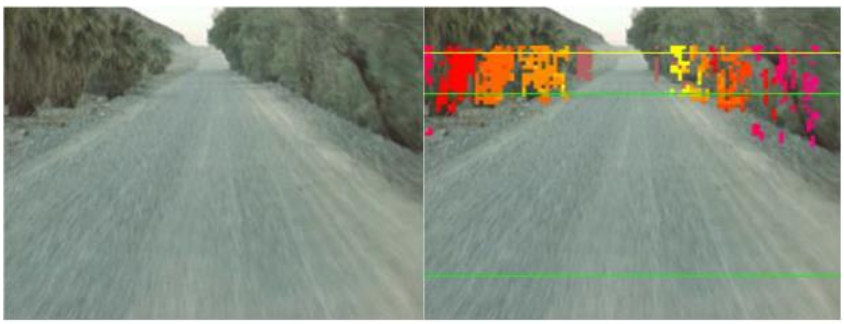


Figure 2-7: Original and Processed Images of Terramax

(Broggi et al., 2006)

2.1.2 Laser Range Finder

Laser range finders and time-of-flight cameras are currently the highest accuracy sensors in range measurement (Stoyanov et al., 2011, Salvi, Pages and Batlle 2004). They are usually used in autonomous mobile robots or vehicles which require precise range measurement to facilitate algorithms such as localization and mapping. Stereo vision performance can be as good as laser range finders under some conditions, but the computational power needed to executes stereo algorithm is significant (Antunes et al., 2012). In other word, laser range finder's data do not require another step of matching process and hence, its sensing speed is another advantage besides its accuracy. Due to their high performance, these range sensors are normally some order of magnitude more expensive than other range sensors.

Laser range finder had been widely used in the robotic development field and holds the crown among other navigator devices due to its speed and data accuracy. This is a popular method to acquiring 3D data as well as generates local map. A 2D laser range finder had been deployed on an autonomous robot (Surmann et al., 2001) to generate 3D map. Their approach is to rotate the 2D laser range finder with horizontal axis of rotation. Therefore, a 3D range finder is built on the basis of 2D range finder with servo motor. The rotation provides vertical scanning and repels the need of sensor upgrading. Therefore, the quality of the map generated is largely depends on the servo motor's resolution.

Figure 2-8: Rotational Laser Range Finder

(Surmann et al., 2001)

Stanford's robot car 'Stanley' was the winner of Defence Advanced Research Projects Agency (DARPA) grand challenge 2005 who had autonomously travels 142 miles through Mojave Desert in 6 hours and 54 minutes (Thrun et al., 2006). Together with GPS, Radar, Stanley was equipped with 5 SICK® laser range finders on the vehicle roof to form an environment sensor group. 5 of them are pointed forward to the driving direction, but slightly different in tilt angles. The laser range finders are used to monitor the surrounding environment while far obstacles are detected by radar range finder.

Figure 2-9: Stanley's Laser Range Finder

(Thrun et al., 2006)

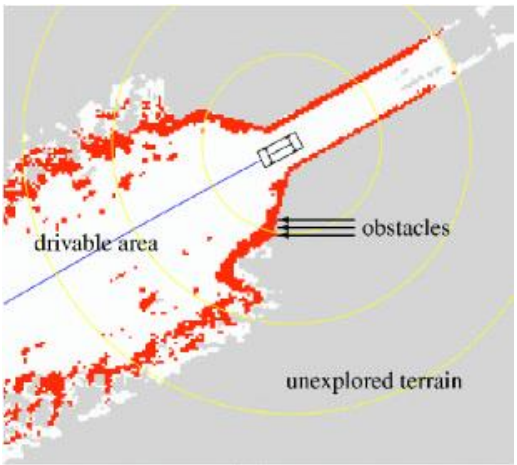


Figure 2-10: Stanley's Local Map

(Thrun et al., 2006)

Stanley is not interested its rear's condition but only the front environment conditions are being monitored carefully because in desert, neither vehicle's

movement nor path planning will be affected by the vehicle's rear condition. However, rear and front conditions are equally important if it wanted to drives in urban road and the complexity is very different from driving in desert.

2.1.3 Microsoft Kinect

Figure 2-11: Kinect's Naked View

(Smisek, Jemcosek and pajdla 2011)

Kinect has becomes more and more popular in robotic development recent years due to its acceptable price and quality. The whole Kinect come together with sensors and actuator listed below:

- RGB camera
- IR projector
- IR receiver
- Accelerometer
- Microphone array

The IR sensor is kind of active stereo vision camera where its depth camera emits structured infra-red light, receives it upon reflection from the object in front. It can easily be affected by sunlight because it is using infra-red and making it not working well under strong light or sunlight (EL-laithy, Huang and Yeh 2012). It also having a limited depth sensing ranges which around 0.5 meters to 5.5 meters. Therefore, almost all robots with Kinect sensors embedded are designed for indoor applications. But it has advantage over passive stereo vision where it able and best to

be function in the dark. Furthermore, Kinect's data streams do not require to undergo matching process likes stereo vision.

Correa et al. (2012) developed an autonomous indoor surveillance robot power by Kinect sensor. The depth image had been segmented into 5 sections and the darkness of each section in represent the distance from the robot to obstacle in front. Simple rules based algorithm navigates the robot away from the section which having shorter distance. It seems simple in navigation algorithm but artificial neural network (ANN) has been applied in the path recognition. ANN is used to recognise 8 types of path the robot is facing from the 5 segments depth image. This method reduces depth map process's difficulty and simplifying the depth image by segmented it into 5, but reduces and wasted the Kinect data's resolution. The path way robot move must not be too confusing in order ANN can recognize it based on low resolution depth image.

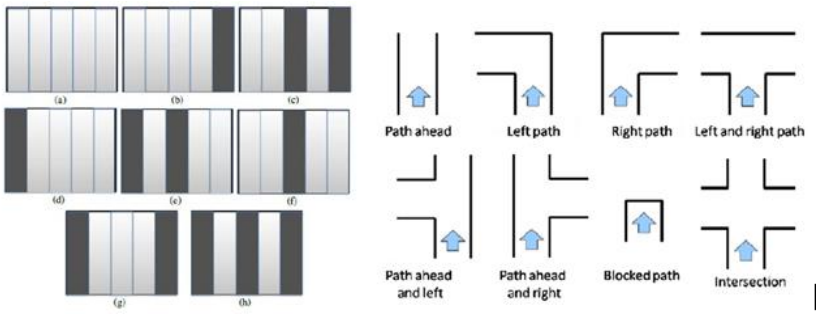


Figure 2-12: Segmented Depth Map & ANN Recognized Paths

(Correa et al., 2012)

Kinect sensor has been applied on altitude control of quadrotor helicopter (Stowers, Hayes and Bainbridge 2011). Kinect is used for measuring the height of the helicopter by depth data. Therefore, it is mounted under the craft and pointing downward to the ground. PID algorithm was applied to maintain helicopter's altitude during flight and the suitability of Kinect sensor in highly vibrates and dynamic environment has been evaluated. The result proved that Kinect is capable to function in high frequency vibration and it is important because some motor or engine used in ground robot create as much vibration as airborne robot that malfunction its sensor.

However, the Kinect was not wirelessly communicates with the main control computer but with cables. This limits the moving range of the quadcopter and reduced the practicality of this design. Ground robot has larger load capacity which able to equip its own main control PC inside itself and its motion only limited by battery capacity.

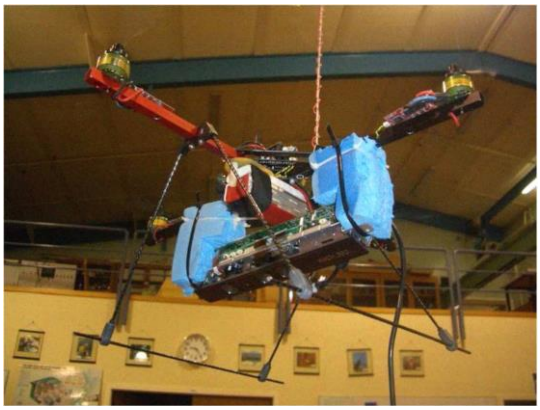


Figure 2-13: Kinect Sensor on Quadrotor Helicopter

(Stowers, Hayes and Bainbridge 2011)

2.1.4 Sensors Merging

No sensor can be optimized in all aspect, each of them having certain degree of disadvantage. In normal practice, multiple and different type of sensors are merged in one application for the purpose of covering each other backside.

Asensio, Montiel and Montano (1998) had merged stereo vision and laser range finder on a mobile robot. It applied a stereo trinocular system which using 3 synchronized CCD monochrome cameras and a 3D lidar laser range finder to generate depth map and perform door localization in indoor environment. Lin and Chen (2012) had merged omnidirectional camera with a SICK laser range finder to perform navigation and simultaneous localization and mapping (SLAM). WJ IV and Belvy (2001) had combined 3 range sensors which are Kinect, Bumblebee stereo camera and SICK laser range finder. Supplementary sensors such as rotary encoders are often added to autonomous mobile robots to perform dead reckoning localization.

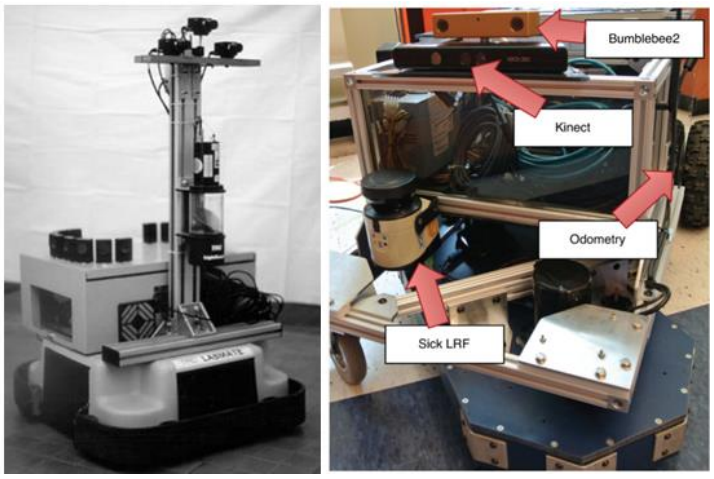


Figure 2-14: Multiple Sensors Merged on Indoor Robot

(Asensio, Montiel and Montano 1998, WJ IV and Belvy 2001)

Besides the sensors mentioned above, other sensors such as global positioning system (GPS) and radar sensors have been used in advanced autonomous vehicle(Montemerlo et al., 2008, Thrun et al., 2006). Self-driving car "Junior" was took part in the DAPRA urban challenge 2007. The competition requires self-driving car driving in urban road condition with taking in all consideration a human driver will need to have. Junior had equipped with many type of sensors in order to hander data such as road location and situation in all directions, traffic light, self-location and speed and other vehicle location, speed and heading.

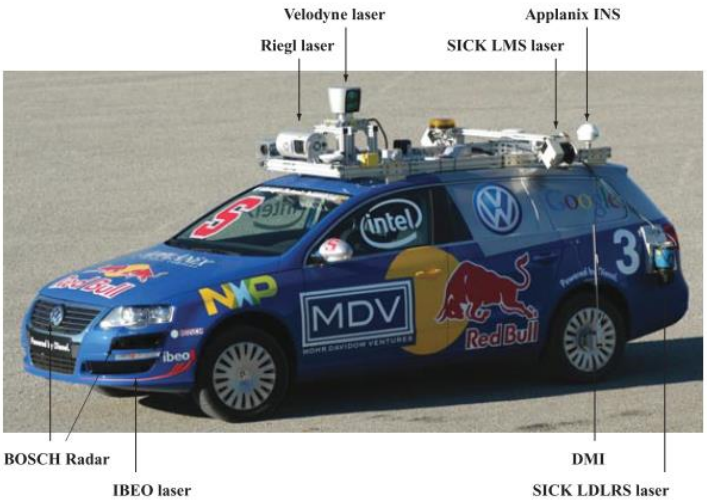


Figure 2-15: Junior The Self-driving Car

(Montemerlo et al., 2008)

Ordinary global positioning system cannot provides accurate measurements because it signals easily affected by several factors such as satellite's position inaccuracy, atmosphere condition, signal's barriers density, co-frequency inferences and receiver noises (earthmeasurements, 2000).

2.2 Navigation and Algorithm

2.2.1 Line Following

Many automation companies such as Savant automation and SwissLog had develop automated guided vehicle (AGV). Transcar is an AGV product developed by SwissLog, designed to navigate in indoor environment for purpose of transporting goods in working environment such as hospital, warehouse or other hazardous area such as high temperature, radioactive or biohazard.

These types of AGV are normally working in swarm manner and everyone is controlled by a centre controlling unit. Transcar navigation system used line following method to navigate through the environment. The entire working environment of transcar is implanted with electromagnetic line that will be picked up by the transcar line sensor during its navigation. By following the line assigned by control unit, transcar able to navigates to its destination with obstacle detection ability. It also equipped with wireless transceiver which enable it to communicate with lift and door to enlarge it working area.

Figure 2-16: Transcar and its Path Network

(Swisslog 2013)

2.2.2 Localization

Autonomous robot always faces local position tracking problem, especially indoor autonomous robot. In order to navigate reliably, robot must have the ability to localize itself in an indoor environment without the aid of GPS.

2.2.2.1 Odometry

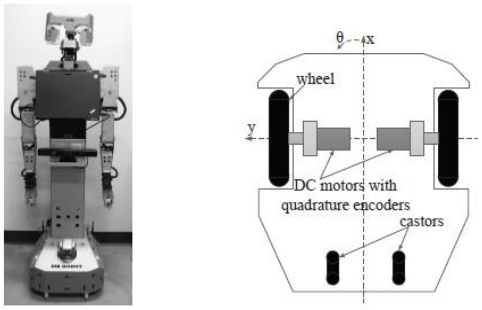


Figure 2-17: Differential Drive Robot

(Ganganath and Leung 2012)

Ganganath and Leung (2012) applied localization on a mobile robot by using odometry system and Kinect sensor. It odometry system used 2 15-bits quadrature encoders to calculate the position and direction of the robot.

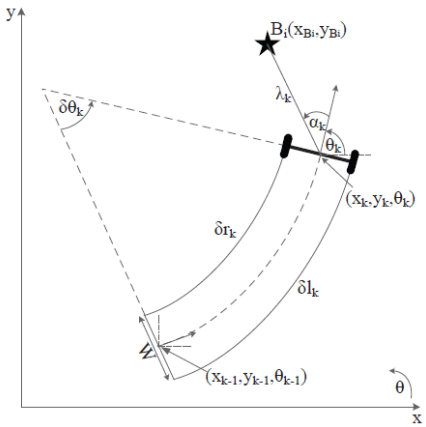


Figure 2-18: Graphical Illustration on Odometry System

(Ganganath and Leung 2012)

The system state vector, $X_k = [x_k\ y_k\ \theta_k]^T$ describes the position of the mid point of the robot's wheels's width, W in X and Y axis while the heading of the robot describes by the θ_k at time step K. The change of heading can be calculated by the equation below:

$$\delta\theta_k = \frac{(\delta r_k - \delta l_k)}{W}$$

Where

δ l = change of distance of left wheel

δ r = change of distance of right wheel

while the distance travelled by the mid point of the wheels, $\frac{W}{2}$ is calculated by the equation below:

$$\delta d_k = \frac{(\delta r_k - \delta l_k)}{2}$$

Therefore, the next position of the robot can be determine by the martrix below:

$$\hat{X}_k = \begin{bmatrix} x_{k-1} + \frac{\delta d_k}{\delta\theta_k}\left[\sin(\theta_{k-1} + \delta\theta_k) - \sin(\theta_k - 1)\right] \\ y_{k-1} + \frac{\delta d_k}{\delta\theta_k}\left[\cos(\theta_k - 1) - \cos(\theta_{k-1} + \delta\theta_k)\right] \\ \theta_{k-1} + \delta\theta_k \end{bmatrix}$$

By using these equations, the robot's position can be estimated. However, the reliability of odometry system is very dependent on the encoder mounted on the robot. If the mechanical structure cannot make sure the accuracy, it only can be one of the algorithms among other localization method applied on the robot.

2.2.2.2 Monte Carlo Localization

Monte Carlo localization (MCL) is a probabilistic approach to solve mobile robot's position tracking problem. Its multi modal distribution basis and abilities to represent ambiguities and uncertainties make it a better choice among other technique. It was evolved for Markov localization and reduced the memory required in Markov's method (Thrun 2002). However, MCL is a discrete state space filter which required to separates it working space into pieces.

MCL mainly consists of 2 phase, measure and motion, and recursively computes it probability density of each space. If the robot starting position is unknown, the probabilities of each and every cell are equal. Then the algorithm goes into measure stage. It use measurement model to incorporate environment information from sensors and produces posterior probability density which content the densities of all cells. The cell who matches the current sensed information will denser than who does not. Normalization is done after multiplying different value to match and not matches' cell's density. After this phase, cell's densities are more differentiable and higher confident in localization.

The next phase is motion phase. The motion of the robot must be integral of cell. If each cell is 5cm, robot cannot move a 12cm as single movement to comply the localizer natural discrete basis. Since the robot movements cannot be 100% certain and accurate, the probability density shifted according to the movement and flatten depend on the degree of robot movement uncertainty.

These 2 phases will repeat recursively every time step in order to track the position of robot.

2.2.2.3 Particle Filter

Particle filter (PF) is very similar to MCL in the measure and motion phases. It is also a multi-model filter as MCL which mean it can track multiple landmarks in a single set of algorithm. However, PF has advantage over MCL on its continuous state space (Thrun 2002).

The first step is to initializes a number of particles within the space where the robot moves. The initialized elements are particle's x-y coordinate and heading. The resolution of the localiser is depends on the number of particle initialized at this state. Then, it enters into measurement stage. Robot's and all particle's distances to landmarks are computed. The different between particle's distances to landmark with robot's distance to landmark are the factor to decide the weight of the particle. A new set of particles will be selected based on their weight. In other word, particles that are closer to robot will have better chance to stay while further particle will less likely to stay as particle.

After that, the robot will move and the movement will apply to all particles. Next, the measurement will repeat again on all particles. The measurement and motion steps will repeat recursively until the specific particle who closest to robot has much more weight than other and confident enough to produces robot coordinate over the space.

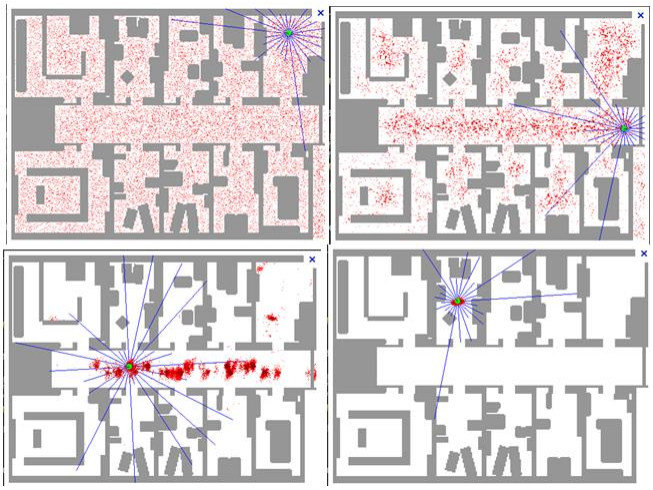


Figure 2-19: Particle Filter in Action

(Thrun 2002)

2.2.2.4 Kalman Filter Localization

It undergo the same measurement and update cycle as MCL does but it using Gaussian method and calling predict and update cycle. In predict state, prior estimation is being moved by a desired distance using total probability method. Its mean is being shifted and its covariance increased. In update state, prior and measurement combined using Bayer's rule to produces lower covariance estimation.

In predict state: $\mu' = \mu + u$

$$\sigma^{2'} = \sigma^2 + w^2$$

In update state: $\mu' = \frac{1}{\sigma^2 + r^2}[r^2\mu + \sigma^2 v]$

$$\sigma^{2'} = \frac{\sigma^2 r^2}{\sigma^2 + r^2}$$

Where

μ = mean	σ^2 = covariance
μ' = new mean	$\sigma^{2'}$ = new covariance
r = measurement uncertainty	w = motion uncertainty
v = measurement mean	u = motion mean

2.2.3 Path Planning

The path planner will partitioned the space into grid cells and every planned movement are either up, down, left or right. The algorithm implemented in the planner is A* search. A map is required for the planner before it starts. The map can be update while the robot is moving due to the detection of new obstacle.

The planner will first search the entire available grid cell adjacent to it to be its frontier for search tree expansion. If multiple grid cells are available, the planner will chooses the grid cell who takes lesser steps. The heuristic function is produced based on the distances between the grid cell and the goal.

Montemerlo et al. (2008) autonomous vehicle plan their path from a global map using dynamic programming based on hybrid A* search. Paths are computed for each discrete cell of route network definition file. It is a type of planner who instead of looking for the shortest path from starting point to goal, it plan the shortest path for every grid cell to goal. In other word, dynamic programming is able to handle large divergent sets of future decisions. This requires significant computation, especially when dealing with a large global map. Therefore, it will take more time in computations.

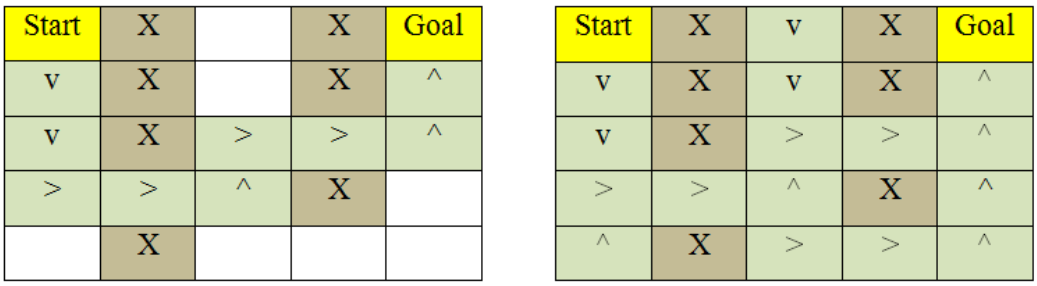


Figure 2-20: A* Search and Dynamic Programming Result

2.2.3.1 Path Smoother

Since path planner produces a discrete action for the robot to move such as move forward, backward, turn 90°, -90°, the robot will move in very 'robot' way. Therefore, we need a smoother to fillet out those 90° sharp edges and make the robot moves in smooth way. The path smoother used gradient descent to minimize the functions below and optimize their minimizations:

$$(p_i - q_i)^2 \rightarrow \min \qquad (q_i - q_{i+1})^2 \rightarrow \min$$

$$q_i = q_i + A(p_i - q_i)$$

$$q_i = q_i + B(q_{i+1} + q_{i-1} - 2q_i)$$

Where

p = original path

q = smoothed path

Repeatedly running the above equations until the different between q_i and q_{i+1} less than a predefine threshold represents the smoothed path has converged. The constants A and B control the degree of filleting. The planned path that smoothened by gradient descent method is suitable to steering robot but not differential driven robot. Since it do not have steered wheels, curved path requires more complex motor control computation than self-turning and straight line moving.

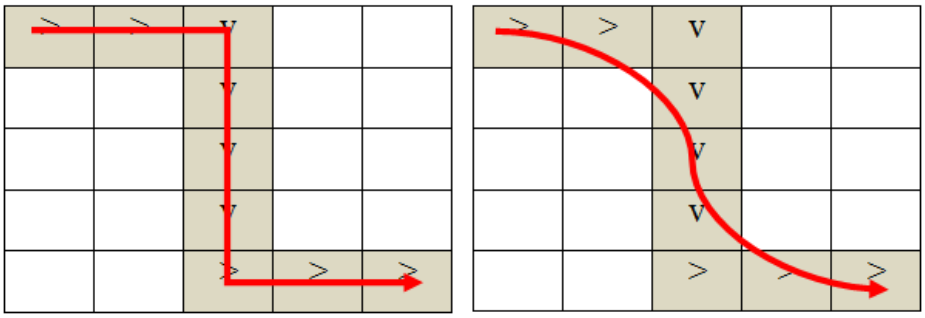


Figure 2-21: Before and After Path Smoother

2.2.4 Obstacle Handling

2.2.4.1 Obstacle Detection

Thrun et al. (2006) autonomous vehicle's obstacle detection is done by detecting vertical points which exist above the detected ground plane. In outdoor environments, terrain slope and the rolling and pitching of the vehicle increases the difficulty of obstacle detection. The obstacles that are higher than the vehicle height are ignored.

The expected difference between3D laser inter-ring distance and range can be computed to solve this issue. In the case for indoor terrain, it is assumed to be flat and the robot is assumed to never roll or pitch.

2.2.5 PID Controller

Proportional, integral and derivative (PID) controller is a classic control system which has been developed for more than 100 years ago. It has been designed for closed loop feedback system to ensure fast respond, low overshot and low steady state error. There are 3 main elements in it:

- Proportional : improves respond time but causes overshot if excessive
- Integral : reduces steady state error but causes windup if excessive
- Derivative : reduces overshot but slow respond time down if excessive

Below is the classic PID equation:

$$u(t) = K_P e(t) + K_I \int_0^t e(\tau) d\Gamma + K_D \frac{d}{dt} e(t)$$

Where:

u(t) = controller's output at time t

e(t) = error between set point and measured value at time t

K_P, K_I, K_D = constant for 3 elements

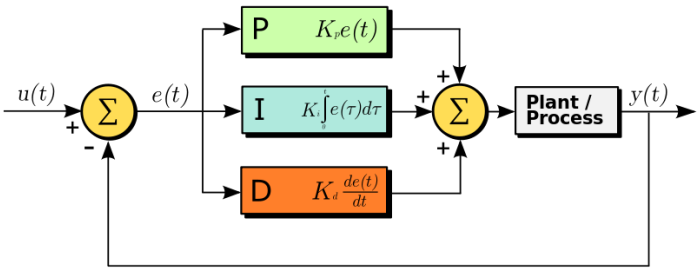


Figure 2-22: PID Block Diagram

2.2.5.1 PID Tuning

There are several PID controller tuning methods available. Manual tuning is the most common method because does not involve any precise calculation but it required experiences and knowledge of each element nature. According to their characteristics,

tune and observe every effect cause by different configuration of each element is the way to tune the PID system manually. The first step is increase Kp until system oscillate constantly and reduce the Kp by half. Then the subsequence steps are to tune Ki and Kd according to the system requirements.

Ziegler-Nichols tuning method is a proven method developed by John G. Ziegler and Nathaniel B. Nichols in the 1940s. It required user to look for the system's ultimate gain, K_u and oscillation frequency, T_u before apply their formulas.

Table 2-1: Ziegler-Nichols Tuning Equations

Control Type	K_p	K_i	K_d
P	$K_u/2$	-	-
PI	$K_u/2.2$	$1.2K_p/T_u$	-
classic PID	$0.60K_u$	$2K_p/T_u$	$K_pT_u/8$
Pessen Integral Rule	$0.7K_u$	$2.5K_p/T_u$	$0.15K_pT_u$
some overshoot	$0.33K_u$	$2K_p/T_u$	$K_pT_u/3$
no overshoot	$0.2K_u$	$2K_p/T_u$	$K_pT_u/3$

Twiddle also called coordinate ascent is a PID algorithm that being introduced to self tuning system. It will try a lot of combinations of the 3 elements until the parameters converged. It starts by initializes all elements as zero, increase the first paramenter by a value dK. The new set of element will be tested and the average error resulted will be calculated and compared with the previous best error. If the new error is better than the best error, increase dK and repeat the above steps. If the error is worse the before, minus instead of add the dK to the element.

If the error does not improves either, reduces the dK and repeats all the above steps until the dK less than a threshold. Therefore, all elements will be fine tune to the threshold automatically.

CHAPTER 3

METHODOLOGY

3.1 Robot Hardware

3.1.1 Robot Driving System

Mobile robot can be constructed to have different driving methods. Tricycle drive or Ackermann steering drive is the most popular method because almost all vehicles on the road using steer drive system (karsten and Ewald 2009). It requires a wheel that able to change its direction of axis of rotation in order to changes the robot's moving heading. In autonomous mobile robot application, steer drive system requires at least a servo motor to control the steer wheel and a motor to propel forward and backward. When doing rotation, it causes 2 different radius of rotation on front and rear wheels. Front wheel or steer wheel will have larger radius than rear wheel. This characteristic increase the complexity of robot's path planning if compares to other drive system.

Omni-drive and Mecanum wheel drive is a different approach on robot driving method. It requires at least 3 driving wheels and do not need any hardware mechanism to control heading direction. Instead, it does not have specific heading direction because it emphasizes the control over the combination of motors' rotation direction and speed so that is can moves in any direction directly. Synchro drive is somehow similar in this characteristic but it rotates all wheels to change its heading direction. Constructing these types of drive system requires more motors and expansive wheels. It does not suitable for budgeted project and low cost robot.

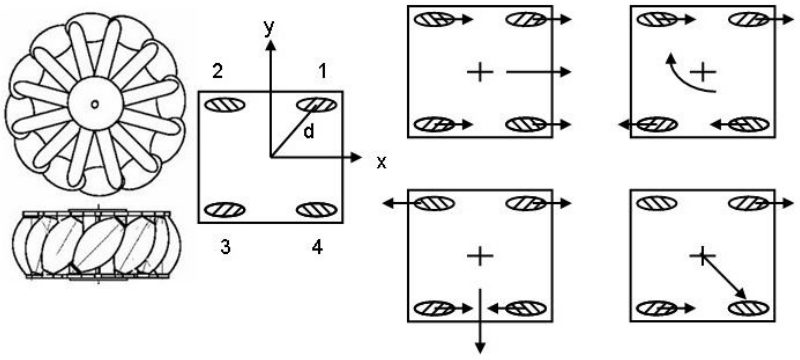


Figure 3-1: Mecanum Wheel and its Drive Method

(karsten and Ewald 2009)

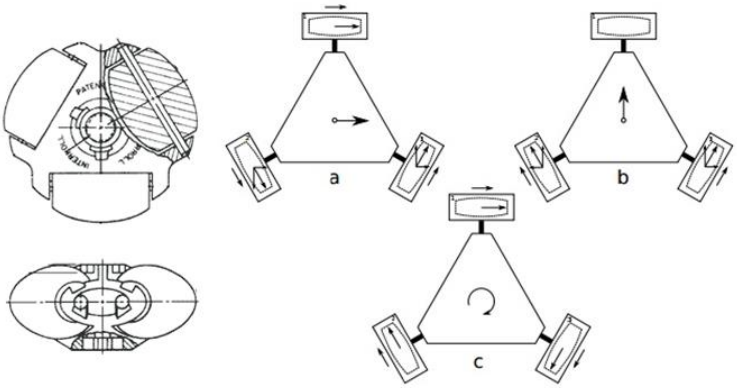


Figure 3-2: Omni-Wheel and its Drive Method

(Wikiversity 2013)

Differential wheeled drive system requires neither servo motor nor special wheel to control its heading direction. Instead it uses only 2 motors to control it heading direction, at the same time propel forward or backward. The degree of rotation is determines by the speed different between 2 wheels. It provides self-turning ability which steer drive system does not and control simplicity that omni-directional drive system does not provides. Moreover, the construction of this driving system requires lesser resources such as motor and wheel. Therefore, it is very suitable for budgeted project or robot.

3.1.2 Embedded System

Robot's actuators require controls from a microcontroller which able to generates low level control signal such as pulse width modulation and digital high-low TTL output. It is also required to handle robot's sensor data which in the form of analog, pulse or different protocol serial inputs since computer cannot generate and receive these levels of signal but serial data through its serial port. The role of microcontroller is to act as manager that managing the operation of robot hardware based on computer command and report back their progress and performance.

3.1.2.1 Arduino

Arduino™ is a very popular embedded system solution. It provides complete designed and preassembled circuit board that includes modules such as microcontroller, power management, IO, communication, memory, and so on. Besides providing hardware, Arduino™ had their software libraries for user that designed specifically to work with the hardware. The tasks of user are reduced to selecting suitable Arduino board and libraries that suitable for their applications. Then, user programs the board using Arduino programming IDE.

Figure 3-3: Arduino™ Logo

(Arduino 2013)

Optimization is always the problem faced when adopting a developed product that designed for general purposes. The final application of Arduino board may not be utilized fully and causing wastages on power, spaces and money. Customized design may require in this situation.

3.1.2.2 PIC Microcontroller

PIC is a series of microcontroller product from Microchip® Technology Inc. It provides a range of microcontroller from 8-bits to 32-bits that suit for different purposes and applications. Besides hardware, Microchip® also provides free microcontroller programming integrated development environment called MPLAB-X and PIC compiler for user to program the microcontroller through a PIC programmer such as PICKIT and ICD.

Figure 3-4: Microchip® Logo

(Microchip 2013)

By selecting a suitable microcontroller and additional integrated circuit such as voltage regulator and voltage comparator, a customized circuit board specialized for application can be produced using smaller board size, lower power consumption and cheaper cost. Hence, PIC microcontroller was selected as the robot's hardware processing unit due to its lower cost, simplicity, flexibility, and customizability.

3.1.3 Robot Sensors

Odometry is an extremely important part for mobile robot, especially autonomous mobile robot. Without odometry module, robot cannot perform autonomous navigation, localization and mapping. Odometry can be carry out be either using encoder sensor or visual sensor.

3.1.3.1 Visual Odometry

By using the same Kinect sensor, visual odometry can be performed by using a open source library called Fovis. Fast odometry from VISion, called Fovis is a visual odometry library developed by VISion that estimate 3D motion of camera that provides RGB and depth data such as calibrated stereo camera and Kinect sensor.

It is different from other visual odometry that using RGB image only to calculates motions. Fovis using not only RGB but also depth images to estimates camera translation in X, Y, and Z axis and camera's rotation in roll, pitch and yaw. By using this visual odometry, the cost for constructing the robot will be reduced because no additional sensors are needed in order to perform robot odometry.

3.1.3.2 Rotary Encoder

Rotary encoder measures rotation angle of its shaft. It is using optical sensors and coded disks to generate pulses. Direction of rotation indicates by the leading or lagging of pulse A and B, while pulse Z indicates a complete rotation. Its shaft can be directly mounted together with rotating motor's shaft to measures motor rotation. The other way is to measures the actual distance travelled of the motor's actuator by adding a wheel to the encoder shaft in contact with the actuator.

The number of pulse per revolution can as high as 600PPR. Hence, it measurement revolution can be very optimism for most of the applications. It is a popular sensor in industry application due to it stability and accuracy and hence, it is categorized as a type of expansive sensor.

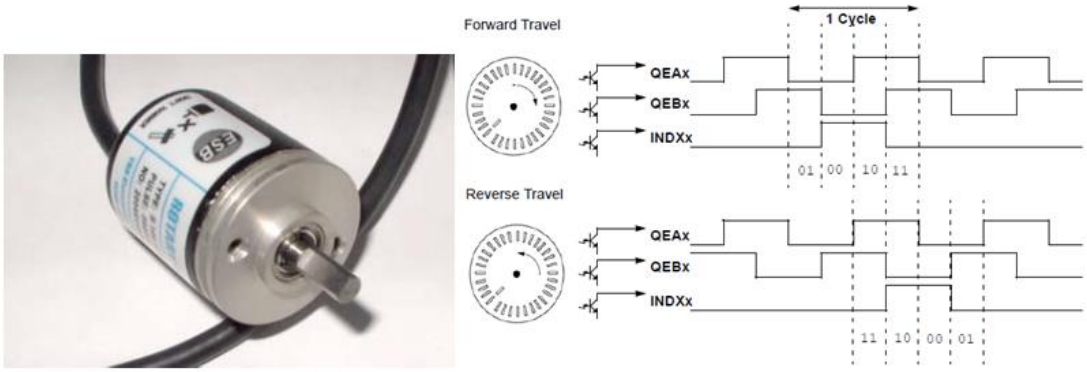


Figure 3-5: ESB Rotary Encoder and its Output Pulses

(Cytron 2013)

3.1.3.3 Optical Encoder

Optical encoder can be built by a pair of Infar-red (IR) emitter and receiver and a coded disk. IR emitter emits IR light bean to the disk and IR receiver activated upon the reflection. The disk normally coded with reflective and non-reflective surface intervene and mounted on the rotating surface.

The sensor measurement's resolution is depends on the number of intervention of different surfaces on the coded disk. Due to the limitation of the size of IR bean emitted, the intervention cannot be smaller than the IR bean size. As the result, the pulse per revolution of this type of encoder is much lower. However, the major advantage of it is its cheap price if compared with rotary encoder.

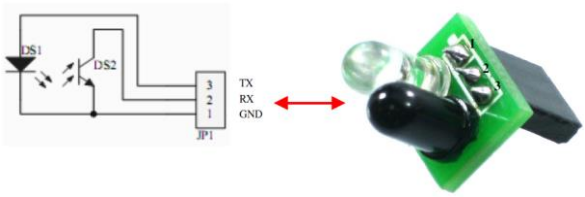


Figure 3-6: Simple Infra-red Emitter and Receiver Pair

(Cytron 2013)

3.1.3.4 Range Sensor

Due to the popularity of Xbox 360 video game console, its accessory Microsoft® Kinect™ is the easiest available range sensor in Malaysia among other 3D range sensor likes Asus® Xtion and PrimeSense® Carmine. Moreover, it having a very competitive price compares to SICK® laser range finder, bumblebee stereo camera or other type of range sensor.

Kinect sensor can be divided into 2 types, Kinect for Xbox and Kinect for Windows. Kinect for Xbox is the designed originally as Xbox game console accessory while Kinect for Windows is designed for computer application development. The one mentioned above is Kinect for Xbox which is cheaper and easier available because most of the Kinect owners use it as video game accessory only. However, it do not granted commercial license for developer to obtain financial benefit in business. On the other hand, Kinect for Windows is more expansive but it does grant commercial license for developer.

They do have some different in hardware such as the new near mode function added into Kinect for Windows. However, Kinect for Xbox can achieve same performance level with Kinect for Windows and yet costing a lower price. Hence, Kinect for Xbox was selected as the range sensor of the robot.

3.2 Robot Software

3.2.1 Kinect Software

Programming tool such as C++, C# and Java cannot handle sensor data directly from its USB port. Kinect driver is needed between sensor and programming tool. There are many open source library available for Kinect developer such as OpenKinect, OpenNI, Kinect for Windows, Point Cloud Library (PCL), and the Robotic Operation System (ROS).

3.2.1.1 Libfreenect

Libfreenect is open source library developed by OpenKinect community. This library provides more Kinect sensor drivers than OpenNI on the three main platforms, Windows, Linux and Mac operating system. It is easier to install Libfreenect in Linux platform because it was primarily developed in Linux, sub-sequencely to Windows and Mac operating system.

Although it cannot access Kinect's audio but compares to other driver, it provides more access to Kinect's:

- RGB Image
- Depth Image
- Servo Motor
- Accelerometer
- Indication LED

Figure 3-7: OpenKinect Logo

(Openkinect 2013)

3.2.1.2 Kinect for Windows SDK

Kinect™ was originated from Microsoft® but the first Kinect open source library was not published by her mother. Microsoft's Kinect library was a closed source but eventually opened because dozen of libraries had developed and published by device hacker communities few months after the release of Microsoft® Kinect™.

Kinect for Windows SDK provides complete hardware access drivers. It not only provides access to data streaming but options and setting for each type data stream and hence, achieves higher level of control over Kinect sensor. In the example

of camera setting, it provides the ability to control RGB camera's contrast, white balance, Gamma, exposure and more setting that other driver do not provides.

Besides basic drivers, Kinect for Windows SDK also provides programming toolkits for developer on application developments. Human interaction is a main development direction of this SDK. Gesture & voice recognition, hand-pointer, face tracking and many other dimensions are the functions that demonstrating this SDK's human comprehension ability.

Since this SDK is developed by Microsoft, the operating system running the SDK must be Windows. This weakens its compatibility from other Kinect software which demonstrates better cross platform capability. Besides, this SDK is full with human interaction functions but it is lacking with 3D space modelling and manipulation functions. This characteristic repels developer who design robot with purpose of navigation through spaces.

Figure 3-8: Kinect for Windows Logo

(Kinect for Windows 2013)

3.2.1.3 OpenNI

Open Natural Interaction, called OpenNI framework is an open source software development kit that facilitates developer in creating middleware library and application for natural and organic interactive device such as Microsoft Kinect, Asus Xtion and PrimeSense Carmine. Together with PrimeScene Nite middleware, it provides sensor driver to acquire data from sensor. It able to access sensor's RGB and depth camera and hence generating image streams to middleware libraries or applications. Besides, its application programming interface able to provide skeleton

and posture detection, image recording and playback based on the images streams read from the registered sensor.

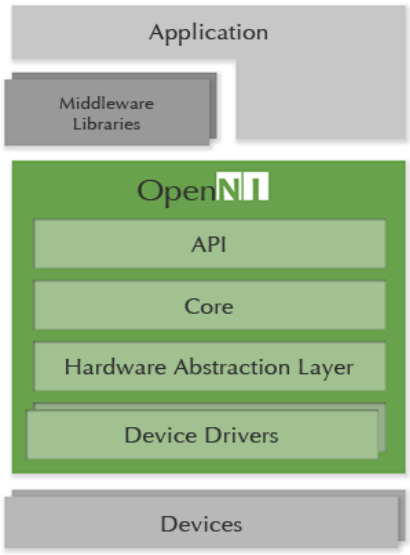


Figure 3-9: Role of OpenNI between Device and Application

(OpenNI 2013)

In the case of autonomous mobile robot this paper concern, OpenNI API will be enough to acquires RGB and depth image streams only. Control over Kinect's servo motor and accelerometer data are additional functions that robot does not need. Driver flexibility over numerous type of sensor is another advantage for OpenNI. Besides, it is a cross platform driver which able to run in different computer OS and hence provides flexibility for developer in platform selection.

3.2.1.4 Point Cloud Library

Point Cloud Library (PCL) is a large scale standalone open source project that contains numerous algorithms and functions for image and point cloud processing. It do not contains specific driver for sensor likes OpenNI does. Instead, it requires sensor driver API to acquire data from sensor before its library can be deployed for

processing. In other word, PCL is a third party library that processes the sensor data for application purposes.

It contains many aspect of algorithm that had been developed contributed by its researcher community around the world. This PCL framework contains many state-of-the art algorithm including segmentation, clustering, filtering, registration, feature estimation and so on. All these functions are processed over a type of data format called point cloud. It is the output of 3D realization process with an input of 2D depth image.

Opposites from Kinect for Windows SDK, this library research direction is more toward 3D space modelling and manipulating instead of human comprehension. Besides, it is supported in the 3 main operation systems and this cross platform ability ease developer in their application development. Hence, it is a better choice for robot navigation application that neglects human comprehension.

Figure 3-10: Point Clout Library Logo

(Pointclouds 2013)

3.2.2 Programming Tools

3.2.2.1 Operating System

Microsoft Windows is a well-developed operating system (OS) for computer, web server, mobile and other devices. It has the highest percentage of market share in computer OS used. Almost all ordinary computer user use Windows as their

computer's OS due to its intuitiveness and user friendliness. However, it does not favoured by some software developers due to the limitations of Windows OS.

Cost is one of the important reasons. Windows is not a free open source OS. Users have to pay for the new OS every time it gets obsoleted by its new versions. Besides, there are times of payments for its software subsequently such as Microsoft Office, Visual Studio, antivirus, and so on.

Linux is a free open source OS. It has preinstalled office software suite, software development IDE and other free software and no antivirus is needed because it is free from virus. It has free long term support and free upgrade to any versions of it. However, machine's driver is a common problem faced by Linux user when they install this OS in new machine for the first time.

Customizability is greatest advantage Linux over Windows OS. Linux OS is virtually limitless customizability. The amount computer resources needed to run Linux OS is much lowers the Windows. Software developed in Linux is able to customize by user so that it run in more efficient manner. Windows uses pre-customized manner that able to suit majority user and normal application but sacrificing it efficiency.

Linux popularity in robotic software development can be observed. Robotic Operating System (ROS) is a library that helps robotic developer in robot software development. It was first sourly developed in and for Ubuntu OS. The other OS such as Windows and OS X are for experimental only. Visual odometry library, Fovis is also a software that only developed for Linux platform.

3.2.2.2 Cmake

Cmake is an open source, cross platform build system that designed to help user to build, test and package software. It is helpful in configuring software linked files and compilation process especially project that includes multiple libraries. By using a

cmake file that contains project specifications, Cmake is able generates native makefiles and workspace for respective IDE.

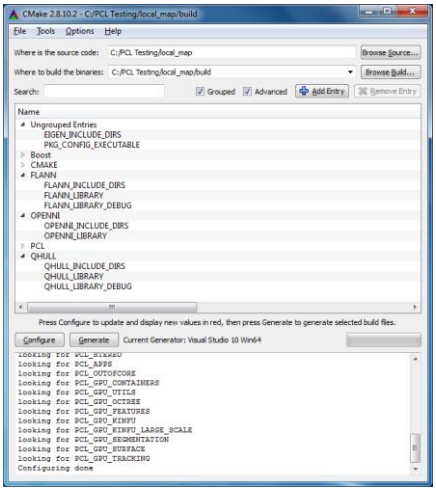


Figure 3-11: Cmake Graphical User Interface

In Linux, source code and cmake file can be written in basic text editor and then compile using ‘cmake’ and ‘make ’commands in terminal or command line interface (CLI). CLI will compile the source code, related header file and generates error if there is any. An executable file of the software is generated if a successful builds with no error and run the program by using ‘./program_name’ command.

An alternative way to program in Linux is to use a developed IDE such as NetBean and Eclipse. They provide a better coding interface and many helpful functions that basic text editor does not have such as pre-compile error detection, debugging, user interface generation and so on.

3.3 Project Job Scope

3.3.1 Hardware design and Fabrication

Autodesk® Inventor® is the 3D computer aided design software implemented in robot’s mechanical hardware structure design. Professional student version of

Inventor® is available for free download and use. It can accelerates the designing process by providing functions such as visualization, measurement, part creation, parts assembly, standard part selection and so on.

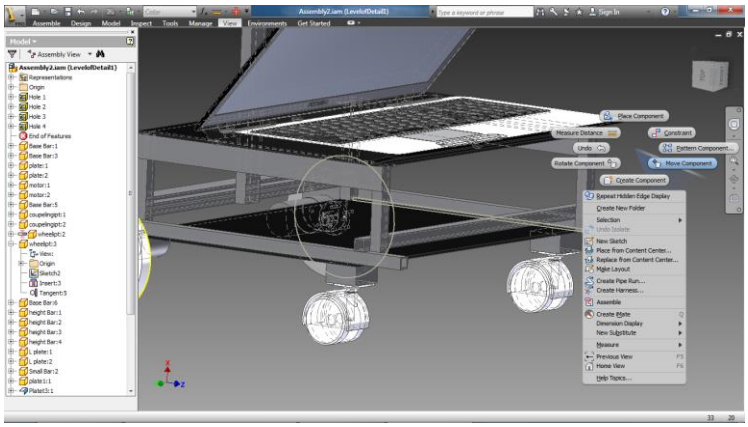


Figure 3-12: Autodesk Inventor User Interface

The design criteria are developed such the way that it must be able to fit in all components onto the structure in correct position and secured holding. No parts and components shell vibrate or moved while the robot moving. Moreover, it must be able moving forward, backward, left turn and right turn with all loads added on-board. The errors caused in movements by the robot shell stay in an acceptable range where it is able to be compensated and corrected by control system during run time.

3.3.2 Hardware Control

For the purpose of navigation, the differential wheeled drive robot only has 2 DC motors to be controlled in this case. Besides output, robot also equipped with sensors that encode and measure the robot motions and feeding data to robot's control system. Therefore, the microcontroller system must be designed to meet these requirements. It must be able to control the motor through motor driver which require control signal such as digital IO and PWM. Moreover, the circuit of board of microcontroller must be able to receive and process encoder sensors data which are in pulse form.

The microcontroller circuit board design can be done by using EAGLE, a printed circuit board (PCB) design software which developed by CadSoft. It enables users to draw schematics by providing a bunch of electronics components libraries. After the schematic is done, user can arranges the components in layout to form the actual PCB. Besides, Eagle provides advances error checking function such as electrical rule check (ERC), design rule check (DRC), and autorouter that helps user to accelerates and identify the imperfections in their design process.

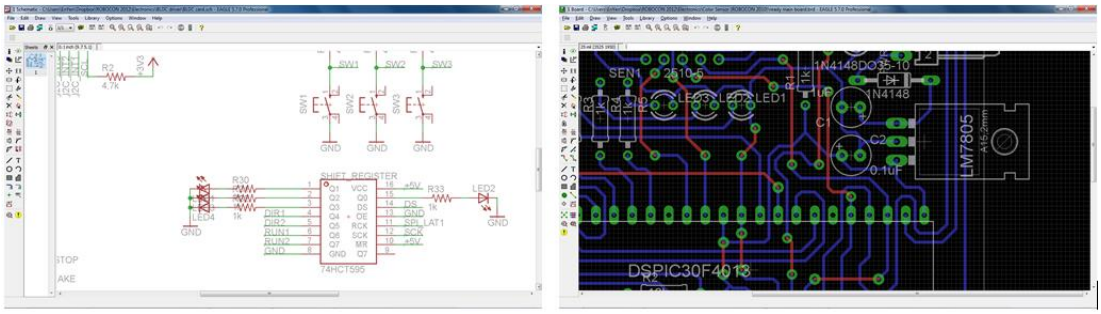


Figure 3-13: Eagle Circuit and Layout Design Interface

PCB then need to be fabricated, soldered and electrical connectivity tested before program can be written into the microcontroller. The microcontroller program is developed in the MPLAB-X IDE and XC-16 compiler provided by Microchip®. It first needs to have a communication module that able to communicate with computer through serial port. Instructions are received at this module and distributed to other respective module.

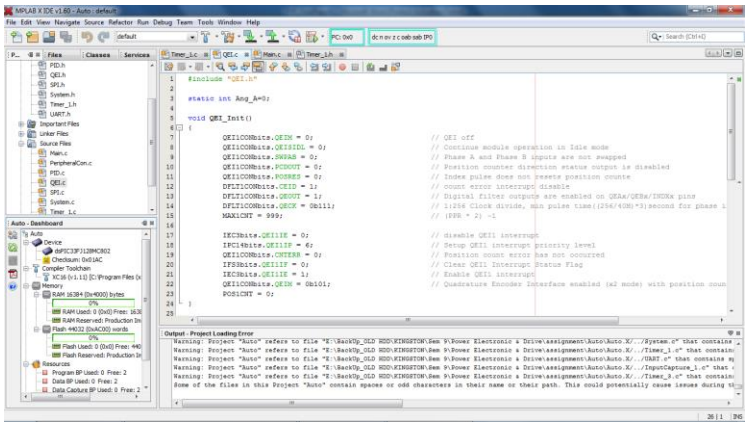


Figure 3-14: MPLAB-X Programming IDE

A timer is required to evoke the control system module which needs to be run in a constant time interval. The control system will first take in encoder sensor data that measures the robot's motion from sensor interface module. Next, error is calculated based on the input command's desired value and the sensor value. A compensator should generate the respective DC motor speed based on the error and its gain. Lastly, PWM and digital IO modules will generate DC motor driver control signals to drive the robot in controlled manner.

3.3.3 Kinect Programming

Kinect programming is the toughest and the most challenging part in this project. Object oriental programming (OOP) is a popular programming paradigm in field of computer software and all Kinect libraries had it developed over OOP manner. However, OOP is an unfamiliar method of programming for author if compared with C or C++ in procedural programming. Therefore, the basic of OOP must be picked and brushed up in order to proceed with those Kinect libraries.

The next task is to explore, learn and get familiar with the libraries used. Trying out those libraries with Kinect sensor directly is required in order to understand it thoroughly in term of its functionally, guiding theory, purposes, effects and the parameters required.

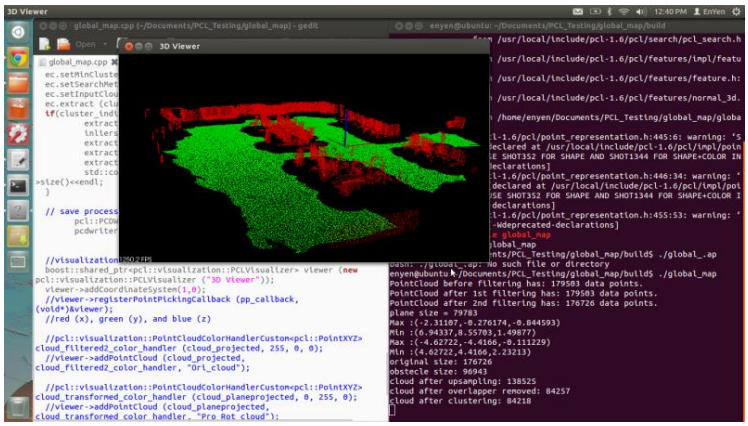


Figure 3-15: Text Editor and Command Line Interface Programming in Ubuntu Platform

There are many modules of libraries PCL provides. For instance noise filters, points clustering, segmentation, registration, transformation, and other useful and related modules to the project.

After knowing the capability of the libraries, robot's software architecture shall be developed. It starts with the abilities or functions of the robot, complexity increases when number of robot's functions increases. A flow of the routines shall be drafted based on the designed robot's functions. Then, suitable libraries or methods are chosen and allocated in respective routine. The connections between routines and modules are the last challenge for the software architecture.

3.3.4 Robot's Intelligent

Robot is required to perform several tasks that fulfilling the project needs. The first task is to perform localization within indoor environment. There are several types of localization method proposed but not all are applicable on this robot and indoor situation. The requirement of the adopted localization method must meet the available sensors' and hardware capabilities such as the field of view of the range sensor, resolution of the encoder, and the processing power of the on-board computer. A localization method will be decided based on its complexity and suitability when applied in this application.

The concept of path planning is not difficult or even easy in understanding it. However, when putting concept onto real application, it requires time to fine-tune until it suit the indoor environment application. Besides, path smoothing algorithm also needed to be modified so that it suits the robot's driving system. In obstacle handling part, robot have to distinguish either the obstacle is a static and dynamic object. After that, specific algorithm shall be used to on respective type of obstacle and making sure the respond time to obstacle is as fast as possible or in an acceptable range to keep robot away from collisions.

3.4 Project Management

There are total 47 weeks for this project, starts from 4th June 2012 until 22nd April 2013. The tasks descried in section 3.3 are well distributed among these 47 weeks. The time line of this project is divided into four major quarters. Each quarter indicates an important milestone and achievement for the project.

The first quarter is mainly focusing on literature review and robot hardware. Start with designing the robot's hardware structure, and follow by the fabrication and assembly of each designed parts. Next, the design and fabrication of robot's electronics takes part. It includes the design of power distribution in the robot and the design of microcontroller and sensors PCB. The last task in this section is the combination and testing of the robot overall hardware functionality. By completing this quarter, a workable differential wheeled drive robot is produced and it can be controlled manually by sending command to its on-board microcontroller.

Second quarter in the project time line is focusing on the study and research on the Kinects' libraries. This task alone had occupied the entire quarter of time line because it is extremely important and its successfulness will rank the overall quality of this project. Besides, well understanding on the libraries facilitates and smoothen the process of system design in the next quarter.

The third quarter contains all software designing part. Based on the available libraries that had been studied in second quarter, module such as localization and path planning are being developed in this section. Moreover, these modules are needed to be linked together in a constructive manner. At the end of this quarter, a robot's software with ability to navigate in indoor environment shall be presented.

The last quarter of the project's time line contains the developments of linkage between software and hardware. Upon the completion of the linkage, a complete robot is considered done. The task after that is to have the robot tested and debug if there is any defect in the system. The performances of robot are recorded to documentation purposes. After all, the last task of this quarter is to produces a report of the robot that documented all parts of the robot and other related information.

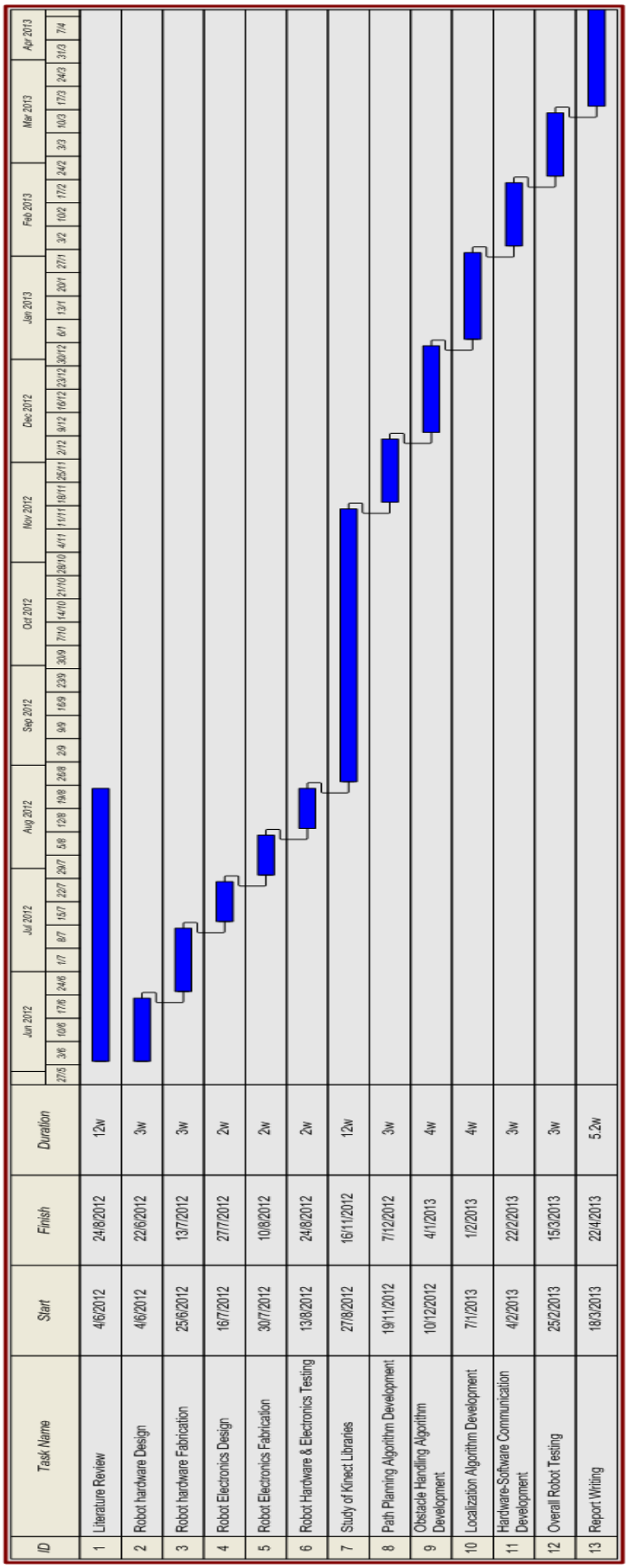


Figure 3-16: Project Gantt Chart

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Global Map

To reduce the required computational power by the robot, it does not perform mapping of the environment during navigation. In other words, it neither keep track of the path is had travelled nor produces a map of the environment at the end. This significantly reduces on-board computational requirements the advantage becomes more obvious when the environment's size growing larger.

However, mobile robots still need a map in order to navigate. Hence, before the robot can be deployed, it first needs a full and complete map of the environment, called the global map. This map contains all the robot navigable locations in the environment and is pre-processed to occupy minimal memory space. The acquisition and processes of global map are carry-out manually and separately from the robot navigation architecture.

4.1.1 Global Map Acquisition

The robot working environment can first be scanned by Kinect compatible 3D reconstruction software such as Scenect, ReconstructMe or Skanect to produce a raw scanned 3D point cloud. There are some limitations on these software such as the point cloud’s size and trial version water mark. Besides, users are required to move

the sensor slowly and smoothly in environment with sufficient object and color to maintain the tracking performance frame by frame.

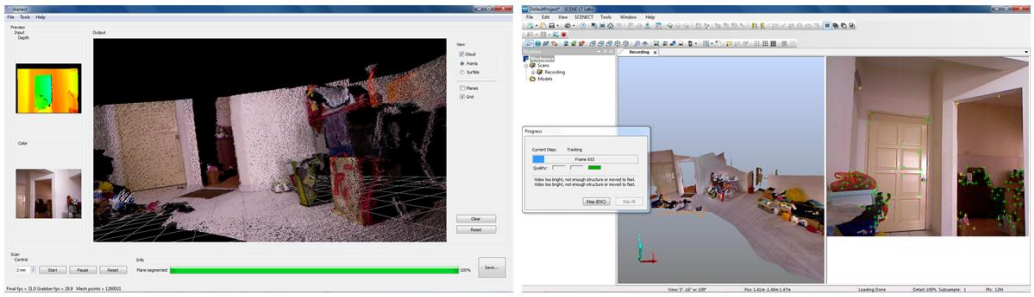


Figure 4-1: Skanect and Scenect 3D Reconstruction Software Interface

Kinect sensor can be hand held while scanning the environment, providing the hand can move in steady and constant speed without jerking. A better approach applied in this project was to place the Kinect sensor on the robot to scans the environment while the robot is manually controlled to slowly move around in the environment. Several scans were carried out due to the size of the environment and the sensor motion's smoothness when scanning the environment.

4.1.2 Global Map Processes

The raw scanned environment point cloud was undergoes a series of processes that specially designed to transforming it into useful data for robot during its navigation.

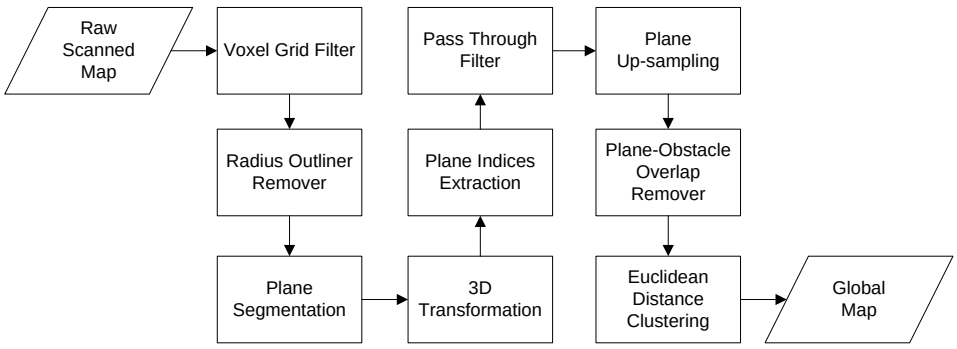


Figure 4-2: Global Map Process Flows

4.1.2.1 Voxel Grid Filter

Raw scanned environment normally contains millions of points depends on its size. Its large file size causes heavy computational load to computer which is not desirable in software optimization. Many points are actually not meaningful because they are just very close to or simply overlapped each other. This filter performs point cloud down-sampling with purpose to reduce the size of point cloud. It removed point cloud's color and using fewer meaningful points to represent multiple adjacent points according to the given density parameter. The density parameter is set according to the needs of the subsequent processes. In this case, the distance between filtered points was set to 0.035 meter and the original global map with 3 million over points had reduced to 179503 points. As the result, down-sampled point cloud with memory size several times smaller than original's is prepared.

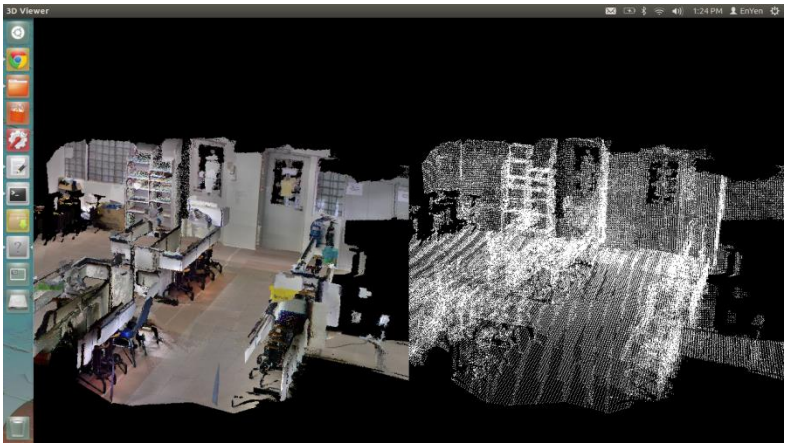


Figure 4-3: Result of Voxel Grid Filter

4.1.2.2 Radius Outliner Remover

Raw scanned point cloud tends to be noisy even after the process of down-sampling. This filter removes points that do not have much close neighbours surrounding it. It counts the number of neighbour a point has in its surrounding with certain radius. The parameters, number of neighbour and neighbour searching radius are set by observing its overall noise remover performance. In this case, point with less than 9

neighbours in its search radius of 0.07 meter is considered as noise and removed from the point cloud. As the result, 2777 noise points had been removed and the point cloud remained 176726 points.

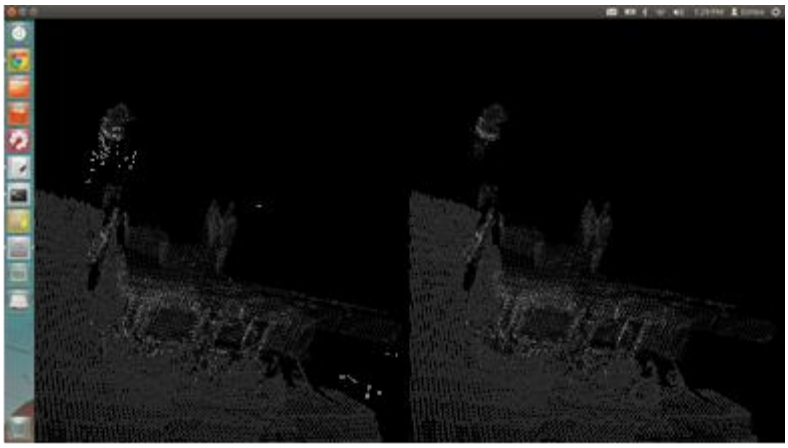


Figure 4-4: Result of Radius Outliner Remover

4.1.2.3 Plane Segmentation

A plane is needed to be detected and extracted from the point cloud. This plane segmentation method use random sample consensus (RANSAC) plane detection method with parameters such as distance threshold, epsilon threshold and maximum iteration to differentiate plane model. Distance threshold is the manipulating parameter and its value is related to the quality of the raw scanned environment point cloud. Low quality scan has planes lay on different plane coefficient and larger threshold is required to include all shifted planes. In this case, distance threshold was set to 0.1 meter and epsilon angle of 25°. As the result, plane model was detected and differenced from the point cloud.

4.1.2.4 3D Transformation

Since the raw scanned point cloud can be produced in free hand holding method, its orientation in coordinate system can only be detected after plane segmentation. Using the output plane's coefficient produced by the planar segmentation method, a transformation matrix is calculated based on the detected plane's coefficient with respected to the global XY-plane. This transformation matrix transforms the entire environment point cloud with respect to the global coordinate system. It aligns the global map's plane/floor to XY-plane. This step provides the global map a capability on 2D map's position processing using only point's X and Y coordinates. Besides, it also facilitates the map's height processing by using only point's Z coordinate.

4.1.2.5 Plane Indices Extraction

This function separates the environment point cloud into two, plane and obstacle point cloud. It saves the indices points that have been recorded in planar segmentation as a plane point cloud and saves the rest as obstacle into another point cloud. This is an effort to facilitate the subsequence processes which requires separate processing for plane and obstacle point cloud.

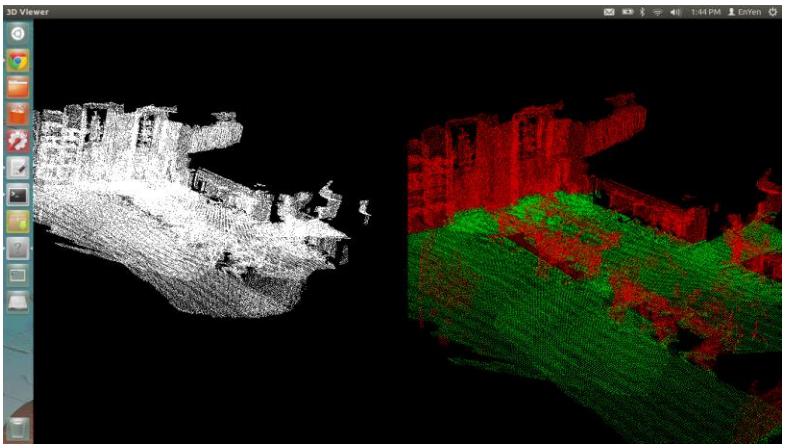


Figure 4-5: Result of Plane Segmentation, 3D Transformation and Plane Indices Extraction

4.1.2.6 Pass Through Filter

At this stage, plane and obstacle point clouds are both ready for processes that based on global coordinate system. Pass through filter is used to remove obstacle point that located outside axis rejection boundaries. By setting the Z axis limits between zero and H, where H is the robot's height with additional tolerances, obstacles or part of obstacles which higher than H or lower than zero are removed. Since the field of interest in detecting obstacle is limited up to robot's height, obstacle which higher than robot's height is considered not an obstacle. In this case, H was set as 0.45 meter given the actual robot height is 0.28 meter and as the result, obstacle point cloud is reduced and limited to robot's obstacle detection region.

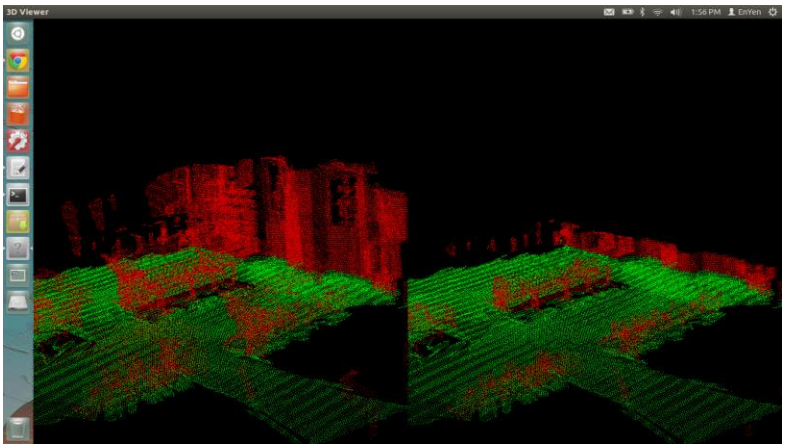


Figure 4-6: Result of Pass Through Filter in Z Axis

4.1.2.7 Plane Up-sampling

The purpose of up-sampling is to standardize plane point cloud density. Raw scanned point cloud normally does not have evenly distributed point density, especially for large environment which had multiple scanned. By up-sampling the plane point cloud, the plane resolution increases and obtains equal density at all regions. This point cloud's quality improvement process facilitates the subsequence processes which require constant inter-point distances in plane point cloud. In this case, the parameter is set so that plane contains 70 points in a circle with 0.05 meter.

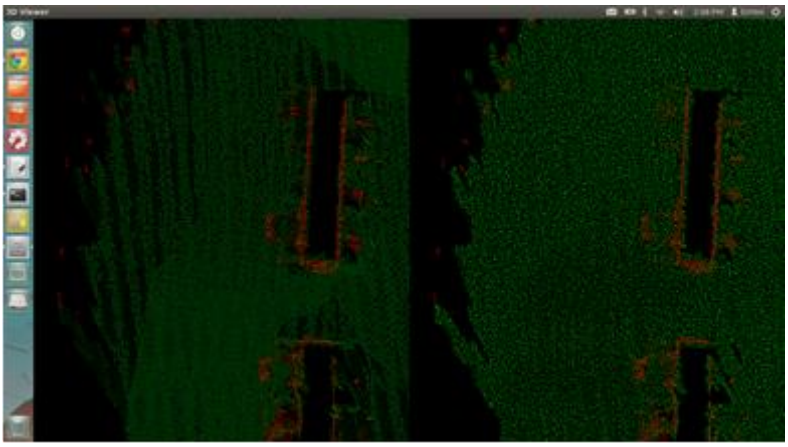


Figure 4-7: Result of Up-Sampling

4.1.2.8 Plane-Obstacle Overlap Remover

Both plane and obstacle point clouds are projected onto XY-plane at the beginning of this process to ensure that they do not have difference in Z axis. Then, radius search is carried-out on every plane's points. The purpose of this radius search is to remove plane's point which having obstacle point located within its circular radius R, given R is the maximum radius length of robot with additional tolerances. This remover designed to reduce plane's size by eliminating point that having obstacle above or even nears it. In this case, R is set as 0.27 meter given the robot dimension is 0.4 meter by 0.35 meter and as the result, the remaining plane point cloud indicates robot navigable location within the environment.

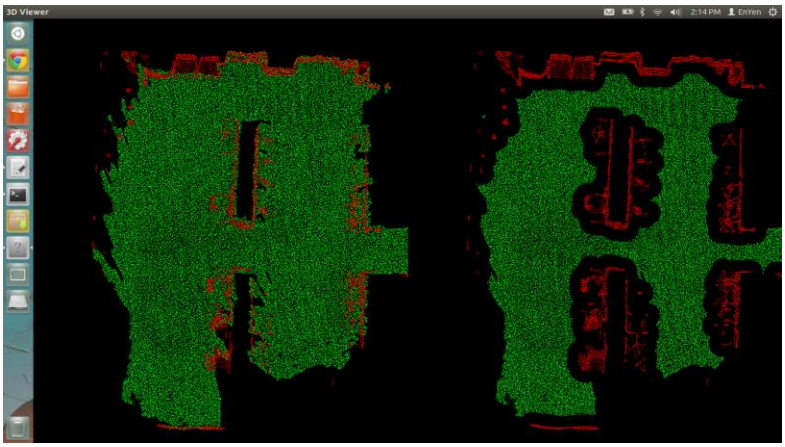


Figure 4-8: Result of Plane-Obstacle Overlap Remover

4.1.2.9 Euclidean Distance Clustering

The last step of these series of processes is to perform Euclidean distance clustering on plane point cloud. This clustering method groups points into different groups based on several parameters such as cluster distance and minimum-maximum cluster size. The purpose of this clustering is to remove plane that discontinued from that main plane. By remains the largest plane cluster and discard the rest, a complete robot navigable map, called global map is produced as the final output of these series of processes. This map in this case contains 84218 which considered smaller memory size than the raw scanned map which contains 3 million over pints and it provides much more meaningful information for robot during its navigation.

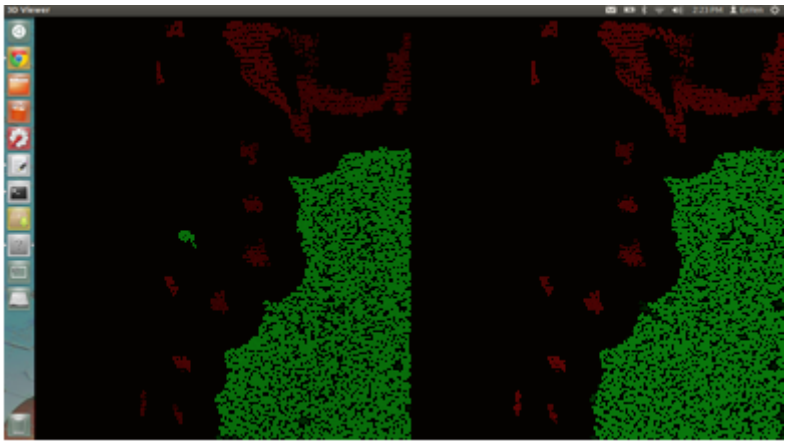


Figure 4-9: Result of Euclidean Distance Clustering

Figure 4-10: Comparison between Raw Scanned and Processed Global Map

4.2 Particle Filter

After the global is prepared, robot is now able to use it in path planning process. The purpose of this specially designed filter is to find out the robot's position and heading direction in its working environment before robot starts its navigations. Without this starting location's information, robot cannot perform path searching and it reduces robot's localization's intelligent because robot might need to start at a same point every time it starts its navigation.

Particles are created on the global map at the beginning. Each particle is assigned a random coordinate and heading and act as a possible robot's location and heading direction. They are initially planned to be evenly distributed throughout global map with a density that depends on the robot's size and the desired localization's resolution. The denser the initial particles are, the higher the resolution of the localization is. However, large number of particle will cause heavy computational load to the robot's on-board computer which having limited computing resources.

Hence, a starting zone with size multiple times smaller than global map is created for robot to start navigates. This starting zone can be assigned as the part of the global map such as living room in house scenario and kitchen in restaurant scenario instead of the entire environment. The purpose of this solution is to effectively reduce the number of tracking particle created. As the result, this method compromised robot's overall localization's intelligent due to the limited localization region, but effectively improves software efficiency.

4.2.1 Filter's Phases

Right after the particles were created, particles entered an update phase. Their distances to the edge of the global map in six directions are recorded. These distances are calculated by searching the present of global map's point in the specific direction and distance. The searching path is increased until no global map's point is found in

the specific search radius. This process is carried-out on all particle's six sixty degree intervene direction with respected to the particle heading.

In measurement phase, robot with unknown location and heading will self-turn to six direction with sixty degree each intervene and measures the distance of first obstacle encountered with width of 6 centimetre. The obstacle might be wall that had recorded in global map or new obstacles which never been recorded. Taking six reading or more in different directions is to reduce the disturbances bought by new obstacle in measuring the range of surrounding environment. Both particles and robot range data are now ready for the next process.

Important weight of every particle is being calculated using Gaussian probability. It takes particle range data, robot range data, and sensor measurement noise to determine how likely this particle located near to the actual robot location. This process assigns higher weight to particle if its measurements have high degree of similarity to robots'. The Gaussian probability equation (4.1) below calculates the probability of sensor range data for 1 dimension Gaussian with mean of particle range data and sensor measurement noise.

$$Gaussian\ Probability = \frac{e^{\frac{(\mu - x)^2}{2 * \sigma^2}}}{\sqrt{2\pi\sigma^2}} \tag{4.1}$$

Where

μ = particle range data

x = robot range data

σ = sensor measurement noise

Particles with heavier weight are most likely to be survived after a resampling process. This resampling process mean to removes particle with lighter weight which locates further from robot actual location. As the result, particle which locates closer to robot actual position will survive and remain in as heavier particle.

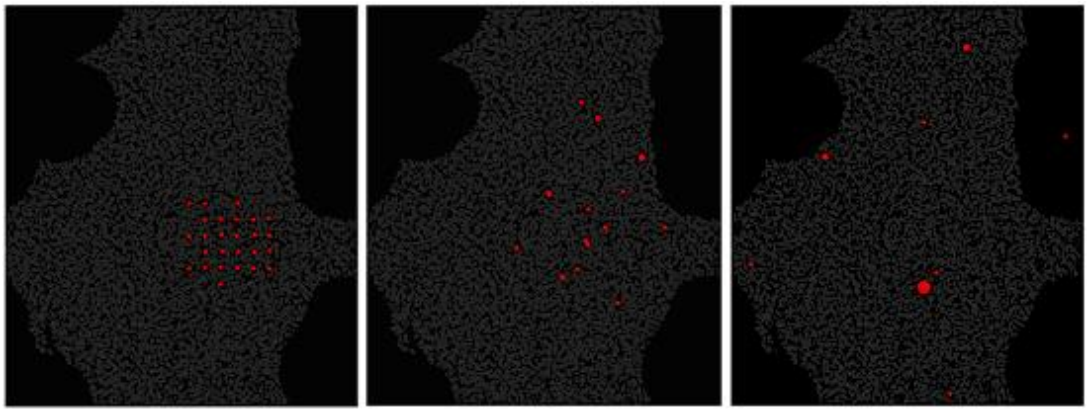


Figure 4-11: Particle filter in Action

If certain particle's weight exceeds 80% of total particles, robot position and heading will be decided using the particle identity. Else if there is no particle having such high value, it indicates that robot measurements are under large new obstacle disturbances or the environment has similar geometry. Robot is required to move forward for 30 centimetres and repeat the above procedures. In this context, particles are also requires to move exactly same distance as robot instructed before entering update phase. The overall filter process is descripted in the flow chart figure 4-12.

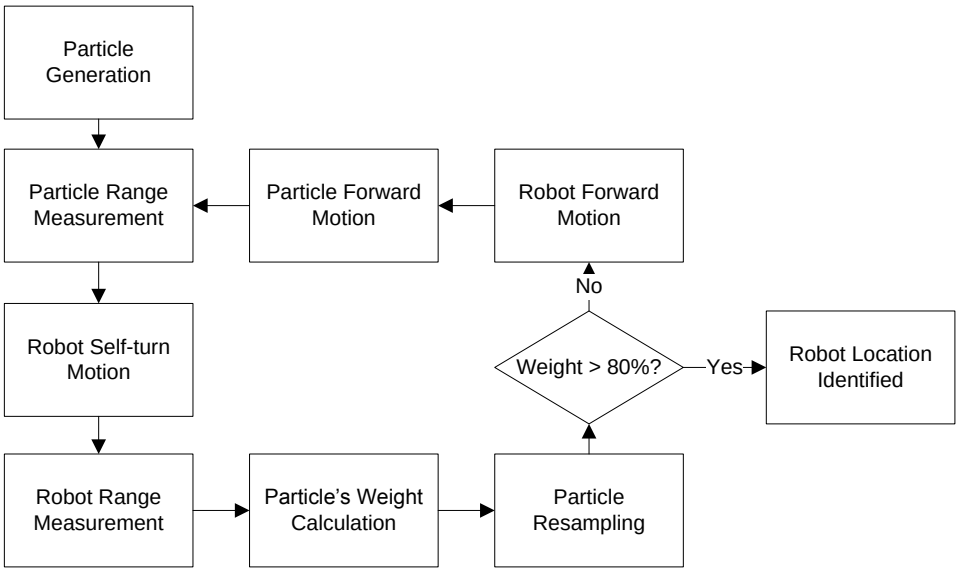


Figure 4-12: Particle Filter Flows

4.2.2 Filter Performance

In successful cases, this filter requires five to six filter loops and takes about 120 seconds to identify robot position. But most of the time its values diverged and fail to localize itself. The percentage of success of this filter is about 20%.

Table 4-1: Particle Filter Experimental Result

Experiments	*Result*	*Time Taken (second)*
1	Success	110
2	Fail	-
3	Diverged	150
4	Diverged	130
5	Fail	-

The poor performance of this filter is caused by the parameter and data it acquires and use in this filter. Taking distance to global map's edges is a good way in particle identification, but distortions do occurs in robot's environment measurement due to new obstacle or small changes in the environment.

4.3 Path Planner

In road mobile vehicle path planning, route network definitions file (RNDF) that indicating the navigable location in an open area is utilized. Indoor mobile robot does not have this pre-acquired, large scale and publicly available vehicle navigable road information. It needs to first acquires a global map and process it in order to identify the navigable region in the environment. Then only it can plan its path on the navigable area indicated in the global map. A planner was specially designed based on the robot specification to plan robot path, allowing the robot to navigate through the environment autonomously.

4.3.1 Path Searching

A* with heuristic is the path searching technique implemented in searching robot navigation's path. The global map is first partitioned into multiple discrete cells. The size of the cells determines the resolution of the path. Smaller cell's size is able to generate smoother path and provides mobile robot the ability in reaching the destination with smaller error.

From the starting point, 8 cells in 8 equally separated directions are explored by a single cell step. The cell's navigability is first being tested by calculating its number of global map's point inside the cell using PCL KdTree search. If it is a navigable cell, it will be stored in a vector. This vector stores all cells that had been explored and every time before the planner explores new cell, it will check this vector if the new cell ever exits and explored before.

A cost is assigned into each cell which is important when determining the next most optimal cell going to be further explored among the eight. The heuristic function used in determining the cell's cost is calculated by the Euclidean distance between the exploring point and end point. The best approach is to use cell distance instead of Euclidean distance as the A* heuristic function as stated in equation (4.2). However, computing heuristic distance is much simpler than cell distance and less time consumption as stated in equation (4.3). Therefore, path searching does not search the best path in some scenarios but the searching process is performed in slightly faster speed. Figure 4-13 demonstrates the effect of using Euclidean distance as heuristic function, the red lines are the non-optimal path search and the green lines are the optimal path.

$$Optimal\ Cost_{cell} = CD_{start,cell} + CD_{cell,end} \tag{4.2}$$

$$Implemented\ Cost_{cell} = CD_{start,cell} + ED_{cell,end} \tag{4.3}$$

Where

CD = Cell distance

ED = Euclidean distance

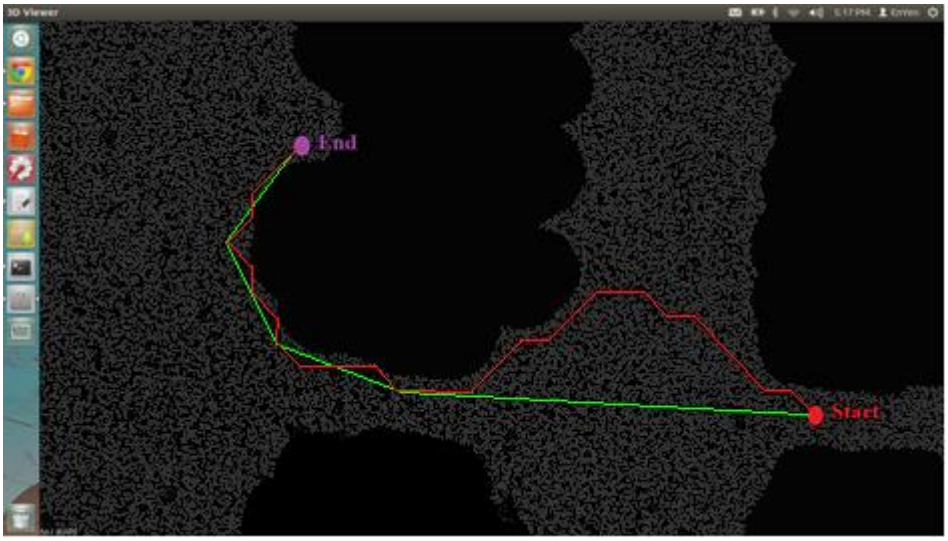


Figure 4-13: Non-Optimized Path Search and Optimized Path Smoother

The searching process will continue until an explored cell has destination point located inside it. By using this method, the maximum steady state error caused is the radius of cell's size. As stated before, smaller cell's size will caused smaller steady state error but time taken in path planning will at the same time increased. This robot has size 40 cm by 40cm and its path planner use 15cm as the cell's size. The maximum steady state error is 7.5cm. This amount of error with respect to the robot's size is acceptable for robot that does not require millimetre accuracy.

4.3.2 Path Smoother

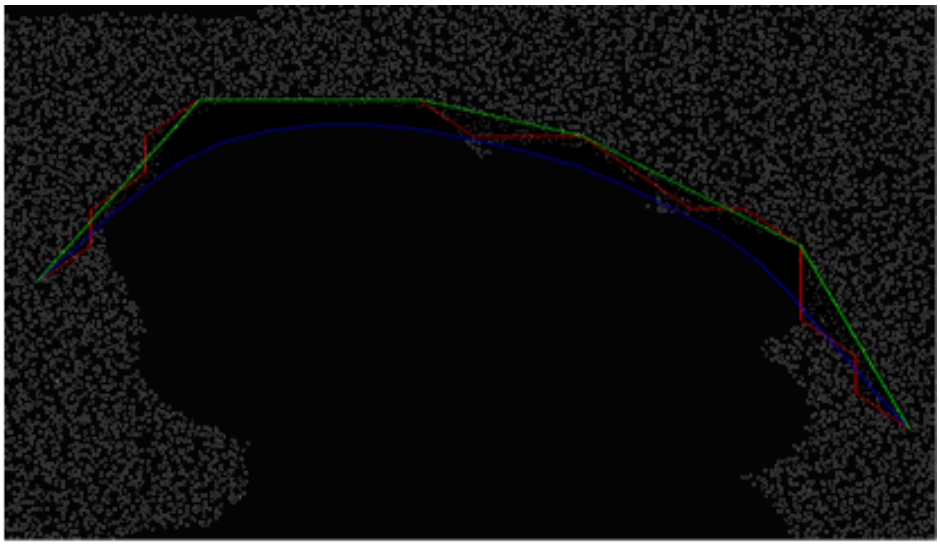
The output of this path search is a vector of point which connects the start and end points by linking them with lines. It is very impracticable for robot to move in that way because it contains too many turning points. Therefore, the path should be optimized after that.

Gradient descent smoother has been tested as one of the path smoother. It output smoothed path in curved-liked form. This type of output is very suitable for steering drive robot but increase driving complexity for differential drive robot.

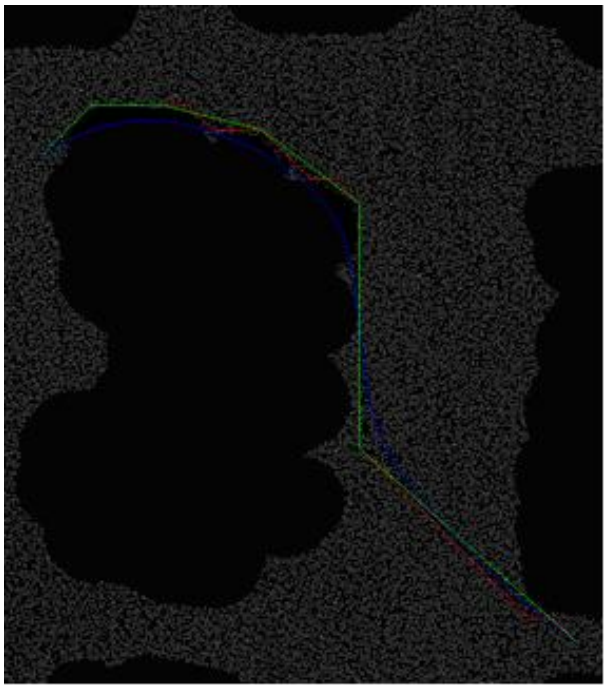
Moreover, this method smoothen path by shifts the path's point and sometimes points are being shifted toward or even on non-navigable region.

A specially designed path smoother this robot implemented checks whether any obstacle exists within the straight line connecting the start and end points. It looks for points that can be bypassed and then connects them up to reduce the number of lines in the path. It uses the same method path planner used in detecting navigable cell. Figure 4-13 above demonstrates how this smoother improves the non-optimal path search. In the figure 4-14 below, the red lines are the path before smoothing and green lines are path after smoothing. The number of turning points the green path has is significantly reduced compared to red path.

This path smoother ensures that none of the path exceeds the navigable map boundary. This is a better approach compared to the gradient descent smoothing method which shifts paths into non-navigable areas. Besides, this smoother output no curved line. Moving in a straight line and self-turning are much easier than moving in an arc path for this robot. Figure 4-14 again shows the comparison between unsmoothed path in red, smoothed path in green and gradient descent smoothed path in blue.



(a)



(b)

Figure 4-14: Comparison of Different Planned Path

4.3.3 Path Planner Performance

The time taken for robot path planning largely depends on the size of the global map, processing unit and the distance between start and goal point. The orientation of obstacles also significantly affects the path process time. Table 4-2 shows an experiment that measures the time taken path planner needs to plan a path. These experiments were conducted using an approximately 80 meter square global map shown in figure 4-10 and a Dell laptop with Intel® i5 2.5GHz processor and 4GB DDR3 ram.

Table 4-2: Result of Path Length against Processing Time

Path Length (meter)	*Processing Time (second)*
3	4
6	10
10	18

4.4 Obstacle Handler

Indoor mobile robot will encounters obstacle such as wall, door, staircase, furniture, human or other robot. All known obstacles which insides global map are already taken into account during path planning. New obstacles are those which never been recorded in global map but exist in local map. It can be detected and responded during robot's navigation using a specially designed obstacle handler.

4.4.1 Obstacle Detection

After the path has been prepared, the robot will start to move to its first point. While it is moving forward, the Kinect sensor captures the depth data (called the local map) in front of the robot with 57 degree by 47 degree field of view.

Instead of checking the entire space captured in front, the robot only checks if any obstacle exists in the given path's direction. A translation in local map's Z axis is done at the beginning. It means to shift the local map backward by robot forward travelled distance. Then, a pass-through filter is used to filter the local map into a rectangular region of interest with dimensions robot width by robot height by path length. It is then checked for any existing points (obstacles) inside this rectangular obstacle detection region. Since the sensor's height on the robot is a known constant, the implementation of plane segmentation to detect floor can be avoided in the local map. Any point below the robot's height can be simply ignored since this robot is not design to work with traverse staircases or steep slopes.

The local map is then transformed to the next point of path, P_{n+1} and heading to P_{n+2} using trigonometry (4.4), (4.5) and transformation equations (4.7). Then it checks again for obstacles points in the rectangular region with length d_2 calculated from equation (4.6). The green lines in figure 4-15 are the robot navigation path and the field of view of the Kinect sensor are transformed to detect obstacle in different locations.

$$a = \tan^{-1}\left(\frac{P_{n,y}-P_{(n+1),y}}{P_{n,x}-P_{(n+1),x}}\right) \quad (4.4)$$

$$d_1 = \sqrt{(P_{n,x} - P_{(n+1),x})^2 + (P_{n,y} - P_{(n+1),y})^2} \quad (4.5)$$

$$d_2 = \sqrt{(P_{(n+1),x} - P_{(n+2),x})^2 + (P_{(n+1),y} - P_{(n+2),y})^2} \quad (4.6)$$

$$Translantion_{FOV} = \begin{bmatrix} 0 \\ 0 \\ -d_1 \end{bmatrix} \qquad Rotation_{FOV} = \begin{bmatrix} 0 \\ a \\ 0 \end{bmatrix} \quad (4.7)$$

Ideally full path obstacle detection can be done immediately as figure 4-15 demonstrated if the whole path is located within the sensor field of view. However, the 5.5 meter maximum depth range limits its ability to achieve that. Moreover, looking too many steps/points forward increases the robot's respond delay since the obstacle detection routine is only performed in the event invoked by a successful depth image obtained.

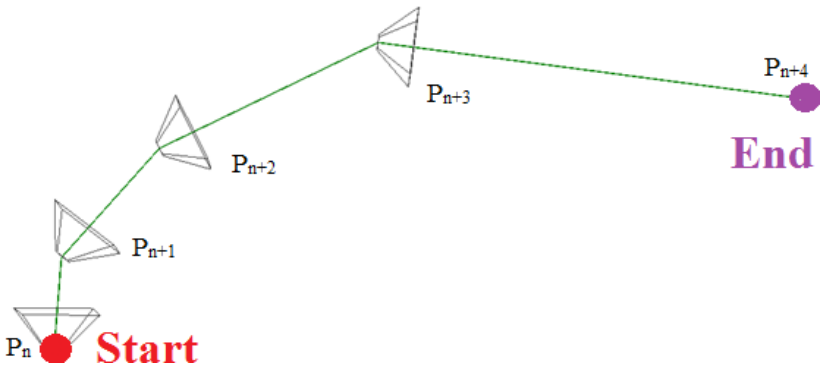


Figure 4-15: Transformation of Kinect Field of View Through-out Entire Path

4.4.2 Obstacle Avoidance

When robot detects any obstacle in its path, it will stop and wait for clearance. This is an effort to distinguish between static and dynamic obstacles. Dynamic obstacle will move away from the robot's path after a few seconds but static obstacles will not.

Static obstacle is assumed and confirmed if clearance does not occur 3 second after robot detected it. If a static obstacle is detected, its size and location are recorded for global map modification process. The obstacle cluster is measured from the local map using Euclidean cluster extraction. It is then transformed to match with the current position of robot referring to the global map using the transformation equation (4.8) and (4.9).

$$Translation_{obstacle} = \begin{bmatrix} robot_x \\ robot_y \\ 0 \end{bmatrix} \tag{4.8}$$

$$Rotation_{obstacle} = \begin{bmatrix} -90° \\ 0 \\ robot_{heading} - 90° \end{bmatrix} \tag{4.9}$$

Since the local map is captured in such the way that the floor is on the XZ-plane with the Z-axis points forward, the obstacle cloud needs to be transformed -90 ° about X-axis so that both obstacle and global map are refer to same floor. Moreover, it also needs to rotate -90 °about Z-axis to align with the robot's heading.

Now the obstacle and global map point cloud are aligned, those points of the global map which have obstacle points above it are removed. The robot is then ready to re-plan its path to reach its destination. Figure 4-16 demonstrate a global map modification and re-planning after a static obstacle found during its navigation.

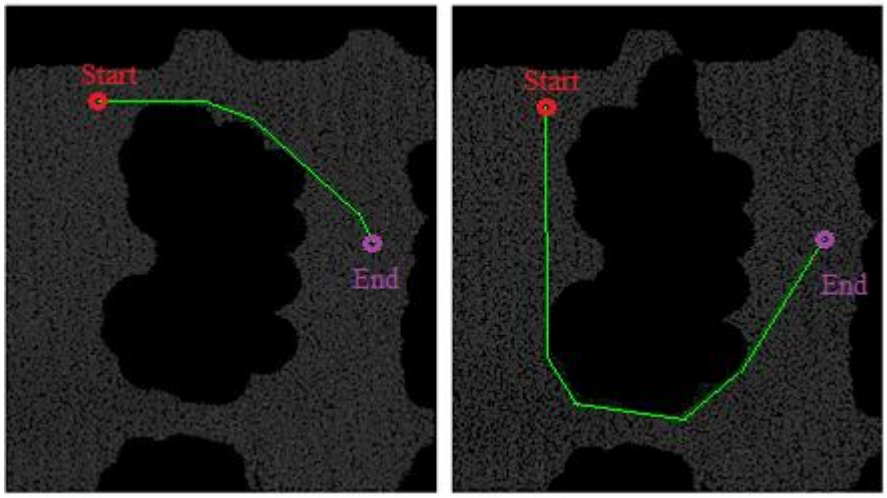


Figure 4-16: Global Map Modification and Path Re-planning

4.4.3 Obstacle Handler Performance

New obstacles in path can be identified and reacted to in less than a second. This respond time is acceptable since both robot and dynamic obstacle are not moving in high speed in indoor environment. However, this value can be altered depending on the number of turning points exist in path and the maximum range for detecting obstacles within the local map. Experiments were carried-out to test the time taken for motors to stop after appearance of obstacle.

Table 4-3: Robot's Respond Time to Obstacle

Maximum Detection Distance (meter)	***Number of Turning Point***	***Respond Time (second)***
3	3	0.5
3	6	0.8
5	4	0.5
5	8	1.0

4.5 Motor Control

Once the robot's smoothed navigation path is prepared, robot has to instruct the hardware to perform motion tasks. Robot's on-board computer controls the robot hardware by sending control signal to the microcontroller through USB serial port. The instructions includes move forward, left self-turn, right self-turn, pause motion, resume motion and travelled distance request.

Self-turn instruction is launched when there is a discrepancy between current robot's heading and the next heading direction of the planned path. After a match is achieved in both heading direction, forward motion instruction is launched to instruct a forward motion. Travelled distance requests are also sent during forward motion to acquire forward travelled distance which needs in obstacle detection module.

When robot is in forward motion, obstacle detection module is invoked to discover the present of new obstacle in robot planned path. When an obstacle is found, a pause motion instruction is launched immediately to avoid collision. Resume motion instruction is launched when obstacle is removed and robot's path is clear to navigate. These motions instructions will keeps sending to control the hardware until robot reached its destination.

4.6 Visual Odometry

To reduce the number of sensors used and the overall cost, robot position measurement is carried out without any hardware sensor at the beginning. Instead, 3D visual odometry using FOVIS was implemented to measure robot motion. XYZ translation, roll, pitch and yaw can be determined using the Kinect sensor. It utilizes RGB and depth images to measures robot's motions during the robot's self-turning and forward motion. The overall software architecture of the robot with visual odometry as the motion sensor is shown in figure 4-17.

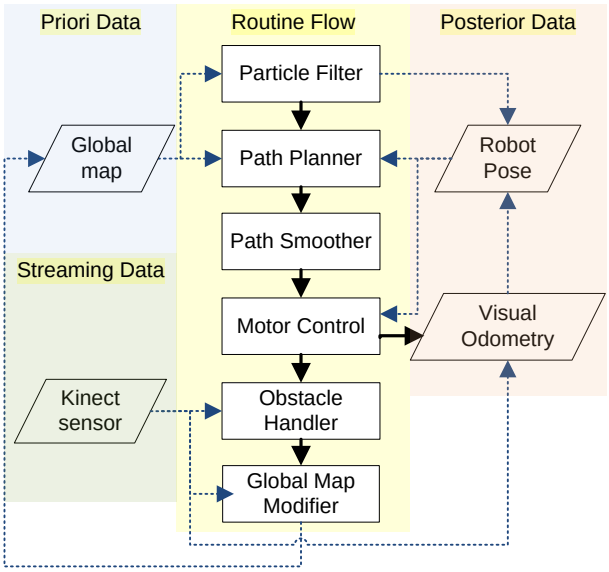
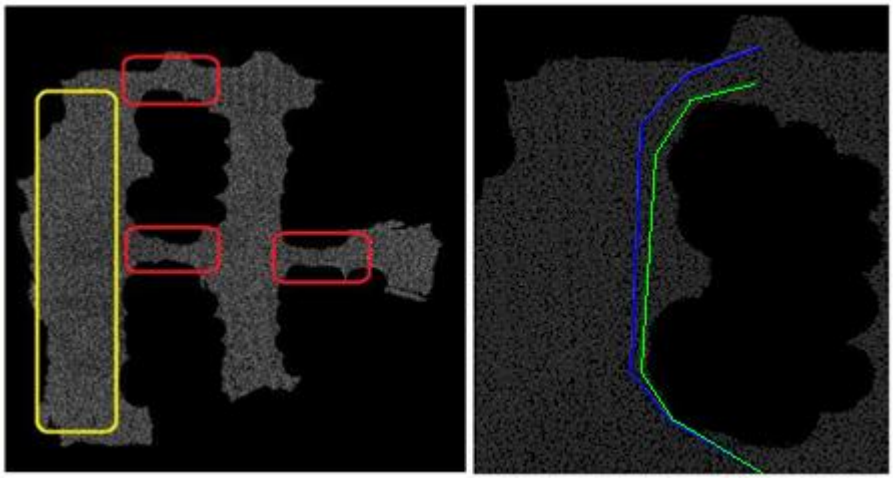


Figure 4-17: Overall Software Architecture with Visual Odometry

Therefore, the mounting accuracy of Kinect sensor on robot is very crucial in order to maintain the motion measurement's accuracy. The centre of the wheels' rotation axis is aligned right below the visual centre of the sensor as shown in figure 4-22. Moreover, the sensor's tilt must be parallel with the floor plane so that the local map does not need to be further transformed before processing.

4.6.1 Visual Odometry Performance

The implemented visual odometry does not measure rotation accurately in confined spaces and places filled with plain surfaces obstacle. Besides, linear movement in a long and empty space also causes inaccuracy in measurement of forward travelled distances. Figure 4-18(a) red rectangles shown the region in the global map where Fovis cannot measures rotation well and yellow rectangle for linear movements.



(a) (b)

Figure 4-18: Week Performed Region (a) and Planned Compared to Actual Travelled Path (b)

As the result the robot loses track of the distance and angle travelled. However, the robot was still able to reach the target area safely with discrepancies. Figure 4-18(b) showed an example of robot test run where the planned path is shown in green and the robot's actual travelled path in blue.

Therefore, this odometry system is replaced by encoder odometry system which provides more stable and reliable performance. It uses optical sensor, infra-red emitter and receiver pair to detect the changes of the encoder disk which attached on the robot's wheel.

4.7 Hardware Architecture

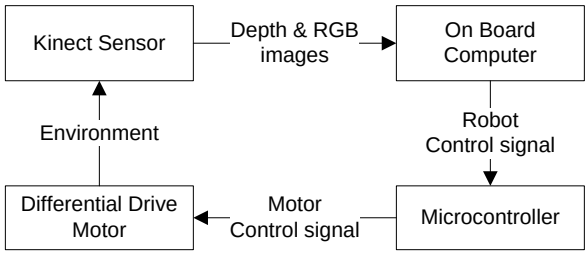


Figure 4-19: Hardware Architecture

Components of hardware include:

- Kinect sensor
- 14 .1" Laptop
- USB-UART converter
- Microcontroller
- Encoder sensor
- DC motor and driver
- 12V battery

All the hardware are mounted and connect on a wheeled differential drive robot in the manner descripted in figure 4-19. The Kinect sensor is powered by a 12V battery and its data is processed by the on-board computer. The computer is then outputs signals to microcontroller through serial port using USB to UART converter. According to the control signals, the microcontroller controls the motors' speed and direction using 2 DC motor drivers. These drivers control the DC motors which powered by a 12V on-board battery.

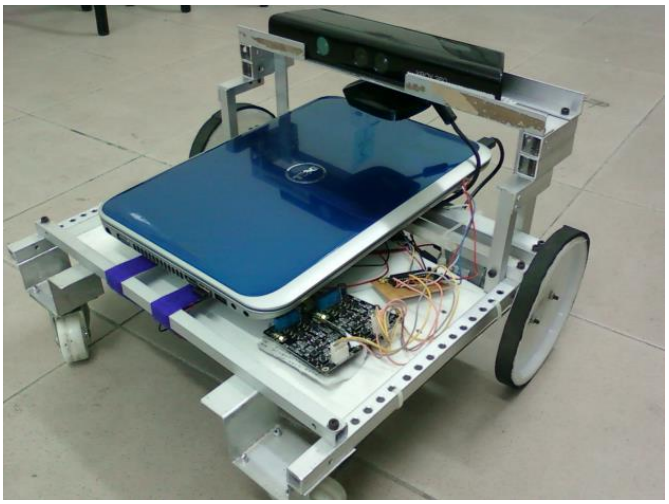


Figure 4-20: The Robot Complete Hardware

4.7.1 Robot Hardware Structure

The structure is designed to hold all components on board. Weight of all components is approximately 8 kg. There are total 4 wheels attached to the robot's structure. 2 free wheels are mounted at the front of the robot base. Its function is mainly providing support and balance for the robot. There are 2 driving wheels mounted at the back of the robot base. They are powered by 2 DC motors separately. Motor output shaft is mounted directly to the wheel using a coupling designed to hold motor's output shaft together with the wheel. Therefore, it is a differential drive based mobile robot with additional 2 supporting free wheels. The length of the robot's structure is 35 centimetres and width is 40 centimetre.

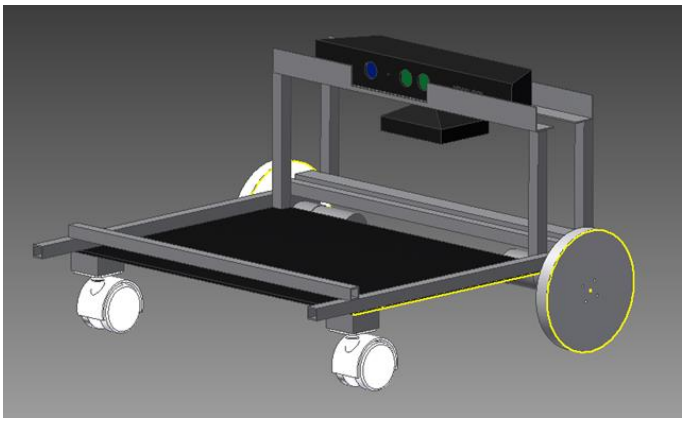


Figure 4-21: Structure Isometric View

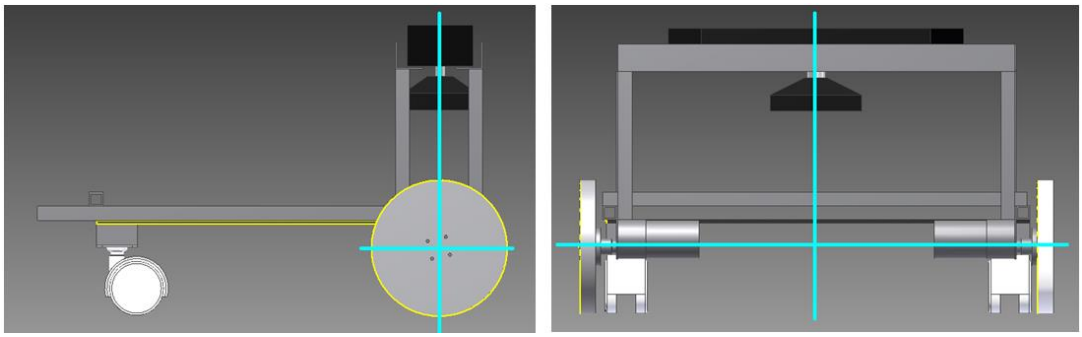


Figure 4-22: Placement of Kinect Sensor on Robot's Structure

Kinect sensor is placed on 2 'L' bars which mounted at 23centimetre above ground. Sensor placement is aligned with the centre point between 2 wheels, both front and side direction. The tilt of the sensor is designed parallel with the floor. After putting the Kinect sensor on there, the centre of sensor is approximately 25 centimetres above ground and 30 centimetres behind the robot's front edge. With a minimum sensing distance of 50centimetres and viewing angle of 57° by 47°, robot have a blind area of 15 centimetres in front as shown in figure 4-23.

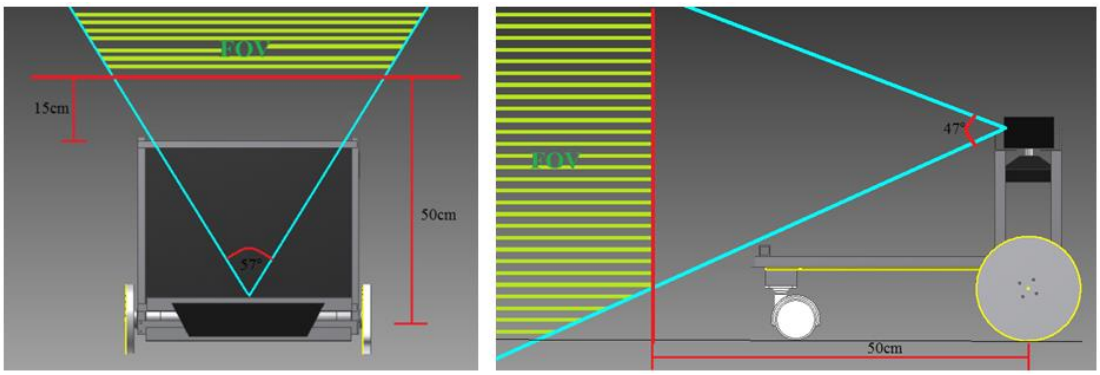


Figure 4-23: Kinect Sensor Field of View

4.7.2 Encoder Odometry

Infra-red emitter and receiver pair does not provide as high resolution, simplicity and stability as rotary encoder does in performing distance encoding. However, to reduce the cost of building this robot, infra-red emitter and receiver pair is used instead of

rotary encoder in performing robot's odometry because it only cost around RM5 each. It is an acceptable track-off between price and encoding quality since high encoder's resolution is not required in this application.

It provides higher stability and reliability in measuring robot's motion than visual odometry. Besides, it reduces the computational load for the on-board computer because this odometry routine is carry-out by PIC. Therefore, the overall software architecture has changed to manner shown in figure 4-24.

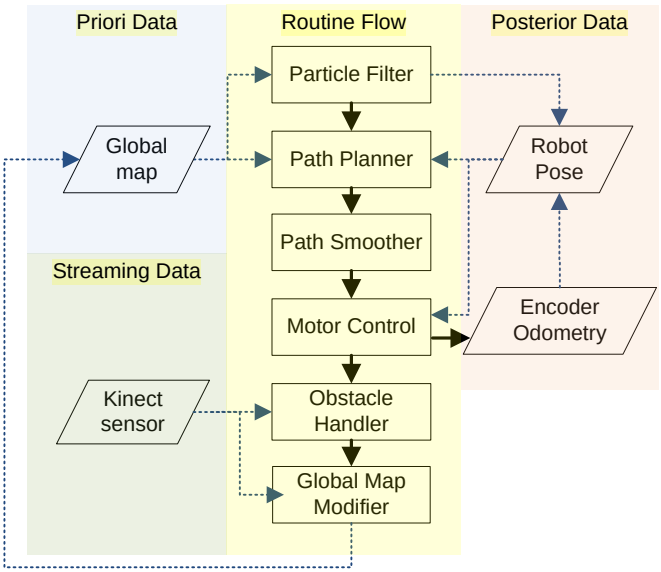


Figure 4-24: Software Architecture with Encoder Odometry

An encoder disk is designed with total 40 black and white color intervals. The number of interval is limited due to the size of IR bean emitted by this emitter. These intervals are printed on a piece of paper which in circular shape with radius same as robot wheels'. It is attached on the inner side of both robots' driving wheels facing the IR encoder. IR encoders placed in 1cm distance near to the disk and parallel with wheel rotational axis.

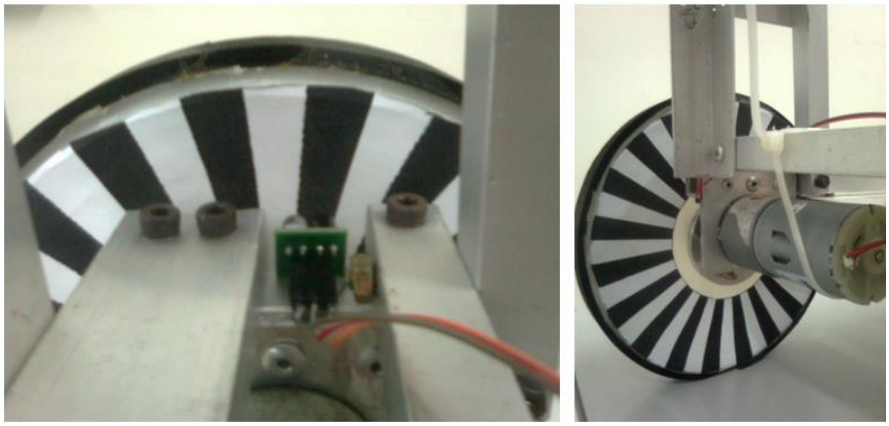


Figure 4-25: IR Sensor and Encoder Disk

The voltage across IR receiver decreases when IR reflected from the white interval of the encoder disk, and increases when IR reflected from the black interval of the encoder disk. Therefore, a LM393 voltage comparator is used to distinguish the color of encoder disk's interval sensed by the IR sensor. It compares the IR receiver voltage with a reference voltage which set by a 5kΩ potential meter. By tuning the potential meter, the threshold of the IR sensor can be fine-tuned to detect color changes more accurately. The digital output of the comparator is then sent to microcontroller which detects and counts the pulses.

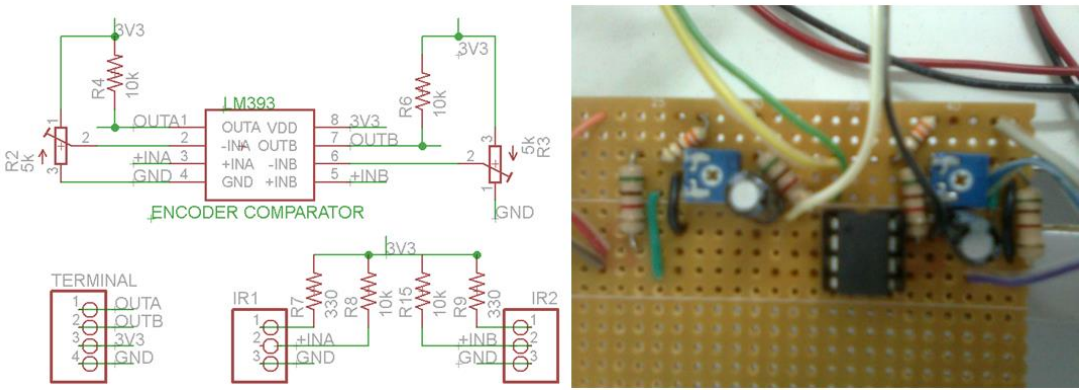


Figure 4-26: Schematic and Board of IR Encoder Board

The performance of encoder odometry is much better than the visual odometry. It provides less than 1.2 centimetre error in a single movement which caused by the encoder disk's resolution.

$$Encoder\ Resolution = \frac{wheel\ circumrefence}{encoder\ pulse\ per\ revolution}$$

$$= \frac{2\pi R}{PPR}$$

$$= \frac{47.123\ cm}{40}$$

$$= 1.178\ cm\ per\ count$$

This error might accumulate and becomes significant as the moving path's length increase. However, it still falls in the acceptable range since the size of the indoor environment is not too large until it can causes meter of error for robot.

4.7.3 DC Motor and Driver

SPG50-100k from Cytron is the DC geared motor model used in the robot. It has higher speed to torque ratio which is suitable for slightly heavy robot moving in slower speed. MD10C from Cytron is the DC motor driver used to control the DC geared motors. It has high maximum continuous motor driving current of 13 Ampere which is important for high torque low speed motor. It requires 2 inputs, direction control in digital input form and speed control in PWM form. A 12V battery is used to power the motor and its driver. Figure 4-27 shown the motor's shaft and wheel coupling mounting on the robot structure.

Figure 4-27: DC Motor and Shaft-Wheel Coupling

4.7.4 Robot Embedded System

dsPIC33FJ128MC802 microcontroller from Microchip was implemented in the robot embedded system as the processing unit that handles encoder sensor data, USB communication data, and generates motor control signals. It is powered by the USB-UART converter which supplying 5V to the circuit board. Since this PIC use only 3.3V, a 3.3V voltage regulator is used to step down the supply voltage to the level PIC need.

Input captures module is used to read encoder sensor data which are in digital pulse form. It takes 2 channels to receive 2 encoders' data. Respective interrupts will be generated when a channel detects a voltage changing edges from the sensor. Numbers of pulses are recorded and by substituting robot wheel's radius, distance travelled of each wheel can be calculated.

$$D = \frac{a}{PPR} * 2\pi R$$

Where
D = wheel's travelled distance
a = accumulated pulse
PPR = encoder's pulse per revolution
R = wheel radius

Pulse Width Modulation (PWM) module is used to generate speed control signal to the DC motor drivers. The frequency of the PWM is set at 1k Hz which allow the motor to function as normal. Universal Asynchronous Receiver Transmitter (UART) module is used to communicate with computer through a UART to USB converter with baud rate of 9600kbps. Serial communication at this speed provides stability and having lesser probability of error if compared with high speed serial communication. Moreover, a timer is used to invoke compensator routine which calculates suitable robot motor's speed based on the input command from computer. The overall schematic and layout design are shown in figure 4-28 and 4-29 below.

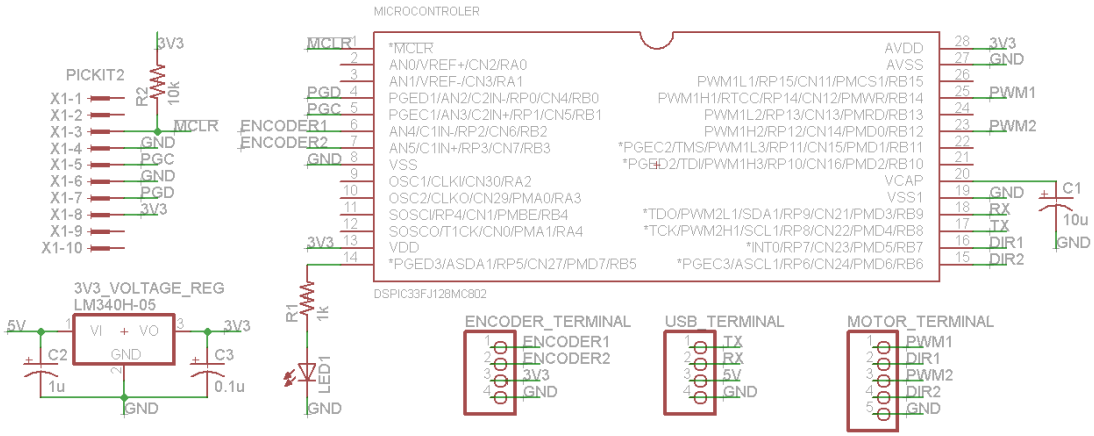


Figure 4-28: Schematic of Microcontroller Board

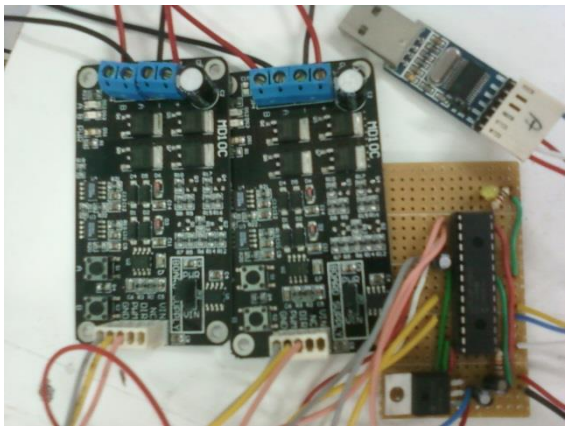


Figure 4-29: Motor Driver, Microcontroller Board and USB-UART Converter

4.7.5 Communication Protocol

To prevent data distortion and loss during the communication between computer and microcontroller, a special serial communication protocol has been developed to ensure the completeness of software-hardware communication system.

The data transferred using this protocol can be divided into 2 types, motion data and function data. Function data are those instructions that instructs robot to pause motion, resume motion, or request motion status from microcontroller.

Motion data are those instructions that instructs robot to move in different manner. Motion data are sent in as a series of bytes, starting by a start byte which assigned as unsigned char 32. When UART receiver module received this start byte from computer, it indicates a starts of the motion data series and prepares to receive the subsequence motion data. The next byte in this series in the motion types can be either move forward, left turn or right turn. The magnitude of the motion, distance or angle is followed after that by using 2 bytes. Therefore, the maximum distance robot can moves in a single path is 65535 in centimetre. After the fourth bytes of motion instruction series has been received, microcontroller will carry-out the motion instruction on the next timer interrupt.

Table 4-4: Motion Data Series

Number of Byte	Data Type	Data Description
1	Start Byte	Assigned as unsigned char 32
2	Motion Type	Forward / Left turn / Right turn
3	Magnitude Byte 1	Lower byte of motion magnitude
4	Magnitude Byte 2	Higher byte of motion magnitude

Communication of the motion data series having one safety feature which function data do not have. From computer point of view, it expects an acknowledgement reply from microcontroller every time after a motion data is sent. The acknowledgement reply must be exactly the same as the previous sent data. Mismatch of the reply and data will cause an abortion of current data series sending and followed by re-sending procedures. This feature ensures correct data are being received by the microcontroller before it starts to perform.

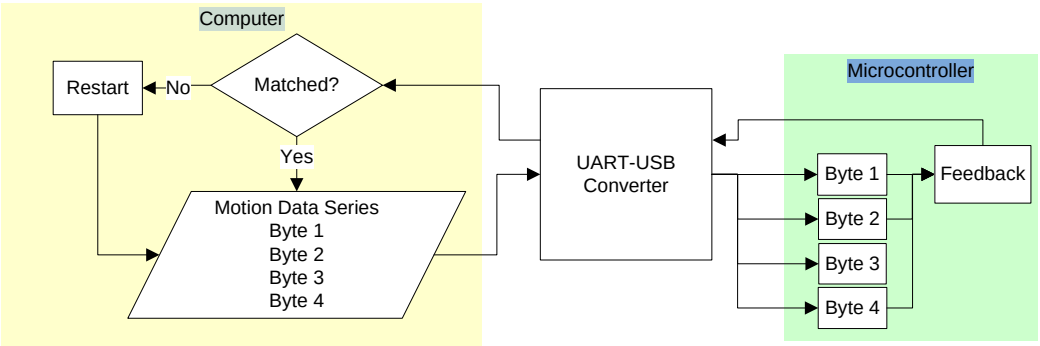


Figure 4-30: Motion Command's Communication Protocol

4.7.6 Motor Control System

The control system routine is invoked by a microcontroller's timer which overflown and interrupted 24 times per second. In other word, the motor control system module will be executed every 0.041667 second. It controls the speed of the 2 motors according to the amount of distance instructed from computer.

This motor control system takes motion command's magnitude either for forward, left turn or right turn as the reference or set distance. Then it compares set distance with travelled distance generated from the input capture module. By averaging the accumulated encoders' counts from both left and right wheels, total robot's travelled distances can be calculated by multiplies the counts with wheel circumference-encoder's pulse per revolution ratio. Distance error is calculated from subtracting set distance from travelled distance.

Robot's wheels are needed to be move in synchronized manner to ensure robot's heading direction while moving forward will not change since it is able to cause inaccuracy in motion. Besides, inaccuracies also occur when the centre of rotation shifted during robot self-turning as the result of unsynchronized motor motion. Therefore, maintaining the synchronization of both motor is the second task of this control system. An alignment error is calculated by finding the different between left and right encoders' counts.

Both distance and alignment errors are then multiply by their respective proportional constant. Left motor output speed is decided by subtraction of distance and alignment output while right motor output is decided by summation of distance and alignment output. Both motors' speeds are limited between 70% and 20% of the top speed. The upper speed limitation is to prevent motor from moving too fast and the lower speed limitation is to overcome the minimum speed motor can respond to when desire speed is lower than that. The overall control system simulation diagram is shown in figure 4-31 below. An example of the averaged motor output speed is demonstrated in figure 4-32.

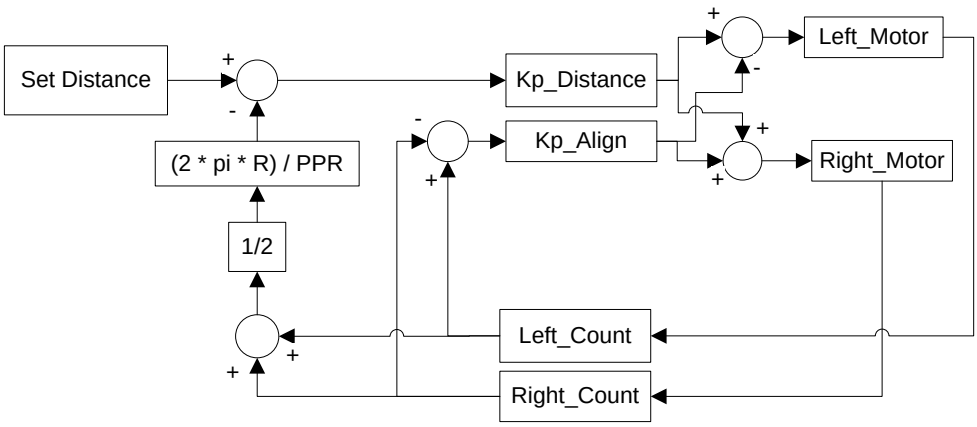


Figure 4-31: Motor Control System Diagram

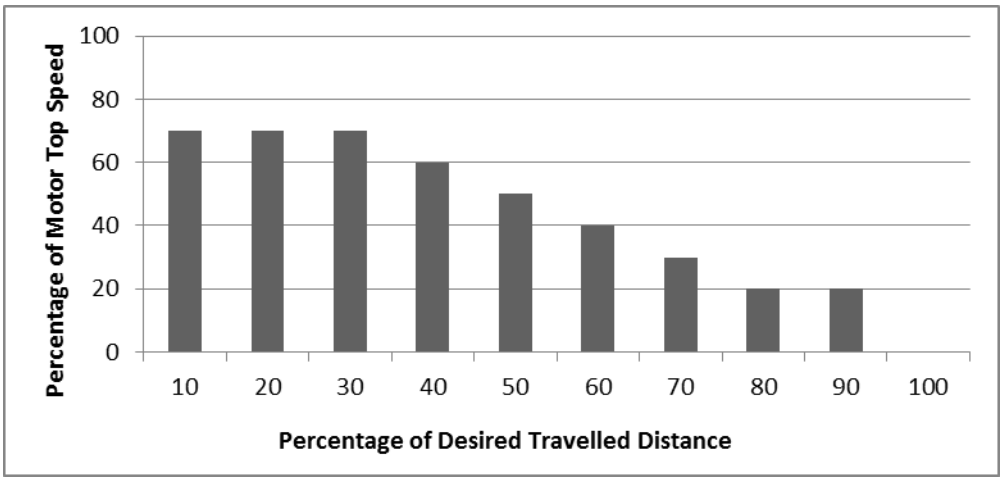


Figure 4-32: Graph of Speed against Travelled Distance

4.8 Overall Performance

By combining all software and hardware modules together, the mobile robot is able to navigate from its starting location to destination point with errors. Errors can built-up and accumulates from the imperfection of every module. Starts from a low quality scan of the environment which causes inaccurate records of robot navigable area. Path planner will be affected and generates the non-optimized path. The accumulated encoder's resolution error also contributed a very small part of overall error.

However, these errors are negligible as long as the global map was acquired in proper manner.

The largest source of error is occurred during the localization section. The poor performance of the particle filter fail to matches robot's surrounding ranges with particles' and hence generates inaccurate starting point for robot. The inaccurate starting point caused certain degree of global map shift and therefore, robot is navigates with error along the path. Figure 4-33 shown the planned path (green) and an example of robot's travelled path (yellow) with error along the entire path which caused by an inaccurate start location.

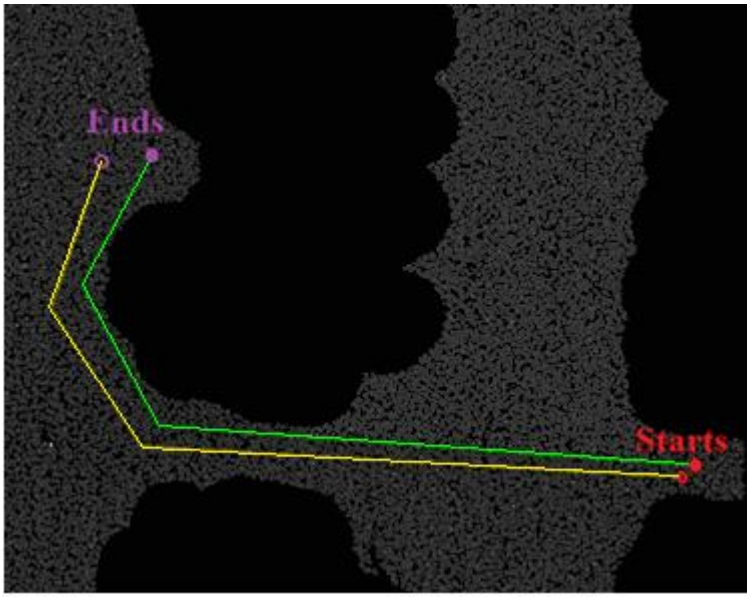


Figure 4-33: Result of Starting Point's Coordinate and Heading Error

A fix point is then assigned for robot as its starting point to start its navigations as the effort of eliminates the starting point error caused by unsuccessful localization. As the result, the map shifted error was solved and the robot is able to navigates by following the flow chart below during it navigation.

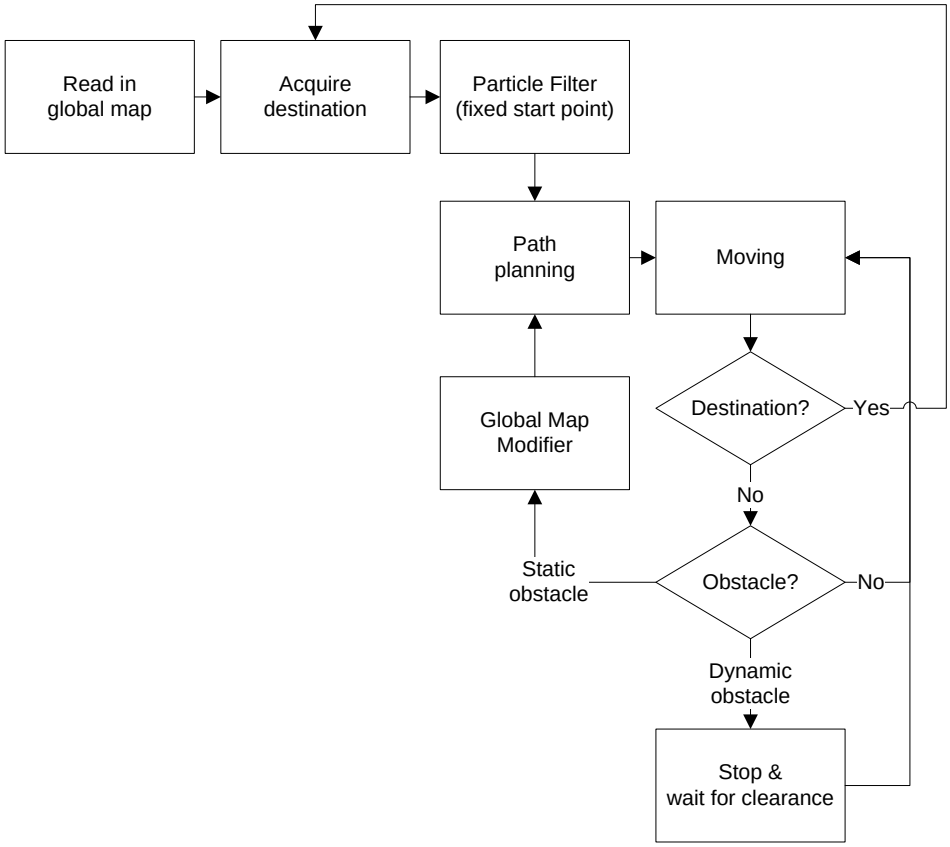


Figure 4-34: Overall Robot's Functionality Flow Chart

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

In conclusion, the overall concept has been proved workable. Kinect sensor has shown its capability in 3D image processing and vast possibility in robotic applications. By using the Kinect sensor's libraries, numerous functions can be added onto the mobile robot to perform tasks such as indoor map acquisition, robot localization, obstacle handling and navigation.

This mobile robot has achieved the project's aim by performing navigation through-out an indoor environment autonomously, safely and some-how accurately. Besides, the cost of building this robot is still maintained far below advance devices equipped mobile robot. As the result, it can be applied in indoor environments like libraries, factories, and houses to perform tasks like goods transportation or surveillance. It can also be further modified if specialist functions are desired.

However, there are still plenty of spaces of improvements on this robot. Particle filter localization has only been performed once at the start of navigation due to the prohibitive computational cost. It is thus not useful for localizing and tracking the robot throughout its path. An alternative method is to track the robot after every motion step (or every few steps) would improve the reliability of the robot's position-sensing, given such methods did not interfere with the response time of obstacle detection.

Instead of acquires environment scan manually, global map can be acquired by robot autonomously without human intervention. Moreover, besides robot

navigable area, global map can include more information such as landmark identification and location. These information are very helpful in many localization methods. Landmark detection is also a better approach to perform particle filter instead of measuring map boundary as this robot's particle filter does.

This work has also shown the limitation of visual odometry in certain situations. Encoders are a possible way to improve dead-reckoning of the robot's motion. Optical encoders are well functioning in this project, but in improving odometry resolution, rotary encoder with 500 pulses per revolution can be implemented together with advance position tracking algorithm such as Kalman filter in 2 dimensions. Additionally, since this robot is only workable in flat indoor environment, accelerometers and gyroscopes could be added to aid inertia measurement and terrain slope as well as robot posture detection which enable robot to expand its workable environment type.

In obstacle handling, this robot distinguish static and dynamic obstacle by checking obstacle's speed. Obstacle in too low speed or no speed which still remain in robot path 3 seconds after the robot stopped is considered a static obstacle. This method cannot identify the moving direction of the obstacle in or outside robot path. By tracking obstacle's heading and speed, prediction and projection of obstacle's moving path can aid the robot's navigation when moving in environment with many obstacles which moving high speed.

REFERENCES

John, D.S., Microsoft Kinect to hit store at November 4th , CNN [Online] 14 June. Available at: http://edition.cnn.com/2010/TECH/gaming.gadgets/06/14/microsoft.kinect.press/index.html [Accessed 18 April 2013].

Ptgrey, 2013. Bumblebee2 Stereo Vision Product, [Electronics print], Available on http://www.ptgrey.com/products/stereo.asp [Accessed on 2nd April 2013].

Mega-Imaging, 2013. 3D Times of flight cameras, [Electronics print], Available on http://www.mesa-imaging.ch/ [Accessed on 2nd April 2013]

Sick, 2013. Laser Safety Scanner, [Electronics print], Available on http://www.sick.com/group/EN/home/products/product_portfolio/optoelectronic_protective_devices/Pages/opto_electronic_protective_devices.aspx [Accessed on 2nd 2013].

Maxbotic, 2013. HRLV –MaxBotic EZ Product, [Electronics print], Available on http://www.maxbotix.com/pictures/HRLV/HRLV-EZ%20Ultrasonic%20Range%20Finder.jpg [Accessed on 2nd 2013] .

Sparkfun, 2013. Sharp Infrared Proximity Sensor. [Electronics print], Available on https://www.sparkfun.com/products/242 [Accessed on 2nd 2013].

Karsten, B. and Ewald, P., 2009. Autonomous Land Vehicle, Wiesbaden, Vieweg+Teubner.

Goldberg, S. B., Maimone, M. W., & Matthies, L. (2002). Stereo vision and rover navigation software for planetary exploration. In Aerospace Conference Proceedings, 2002. IEEE (Vol. 5, pp. 5-2025). IEEE.

Hile, H., and Zheng, C. (2004). Stereo video processing for depth map. Technical Report, University of Washington.

Mingxiang, L., and Yunde, J. (2006, October). Stereo vision system on programmable chip (SVSoC) for small robot navigation. In Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on (pp. 1359-1365). IEEE.

Hasan, A. H. A., Hamzah, R. A., and Johar, M. H. (2009, December). Disparity mapping for navigation of stereo vision autonomous guided vehicle. In Soft Computing and Pattern Recognition, 2009. SOCPAR'09. International Conference of (pp. 575-579). IEEE.

Su, L., Luo, C., and Zhu, F. (2006, August). Obtaining obstacle information by an omnidirectional stereo vision system. In Information Acquisition, 2006 IEEE International Conference on (pp. 48-52). IEEE.

Broggi, A., Caraffi, C., Fedriga, R. I., and Grisleri, P. (2005, June). Obstacle detection with stereo vision for off-road vehicle navigation. In Computer Vision and Pattern Recognition-Workshops, 2005. CVPR Workshops. IEEE Computer Society Conference on (pp. 65-65). IEEE.

Liu, Z. (2007). Performance evaluation of stereo and motion analysis on rectified image sequences. Computer Science Department, The University of Auckland, New Zealand.

Stoyanov, T., Louloudi, A., Andreasson, H., and Lilienthal, A. J. (2011). Comparative evaluation of range sensor accuracy in indoor environments. In Proceedings of the 5th European Conference on Mobile Robots, ECMR 2011 (pp. 19-24).

Antunes, M., Barreto, J. P., Premebida, C., and Nunes, U. (2012, October). Can stereo vision replace a Laser Rangefinder?. In Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on (pp. 5183-5190). IEEE.

Salvi, J., Pages, J., and Batlle, J. (2004). Pattern codification strategies in structured light systems. Pattern Recognition, 37(4), 827-849.

Surmann, H., Lingemann, K., Nüchter, A., and Hertzberg, J. (2001, April). A 3D laser range finder for autonomous mobile robots. In Proceedings of the 32nd ISR (International Symposium on Robotics) (Vol. 19, No. 21, pp. 153-158).

Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., ... and Mahoney, P. (2006). Stanley: The robot that won the DARPA Grand Challenge. Journal of field Robotics, 23(9), 661-692.

Lin, S. Y., and Chen, Y. C. (2012). SLAM and navigation in indoor environments. In Advances in Image and Video Technology (pp. 48-60). Springer Berlin Heidelberg.

Earth measurements consulting, 2013. [Online] Available at: http://earthmeasurement.com/GPS_accuracy.html [Accessed on August 31 2012,].

El-laithy, R. A., Huang, J., and Yeh, M. (2012, April). Study on the use of Microsoft Kinect for robotics applications. In Position Location and Navigation Symposium (PLANS), 2012 IEEE/ION (pp. 1280-1288). IEEE.

Smisek, J., Jancosek, M., and Pajdla, T. (2013). 3D with Kinect. In Consumer Depth Cameras for Computer Vision (pp. 3-25). Springer London.

Correa, D. S. O., Sciotti, D. F., Prado, M. G., Sales, D. O., Wolf, D. F., and Osório, F. S. (2012, May). Mobile robots navigation in indoor environments using kinect sensor. In Critical Embedded Systems (CBSEC), 2012 Second Brazilian Conference on (pp. 36-41). IEEE.

Stowers, J., Hayes, M., and Bainbridge-Smith, A. (2011, April). Altitude control of a quadrotor helicopter using depth map from Microsoft Kinect sensor. In Mechatronics (ICM), 2011 IEEE International Conference on (pp. 358-362). IEEE.

Asensio, J. R., Montiel, J. M. M., and Montano, L. (1998, March). Navigation Among Obstacles by the Cooperation of Trinocular Stereo Vision System and Laser Rangefinder. In Anais do 3rd IFAC Symposium on Intelligent Autonomous Vehicles (pp. 456-461).

WJ IV, W., and Bevly, D. (2001, December). Using the Microsoft Kinect for 3D Map Building and Teleoperation. In Proceedings of IEEE/ION PLANS 2012 (pp. 1054-1061).

Montemerlo, M., Becker, J., Bhat, S., Dahlkamp, H., Dolgov, D., Ettinger, S., ... and Thrun, S. (2008). Junior: The stanford entry in the urban challenge. Journal of Field Robotics, 25(9), 569-597.

Swisslog, 2013. Transcar Automated Guided Vehicle, [Electronics print], Available on http://www.swisslog.com/index/hcs-index/hcs-systems/hcs-agv/hcs-agv-transcar.htm [Accessed on 2nd April 2013].

Ganganath, N., and Leung, H. (2012, January). Mobile robot localization using odometry and kinect sensor. In Emerging Signal Processing Applications (ESPA), 2012 IEEE International Conference on (pp. 91-94). IEEE.

Thrun, S. (2002, August). Particle filters in robotics. In Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence (pp. 511-518). Morgan Kaufmann Publishers Inc.

Wikiversity, 2013. Digital Circuit Lab, [Electronics print], Available on http://en.wikiversity.org/wiki/Digital_Circuit_Lab [Accessed on 2nd April 2013].

Cytron, 2013. Rotary Encoder, [Electronics print], Available on http://www.cytron.com.my/viewProduct.php?pcode=B-106-23983 [Accessed on 2nd April 2013].

Cytron, 2013. IR Sensor Set, [Electronics print], Available on http://www.cytron.com.my/viewProduct.php?pcode=SN-IRS-01&name=IR%20Sensor%20Set [Accessed on 2nd April 2013].

MicroChip, 2013. Home Page, [Electronics print], Available on http://www.microchip.com/ [Accessed on 2nd April 2013].

Arduino, 2013. Home Page, [Electronics print], Available on http://www.arduino.cc/ [Accessed on 2nd April 2013].

OpenKinect, 2013. OpenKinect Project Main Page, [Electronics print], Available on http://openkinect.org/wiki/Main_Page [Accessed on 2nd April 2013].

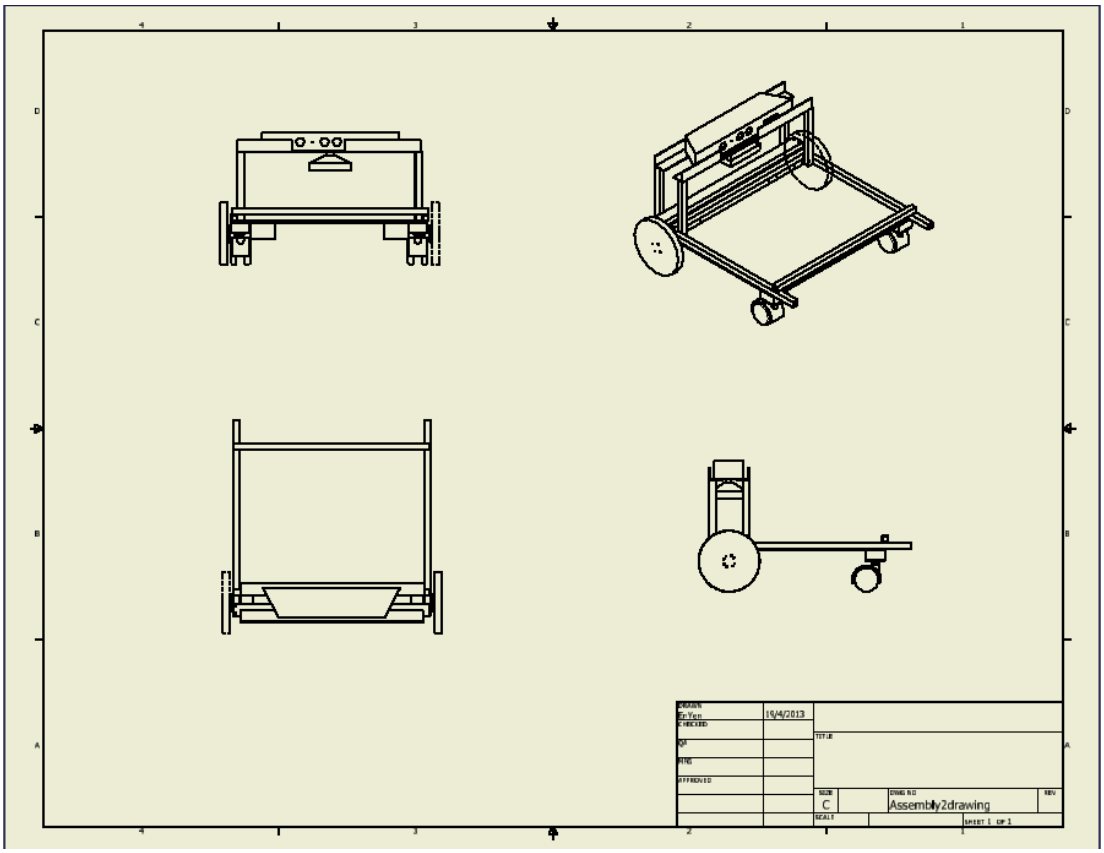
Kinect for Windows, 2013. Kinect for Windows Main Page, [Electronics print], Available on http://www.microsoft.com/en-us/kinectforwindows/ [Accessed on 2nd April 2013].

OpenNI, 2013. SDK Architecture, [Electronics print], Available on http://www.openni.org/ [Accessed on 2nd April 2013].

PointClouds, 2013. Home Page, [Electronics print], Available on http://pointclouds.org [Accessed on 2nd April 2013].

APPENDICES

APPENDIX A: Robot Hardware Structural Drawing



APPENDIX B: Project's Conference Paper

Kinect Sensor on Mobile Robot Autonomous Indoor Mobile Robot Navigation

Puang En Yen
Faculty of Engineering and Science
Universiti Tunku Abdul Rahman
Kuala Lumpur, Malaysia
puangey1@1utar.my

Dr Ng Oon-Ee
Faculty of Engineering and Science
Universiti Tunku Abdul Rahman
Kuala Lumpur, Malaysia
ngoe@utar.edu.my

***Abstract*—Autonomous robotics in indoor environments are traditionally done using a combination of laser range finders, time of flight cameras, and/or stereo vision. This work proposes the use of the Microsoft® Kinect sensor as an alternative lower-cost modality for low-cost mobile robotics. The mobile robot is shown to be capable of navigating a previously captured global map autonomously with obstacle detection and avoidance functions.**

Index Terms - Indoor autonomous mobile robot, path planning, obstacle avoidance, localiazation, kinect sensor.

INTRODUCTION

As technology evolves, autonomous robots have become more and more popular in the manufacturing, health care, security, military and similar sectors. The primary benefit of this technology is that it can function without human intervention and yet perform better under certain circumstances. In other words, autonomous robots can replace human in tasks which are dangerous, repetitive and require high levels of concentration over a long time.

Unlike manual-control robot, autonomous robot do not rely on live input commands. An autonomous robot makes decisions by itself based on its sensor inputs and pre-defined behaviors. Therefore, it can operate with a minimum of human supervision and correction and hence, it behaves as an autonomous individual in its environment.

Autonomous indoor mobile robots are required to navigate through an indoor environment. In order for an indoor mobile robot to be considered autonomous, it must at least have the following capabilities. First, it should be able to locate itself within the indoor environment. Second, it should be able to plan its path to reach its destination. Third, it should be able to correct itself if it deviates from the planned path. Fourth, it should be able to detect and avoid new obstacles. There are also many other capabilities an autonomous mobile robot needs, depending on its function and how the word 'autonomous' being defined by the designer.

This work focuses on navigation in an indoor environment. The autonomous robot only needs to process data concerning the navigable floor areas and possible obstacles without also considering forests, slopes, traffic lights, road lanes, pedestrians, and other environmental factors common in outdoor environments.

There are many types of sensors which can be used by autonomous indoor mobile robots. Popular choices include laser range finders, time-of-flight cameras and stereo vision. However, these sensors can be either too expensive or too computationally complex (or both) for this application.

Kinect™ was introduced by Microsoft® at year 2010.It immediately created a huge wave that hit the world of video gaming and drew interest from robotics hobbyists as well. The Kinect sensor is a low cost device which produces an RGB image, a depth image and sound recordings. Although the depth accuracy and range are not as good as that of a laser sensor, it still provides a good trade-off between price and data quality for indoor mobile robotics.

One other benefit of the Kinect sensor is the numerous open source Kinect™ libraries available for developers, easing implementation of the Kinect sensor on mobile robots. Some examples of these libraries are OpenKinect, OpenNI, Kinect for Windows, Point Cloud Library (PCL), and the Robotic Operation System (ROS).

This paper presents a set of algorithms implemented on an autonomous indoor mobile robot to do path searching, path smoothing, particle filter, obstacle detection and obstacle avoidance. The overall process (including obtaining Kinect sensor data) is also presented.

RELATED WORKS

a. *Range Sensors*

Autonomous mobile robots are not a new development and many types of range sensors have been implemented. Using very basic ultrasonic and infra-red distance sensors, mobile robots are able to find out the distance of the obstacle in front. These sensors are very simple and easy to build but have poor performance due to the limited sensing range and accuracy.

Laser range finders and time-of-flight cameras are currently the highest accuracy sensors in range measurement[1,2]. They are usually used in autonomous mobile robots or vehicles which require precise range measurement to facilitate algorithms such as localization and mapping[3,4]. Due to their high performance, these range sensors are normally at least an order of magnitude more expensive than ultrasonic sensors.

Stereo vision does not actively transmit pulses and observe reflections. Instead it passively captures light from 2 different

positions with overlap. Algorithms which generate depth maps from stereo cameras for mobile robotics are a popular research topic[5,6]. Stereo vision performance can be as good as laser range finders under some conditions, but the computational power needed to executes stereo algorithm is significant[7].

Therefore, robot developers have experimented with merging multiple sensors on a single mobile robot[8,9]. Besides the sensors mentioned above, other sensors such as global positioning system (GPS) and radar sensors have been used in advanced autonomous vehicle[10,11].

The Microsoft® Kinect™ is a kind of active stereo vision camera with an RGB camera and microphone array. Its depth camera emits structured infra-red light which can easily be affected by sunlight[12]. Therefore, almost all robots with Kinect sensors embedded are designed for indoor applications. It also been used together with laser and stereo range finders to perform indoor navigation tasks[13]. Supplementary sensors such as rotary encoders are often added to autonomous mobile robots to perform dead reckoning localization[14, 15].

b. *Navigation*

Some autonomous vehicles[10, 11] plan their path from a global map using dynamic programming based on hybrid A* search[8]. Paths are computed for each discrete cell of route network definition file. Dynamic programming is able to handle large divergent sets of future decisions and is used to provide a complete path. This requires significant computation, especially when dealing with a large global map. The planned path is then smoothened by gradient descent method since self-turning is not applicable for steering vehicle.

Obstacle detection is done by detecting vertical points which exist above the detected ground plane. In outdoor environments, terrain slope and the rolling and pitching of the vehicle increases the difficulty of obstacle detection. The obstacles that are higher than the vehicle height are ignored. The expected differencebetween3D laser inter-ring distance and range can be computed to solve this issue. In the work presented here, the indoor terrain is assumed to be flat and the robot is assumed to never roll or pitch.

Kalman and particle filters are a popular method to predict dynamic obstacle motions and paths. Collision can be avoided by avoiding the predicted dynamic obstacle path. Kalman and particle filters are also used to estimate vehicle state such as position, heading and speed, hence performing absolute localization[15,8].

NAVIGATION CONCEPT

a. *Software Architecture*

OpenNI is the library used to interfaces with Kinect sensor. It provides 2 data stream, RGB image and depth image. The libraries used to process these images are the Point Cloud Library (PCL) and Fovis. The overall data and process flow are shown in Fig. 1.

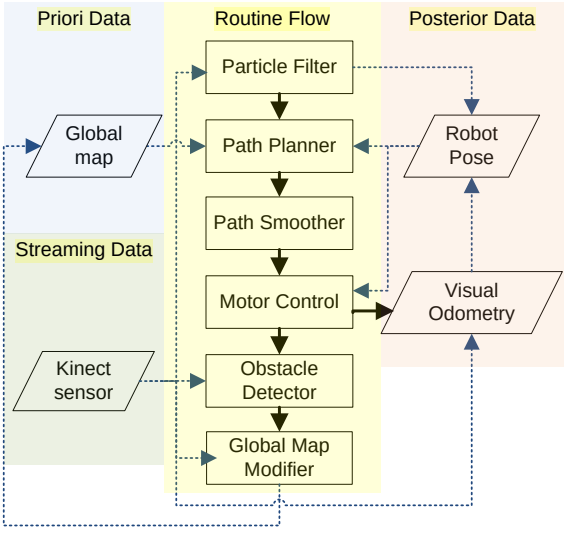


Figure 1. Software Architecture.

b. *Global Map*

To reduce the required computational power by the robot, it does not perform mapping of the environment during navigation. In other words, it does not keep track of the path is has traversed or produce a map of the environment at the end. This significantly reduces on-board computational requirements, with this advantage becoming more obvious when the environment's size growing larger.

However, mobile robots still need a map in order to navigate. Hence, before the robot can be deployed, it first needs a full and complete map of the environment, called the global map. This map contains all the robot navigable locations in the environment and is pre-processed to occupy minimal memory space.

The global map is produced manually and separately from the robot. The environment can first be scanned by Kinect™ compatible 3D reconstruction software such as Scenect, ReconstructMe or Skanect to produce a 3D point cloud (Fig. 2, left). This global map then undergoes a series of modifications for suitability (Fig. 2, right).

- Voxel Grid filter – Down-sampling to reduce point cloud size and hence the computational time.
- Radius Outliner Remover – Filters out noise points.
- Plane Segmentation – Determines planes and their plane coefficients. The largest is set to be the floor.
- 3D transformation – Transforms point cloud so that the floor detected is on XY-plane.
- Extract floor point indices – To differentiate between floor and obstacle.
- Remove overlapped indices – Remove floor points which have obstacles above or near it.
- Floor clustering – Only continuous floor points are kept.

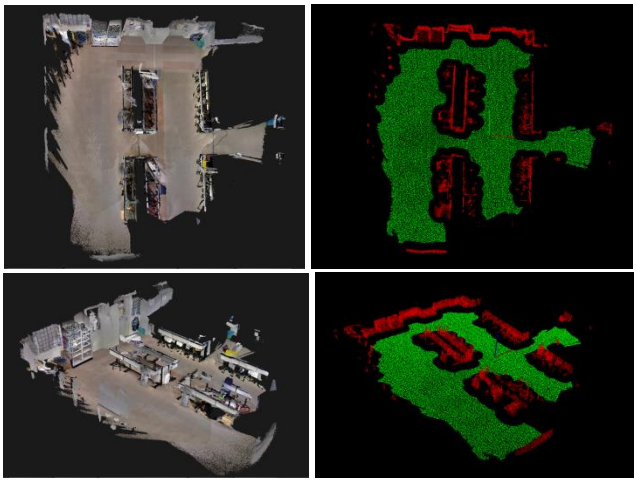


Figure 2. Raw scanned (left) and processed (right) global map.

c. *Particle Filter*

The particle filter is a very easy and direct localization method[16]. Particles are created on the global map with a density that depends on the robot's size. Each particle is assigned a random coordinate and heading.

In the measure phase, robot motions are recorded. By self-turning to certain degrees, the robot measures the distance of its surrounding in the environment. In the update phase, all particles are moved by the robot's motions as recorded at measure phase. The particles' surrounding distances are being measured too in the global map. The particles' weights are then computed based on the similarity between its' and the robot's surrounding distances.

After resampling, particles which are located closer to robot exact location are more likely to survive. The robot's initial position will be confirmed if particle weight has exceeded a certain threshold. No particle filter routine will be executed after that in order to reduce computational resource demands during navigation.

d. *Path Searching*

A* with heuristic is the path searching technique implemented in this paper. The global map is first partitioned into discrete cells, the size of the cells determines the resolution of the path. From the starting point, 8 cells in 8 equally separated directions are explored by a single cell step and a cost is assigned into each cell. The cell's cost is used to determine the next cell which is optimal to be further explored among the eight. The heuristic function used in determining the cell's cost is calculated by the Euclidean distance between the exploring point and end point, as shown in (1).

$$Cost_{cell} = CD_{start,cell} + ED_{cell,end}. \quad (1)$$

Where CD = Cell Distance,
ED = Euclidean Distance

The output of this path search is a vector of point which connects the start and end points by linking them with lines. It is very impracticable for robot to move in that way because it contains too many turning points. Therefore, the path should be optimized after that.

e. *Path Smoothing*

The path smoother checks whether any obstacle exists within the straight line connecting the start and end points. It looks for points that can be bypassed and then connects the others to reduce the number of lines in the path. In Fig. 3, the red lines are the path before smoothing and green lines are path after smoothing. After smoothing, the number of turning points required is reduced.

The path smoother here ensures that none of the path exceeds the navigable map boundary. This is better compared to the gradient descent smoothing method[10] which produces paths which cut into non-navigable areas. In this case, there is no arc line planned because the robot is a differential drive robot, not steering robot. Moving in a straight line and self-turning are much easier than moving in an arc path. Fig. 3 shows the unsmoothed path in red, the smoothed path in green and gradient descent smoothed path in blue.

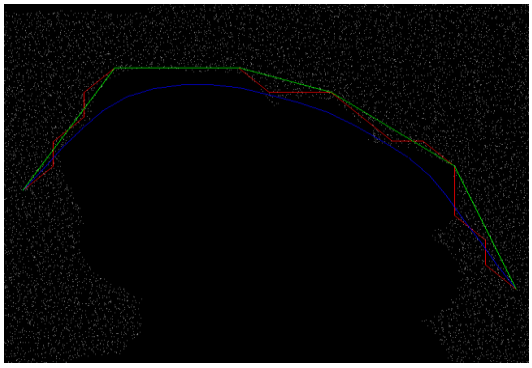


Figure 3. Three different planned paths.

f. *Obstacle Detection*

After the path has been prepared, the robot will start to move to its first point. While it is moving forward, the Kinect sensor captures the depth information (called the local map) in front of the robot with 57 degree by 47 degree field of view.

Instead of checking the entire space in front, the robot only checks if any obstacle exists in the given path direction within the local map. A pass-through filter is used to filter the local map to a rectangle of robot width by robot height by path length which is then checked for any existing points (obstacles). Since the sensor's height on the robot is a known constant, the implementation of plane segmentation to detect floor can be avoided in the local map. Every point below the robot's height can be simply ignored since this robot is not design to traverse staircases or steep slopes.

The local map is then transformed to the next point of path, P_{n+1} and heading to P_{n+2} and checked again for obstacles using trigonometry (2), (3) and the transformation shown in (4). Ideally full path obstacle detection can be done immediately if the whole path is located within the sensor field of view. However, the 5.5 meter maximum depth range limits its ability to achieve that. Moreover, looking too many steps/points

forward increases the robot's respond delay. The obstacle detection routine is only performed in the event that a successful depth image obtained.

$$\alpha = tan^{-1}\left(\frac{P_{n,y}-P_{n+1,y}}{P_{n,x}-P_{n+1,x}}\right). \quad (2)$$

$$d = \sqrt{(P_{n,x}-P_{n+1,x})^2 + (P_{n,y}-P_{n+1,y})^2}. \quad (3)$$

$$Translation_{FOV} = \begin{bmatrix} 0 \\ 0 \\ -d \end{bmatrix}, \quad Rotation_{FOV} = \begin{bmatrix} 0 \\ \alpha \\ 0 \end{bmatrix}. \quad (4)$$

g. *Obstacle Avoidance*

When the robot detects any obstacle in its path, it will stop and wait for a few seconds. This is to distinguish between static and dynamic obstacles since dynamic obstacle will have moved out from the robot's path after a few seconds while static obstacles would not.

If a static obstacle is detected, its size and location are marked for modification of the global map. The obstacle cluster is measured from the local map using Euclidean cluster extraction. It is then transformed to match with the current position of robot referring to the global map using (5) and (6).

$$Translation_{obstacle} = \begin{bmatrix} robot_x \\ robot_y \\ 0 \end{bmatrix}. \quad (5)$$

$$Rotation_{obstacle} = \begin{bmatrix} -90° \\ 0 \\ robot_{heading} - 90° \end{bmatrix}. \quad (6)$$

Since the local map is captured in such a way that the floor is on the XZ-plane and the Z-axis points forward, the obstacle cloud needs to be transformed -90° about X-axis so that both obstacle and global map are refer to same floor and -90° about Z-axis to align with the robot's heading.

Now the obstacle and global map point cloud are aligned, those points of the global map which have obstacle points are removed. The robot is then ready to re-plan its path to reach its destination (section III.D), shown in Fig. 4.

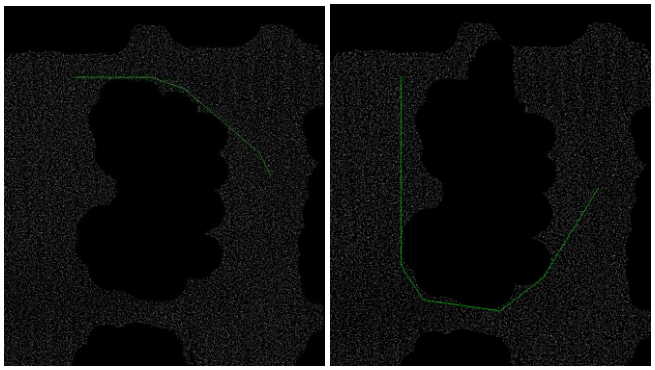


Figure 4. Global map modification and re-planning after static obstacle found.

Robot Design

a. *Hardware Architecture*

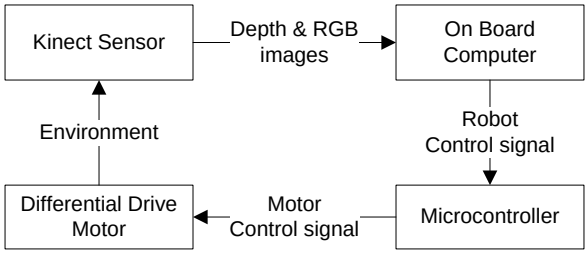


Figure 5. Hardware Architecture.

All the hardware (Fig. 5) is mounted on a wheeled differential drive robot (Fig. 6). There is an on-board computer processing Kinect™ data which outputs signals to a microcontroller through serial port by a USB to UART converter. The microcontroller controls the motors' speed and direction using 2 DC motor drivers. All devices besides the computer are powered by a 12V on-board battery.

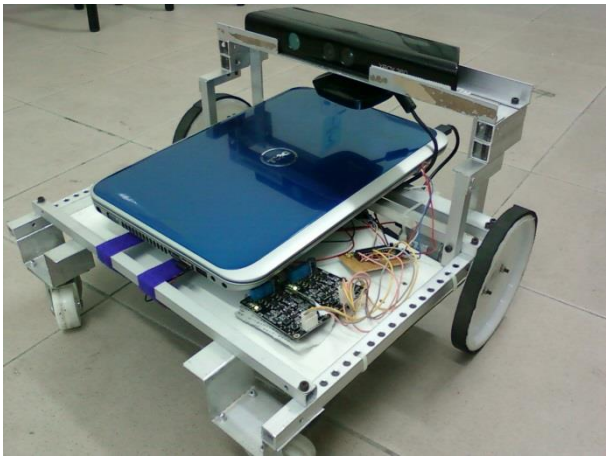


Figure 6. Robot hardware.

b. *Visual Odometry*

To reduce the number of sensors used and the overall cost, robot position measurement is carried out without any rotary encoder. Instead, 3D visual odometry using FOVIS was implemented to measure robot motion. XYZ translation, roll, pitch and yaw can be determined using the Kinect sensor. It utilizes RGB and depth images to measures robot's motions during the robot's self-turning and forward motion.

Therefore, the mounting accuracy of Kinect sensor on robot is very crucial in order to maintain the motion measurement's accuracy. The center of the wheels' rotation axis is aligned right below the visual center of the sensor so that the local map does not need to be transformed before processing. Moreover, the sensor's tilt must be parallel with the floor plane.

Experiments and Results

The time taken for robot path planning largely depends on the size of the global map, processing unit and the distance between start and goal point. The orientation of obstacles also significantly affects the path process time. Table 1 shows a test measuring time taken to plan a path. This test was conducted using the approximately 80 meter square global map shown in fig. 7 and a Dell laptop with Intel® i5 processor and 4GB DDR3 ram.

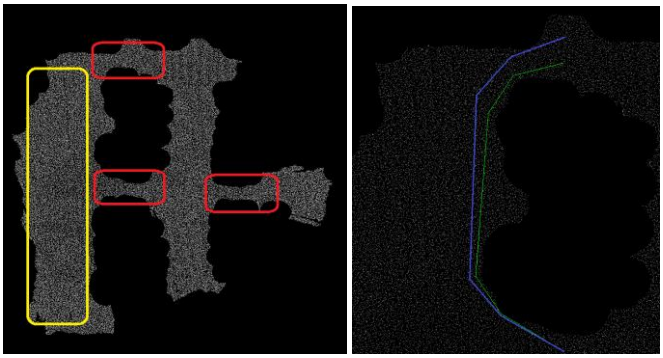
Table 1. Result of path length against processing time.

Path length (meter)	Processing time (seconds)
3	1
6	3
12	6

New obstacles in path can be identified and reacted to in around half a second. This value can be altered depending on the number of turning points which exist in path and the maximum range for detecting obstacles within the local map.

The implemented visual odometry does not measure rotation accurately in confined spaces and places filled with plain surfaces obstacle (shown as red boxes in fig. 7(a)). Besides, linear movement in a long and empty space (shown as a yellow box in fig. 7(a)) also causes inaccuracy in measurement of forward traveled distances.

As the result the robot loses track of the distance and angle traveled. However, the robot was still able to reach the target area safely with few centimeter discrepancies. Fig. 7(b) showed an example of robot test run where the planned path is shown in green and the robot's actual traveled path in blue.



(a) (b)

Figure 7. Sections of global map and an example of robot traveled path.

Conclusion and Future Works

In conclusion, the overall concept has been proved workable. The mobile robot presented in this paper can be applied in indoor environments like libraries, factories and houses to perform tasks like goods transportation or surveillance. It can also be further modified if specialist functions are desired.

Particle filter localization has only been performed once at the start of navigation due to the prohibitive computational cost. It is thus not useful for localizing the robot throughout its path. An alternative method is to localize the robot after every motion step (or every few steps) would improve the reliability of the robot's position-sensing, given such methods did not interfere with the response time of obstacle detection.

This work has also shown the limitation of visual odometry in certain situations. Encoders are a possible way to improve dead-reckoning of the robot's motion. Additionally, accelerometers and gyroscopes could be added to measure terrain slope and robot posture.

REFERENCES

[1] S. Todor, A. Louloudi, H. Andreasson, and J.Lilienthal, "Comparative evaluation of range sensor accuracy in indoor environments," In Proceedings of the 5th European Conference on Mobile Robots, pp. 19-24, 2011.

[2] T. Bernhard and D. Zukowski, "A comparative study of structured light and laser range finding devices," unpublished, Correll Lab in the University of Colorado, 2013.

[3] Gorte, B., Khoshelham, K., and Verbree, E, "Indoor navigation by using segmentation of range images obtained by laser scanners," International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 37, pp. 971-976, 2008.

[4] Wang C. C. and Thorpe, C., "Simultaneous localization and mapping with detection and tracking of moving objects," In Robotics and Automation, IEEE, Vol. 3, pp. 2918-2924, 2002

[5] Hasan A., Hamzah R. and Johar M., "Disparity Mapping for Navigation of Stereo Vision Autonomous Guided Vehicle," In Soft Computing and Pattern Recognition, IEEE, pp. 575-579, 2009.

[6] Yong-guo, Z., Wei, C., and Guang-liang, L., "The Navigation of Mobile Robot Based on Stereo Vision," In Intelligent Computation Technology and Automation, IEEE, pp. 670-673, 2012.

[7] M. Antunes, P. Barreto, C. Premebida, and U. Nunes, "Can stereo vision replace a laser rangefinder?," International Conference on Intelligent Robots and Systems, IEEE, pp. 5183–5190, 2012.

[8] Lin, S. Y., and Chen, Y. C., "SLAM and navigation in indoor environments," Advances in Image and Video Technology, pp. 48-60, 2012.

[9] R. Asensio and M. Montiel, "Navigation among obstacles by the cooperation of trinocular stereo vision system and laser rangefinder," 3rd IFAC Symposium on Intelligent Autonomous Vehicles, 1994.

[10] S. Thrun et al, "Stanley : The robot that won the darpa grand challenge," The 2005 DARPA Grand Challenge, pp. 661–692, 2007.

[11] M. Montemerlo et al, "Junior : The stanford entry in the urban challenge," Junior: The stanford entry in the urban challenge. Journal of Field Robotics, 25(9), pp. 569-597, 2008.

[12] El-laithy, R. A., Huang, J., and Yeh, M., "Study on the use of Microsoft Kinect for robotics applications," In Position Location and Navigation Symposium IEEE, pp. 1280-1288, 2012.

[13] Henry, P., Krainin, M., Herbst, E., Ren, X., and Fox, D., "RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments," The International Journal of Robotics Research, 31(5), pp. 647-663, 2012.

[14] W. J. Woodall and D. Bevly, "Using the microsoft kinect for 3D map building and teleoperation," Position, Location and Navigation Symposium IEEE, pp. 1054–1061, 2012.

[15] Ganganath, N., & Leung, H., "Mobile robot localization using odometry and kinect sensor," In Emerging Signal Processing Applications, IEEE, pp. 91-94, 2012

[16] Thrun, S., "Particle filters in robotics," In Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence, pp. 511-518, 2002.

APPENDIX C: Conference Paper Acceptance Letter and Review Form

2nd International Conference on Recent Advances in Sciences and Engineering
(ICRASE 2013)

Notification of Acceptance of the ICRASE 2013

April 29-30, 2013, Hyderabad, India
http://icrase.aspwedo.org/

Dear **Puang En Yen, Dr Ng Oon-Ee***

Paper ID: **961**
Paper Title: **Kinect Sensor on Mobile Robot-Autonomous Indoor Mobile Robot Navigation**

Congratulations! The conference received submissions from different regions, and papers have been selected for presentation and publication. We are pleased to inform you that your paper identified above has been accepted for publication and oral presentation. You are cordially invited to present the paper orally at ICRASE 2013 to be held on April 29-30, 2013, Hyderabad, India

ICRASE 2013 is sponsored by **ASP (Association of Scholars and Professionals)**.

(Important) So in order to register for the conference and have your paper included in the proceeding successfully, you must finish the following SIX steps.

1. Revise your paper according to the Review Comments carefully sent as an attachment with this mail.
2. Format your paper according to the Template carefully.
   http://icrase.aspwedo.org/FormatingInstructions.doc
3. Download and complete the Registration Form sent as an attachment with this mail.
4. Finish the payment of Registration fee at the Bank.
   (The bank payment details can be found in the Registration form)
5. Download and complete the IJSAA Copyright Form sent as an attachment with this mail.
6. Send your final papers (both .doc/.docx and .pdf format), filled registration form (.jpg), copyright form (.jpg format) and the scanned payment (in .jpg format) to us at icrase13@aspwedo.org (Before 4th April, 2013 for Early bird registration/14th April for Regular registration)

If the above requirements are met by the set deadlines, the paper will be included in the ICRASE 2013 conference proceeding, and will be published as a Special Issue of International Journal of Systems, Algorithms and Applications (IJSAA) journal.

Maybe some unforeseeable events could prevent a few authors not to attend the event to present their papers, so if you and your co-author(s) could not attend ICRASE 2013 to present your paper for some reasons, please inform us. And we will send you, the official receipt of registration fee, proceedings after ICRASE 2013.

For the most updated information on the conference, please check the conference website at **http://icrase.aspwedo.org/** The Conference Program will be available at the website in the third week of April, 2013.

Finally, we would like to further extend our congratulations to you and we are looking forward to meeting you in Hyderabad, India!

Yours Sincerely,

ICRASE 2013 Organizing Committees
http://icrase.aspwedo.org/
Hyderabad, India

Review Form of the ICRASE 2013

April 29-30, 2013, Hyderabad, India
http://icrase.aspwedo.org/

Paper ID: 961
Paper Title: Kinect Sensor on Mobile Robot-Autonomous Indoor Mobile Robot Navigation

Evaluation:					
	Poor	**Fair**	**Good**	**Very Good**	**Outstanding**
Originality	○	○	●	○	○
Innovation	○	○	●	○	○
Technical Merit	○	○	○	●	○
Applicability	○	○	○	●	○
Presentation & English	○	○	○	●	○
Match to the Conference Topic	○	○	●	○	○
Recommendation to the Editors:					
	Strong Reject	**Reject**	**Marginally Accept**	**Accept**	**Strong Accept**
Recommendation	○	○	○	●	○
Comments:					
No Comments.					