International Journal of Engineering & Technology, 7 (4.40) (2018) 26-30

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper



Alternative Platform for Vision based Robot Navigation

Himashi A. Peiris[1], Rajitha G. Ranasinghe[2], Tharindu D. Peiris[3]

[1]University of Moratuwa, Sri Lanka & Pearson Lanka (pvt) Ltd, Sri Lanka
[2]University of Moratuwa, Sri Lanka & Vidul Biomass (pvt) Ltd, Sri Lanka
[3]Information Institute of Technology, Colombo, Sri Lanka
**Corresponding author E-mail: himashi.ama92@gmail.com*

Abstract

Robots are closer than ever to leap from the defined world navigation to the undefined world navigation. For that it is crucial to have flexible and alternative platforms to enhance mobile self-made robots to the next level. In this project a vision based navigation robot is developed using Microsoft Windows based software as an alternative to most common Linux based Robot Operating System (ROS). Currently almost all self-made robots with advanced functionalities use the ROS. But what lacks in Linux based system is flexibility for beginners. Developing a mobile robot based on Windows platform was not possible few years ago, but with the development of Single Board Computers (SBC), now finally it is possible to mobilize the windows. It's a marriage between windows flexibility to robotic mobility. The final result of this research is an autonomous robot which is able to navigate through the environment by avoiding obstacles in a dynamic environment using 3D vision. Solution comprises occupancy grid generation by using depth image of Kinect as main input feed combined with data gathered by an Inertial Measurement Unit (IMU). Data then further processed real-time to identify current position of the robot. Also, by using Fuzzy Logic approach, dynamic obstacle avoidance and navigation is achieved. Entire application was developed by LabVIEW and it is installed inside LattePanda single board computer. LattePanda is playing a crucial role here as the central processing unit for all the functions. The robot is developed as a first step for a flexible test platform which can be further improved to achieve flawless undefined environment navigation through tweaking algorithms and introducing more sensory data.

Keywords: *Fuzzy Logic; Inertial Measurement Unit; Kalman Filtering; Kinect; LabVIEW; LattePanda; Occupancy Grid Generation; Path Planning; Single Board Computers*

1. Introduction

The research project "Alternative Platform for Vision Based Robot Navigation" implemented an autonomous robot platform operated on Windows Operating system, which is able to navigate through environment by avoiding obstacles using 3D vision. This is an alternative solution for Robot Operating Systems (ROS) which is based on Linux. This is an integration of existing windows based software solutions with newly introduced Hardware to build a complete flexible robot platform for windows users. Kinect 360, Inertial Measurement Unit (IMU), Arduino UNO and LattePanda Single board computer (SBC) are the main hardware components that are acquired in this research. LabVIEW is used as the main programming language along with the Arduino. This solution supports Virtual Network Computing (VNC) server which allows the user to access robot vision and controls on local network remotely. This platform uses three voltage levels, 12V for Kinect and 5V for LattePanda and 11.1V battery voltage for Motor Driver. 12V and 5V are regulated through buck-boost converters. All components are powered using a 5200mAh Li-Poly battery which can run the system about 1.5 hours.

2. Background

There are three fundamental difficulties in robotics when it comes to autonomous navigation. Namely, mapping, localization and path-planning. To perform path planning and motion planning robot needs a grid or a map [3]. This map will be used for localization when there is no positioning system is available. A lot of researches have been established covering this field of area. With newly emerging ranging sensors, like Laser Rangefinder, Kinect and Time of Flight (TOF) camera. Because of the diversity and different requirements of applications there are many approaches for path planning problems developed by many researchers. Among them vision based approach is one of the key areas in robot navigation at the present. This approach is more accurate when it comes to dynamic object projection and avoidance [4][5].Communication between hardware and software components can be commonly achieved by using communication methods like I²C and Serial Peripheral Interface communication (SPI). Remotely accessing to generated occupancy grids and letting the client to decide the goal point is one of the key features in this research solution. To implement human computer interaction, most of the developers use VNC. Such services will allow the user to access several computing devices on local network. Therefore, setting up VNC server on a headless server is a good combination to make the solution smart. This will provide an interface with a Graphical User Interface and user-friendly solution. Today, most of SBCs provide a Wi-Fi capability on board. So that this will be a low cost solution rather than going for transmitting device which costs more to implement like in self-made projects. Integration of hardware and software platform is not an easy straightforward operation. . For an example, LattePanda provides Inter-Integrated Circuit for communication between arduino and Visual studio code base.

Copyright © 2018 Authors. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

3. Design and Implementation

The entire system architecture and hardware architecture is shown in following Figure 1 and 2.

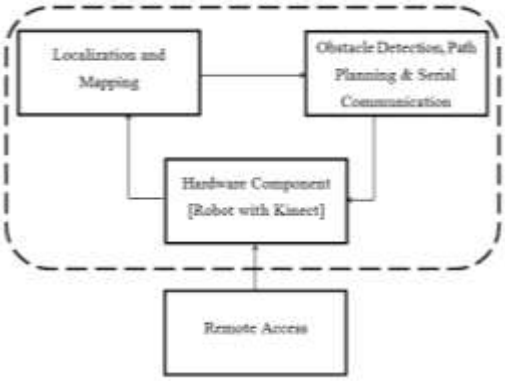

Fig. 1: System Architecture

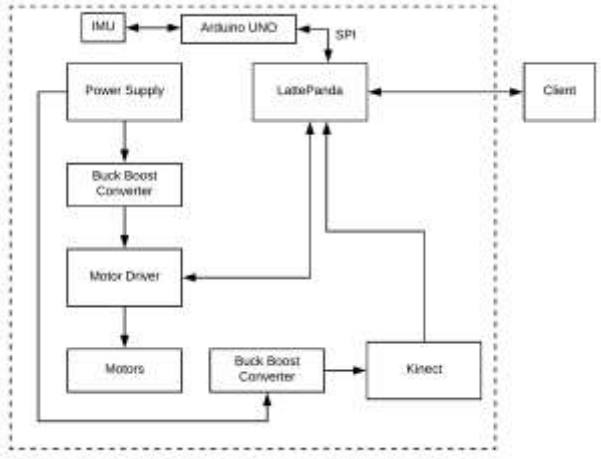

Fig. 2: Hardware Architecture

The real environment is scanned by Kinect 360 sensor and construct Depth image of the real environment using kinesthesia components of LabVIEW. Generation of 2D projection by Depth image and saving the optimized 2D projection map as a text file is the next step of the work flow. To implement this module, first need to set up Kinect Software Development Kit, LabVIEW components and required configurations. Figure 03 shows the process of map generation.

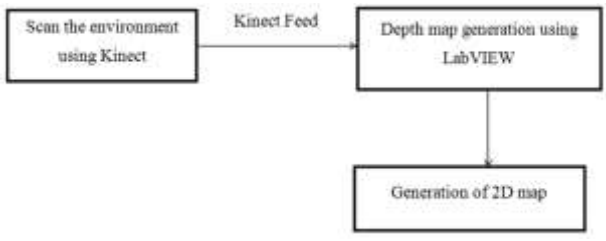

Fig. 3: Mapping Process

RGB video feed from the Kinect sensor is directly monitored. Depth image (80 x 60) is processed to get an approximated distance array to generate the Occupancy grid. This array is limited to 80x60 because of the limited computational power of the SBC. Figure 4 shows the process of distance array generation and division of distance array.

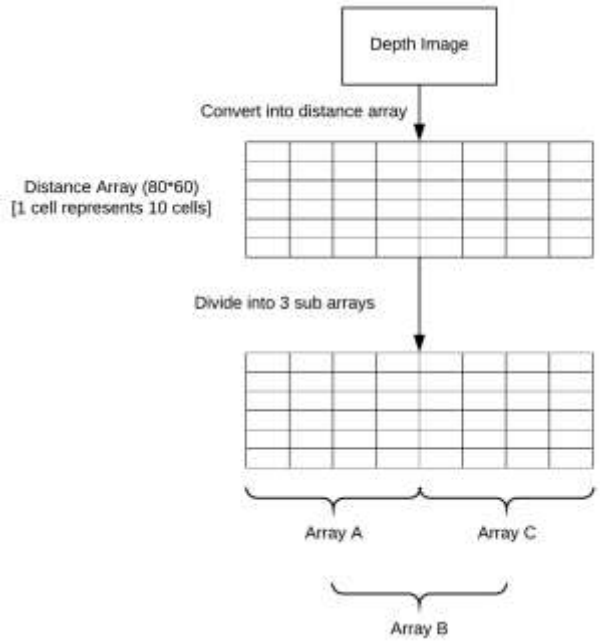

Fig. 4: Distance array generation and division

For the distance approximation, the following second order equation 1 is used.

A. Calculate distance for the point of obstacle in meters using Equation 1. Here r is Raw Disparity. When processing Kinect raw depth data, there is a noticeable difference in point cloud if disparity is not approximated properly. This difference is known as raw disparity in Kinect depth data processing. As per the Open Kinect imaging information following tan approximation function is better than basic first order equation [6].

$$\text{Distance} = 0.1236 \times \tan\left(\frac{\text{r}}{2842.5} + 1.1863\right) \quad (1)$$

B. Use tan approximation, which has a sum squared difference of 0.33 cm while the 1/x approximation is about 1.7 cm.

C. Then add a final offset term of -0.037 centers the original data.

D. Equation 2, 3, and 4 is used for converting coordinates (i, j, z) to (x, y, z) once you have the distance. Here, w is width, h is height, m is minimum distance equals to -10 and scale factor = .0021. This is called as Second Order Depth Approximation Equation.

$$\text{x} = \left(i - \frac{w}{2}\right) \times (\text{z} + \text{m}) \times \text{scale factor} * \times \frac{\text{w}}{\text{h}} \quad (2)$$

$$\text{y} = \left(\text{j} - \frac{\text{h}}{2}\right) \times (\text{z} + \text{m}) \times \text{scale factor} \quad (3)$$

$$z = z \quad (4)$$

IMU sensor has an accelerometer and a gyroscope. Z axis gyroscope data is used to process the Heading (Yaw angle) of the robot. Raw data from the Inertial Measurement Unit is translated to "radians per second" by series of functions and then integrated to obtain the yaw angle in radians. At later stage this is converted to degrees for user-friendliness. For sensor calibration two parameters are used for clockwise and anti-clockwise calibration of the gyroscope. Interfacing with IMU sensor calibrations and filtering process are done by using an external Arduino UNO board which is also programmed using LabVIEW as shown in Figure 5.

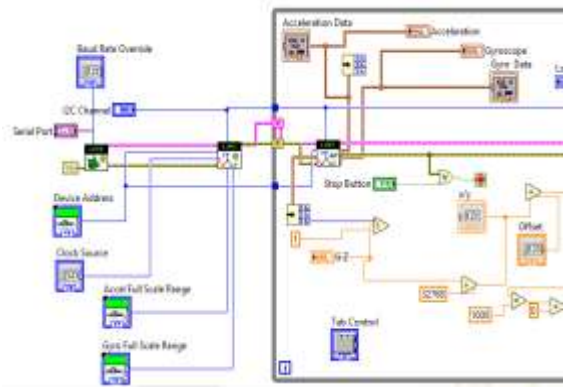

Fig. 5: Initial Inertial Measurement Unit calculations, calibrations and filtering block diagram

Distance array data from Kinect, heading angle from IMU and the current X and Y increment from motor function is used to generate the occupancy grid. Kinect cannot accurately get distances for objects which are closer than 15~20cm. So every object which is closer than 20 cm to robot is represented as number "1" which also indicates it as an obstacle. And other data points of the occupancy grid are filled with approximated distances. This grid is set to have 100 by 100 array and it is saved to a TXT file when the robot stops Kinect data feed. Occupancy grid is generated by applying these logical implementations as shown in the Figure 6.

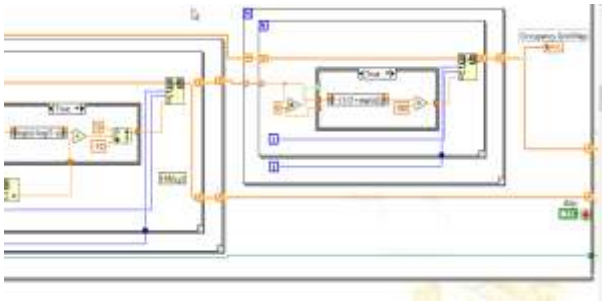

Fig. 6: Occupancy grid map generation block diagram

Generation of the occupancy grid is automated through autonomous navigation. Robot uses current distance array and process it and decide which way to go as next step. Following Figure 7 shows the display when generating the occupancy grid while robot navigates through an unknown environment.

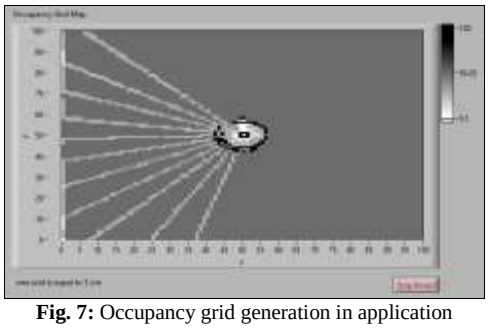

Fig. 7: Occupancy grid generation in application

As mentioned before distance array is divided into eight sub arrays and mean average distances are calculated. Maximum distance point (Maximum Depth) is also calculated and later these average distances are put into main 3 arrays which are used to generate Fuzzy logic controller. Membership function for the fuzzy controller is computed based on three main arrays and maximum distance point. Initially, implemented the fuzzy model used Iterative closest point. However, due to constraints of the Kinect sensor it was not an accurate approach to avoid or detect obstacles while moving (within first 10 cm it cannot see or track obstacles). Fuzzy logic controller is implemented from base binary gates using high-level LabVIEW language. The fuzzy controller is as follows. Fuzzy Logic is implemented based on a variation on set theory where a variable can partially be an element of a set. Fuzzy Logic Controllers are intended to "think" like humans do, in this way they are helpful for problems. This methodology is highly used in robotics and Artificial Intelligence field that cannot easily be set up mathematically, but can easily be expressed in words. Figure 8 illustrates Fuzzy Logic membership function for this scenario.

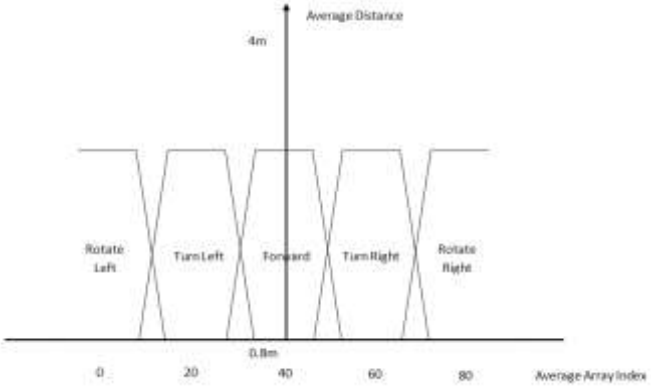

Fig. 8: Fuzzy Logic Membership Function

In this function both maximum depth point and average distance array is used when making a decision. When there is conflict between the Maximum Depth point array index and Average Maximum distance array index the decision is taken to rotate the robot until the conflict is solved.

This way the robot avoids going into tight spaces or going into a narrow path. This logic chooses the path which have most space (given by Maximum average distances array index) and longer path (given by Maximum Depth point). There is a little possibility of going into an infinite loop when the conflict is not lifted even the Robot rotates and scans 360 degrees. This is avoided by fuzzy logic by slightly moving robot into the direction where these two points are closer. Figure 9 and Table 1 illustrates directions that will be taken by the robot according to the aforementioned conditions.

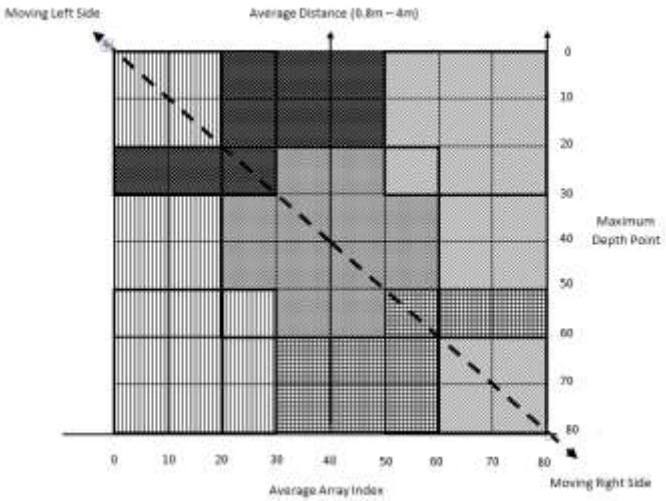

Fig. 9: Decision making with fuzzy logic

Table 1: Legend for fuzzy logic decision chart

Pattern	Description	Motor Driver Instruction
	Full Forward	Move all 4 wheels Forward in full Speed

	Turn Right	Move 2 Left side wheels Forward in full Speed and 2 Right side wheels in low Speed
	Rotate Right	Move 2 Left side wheels Forward and 2 Right side wheels in Backward
	Rotate Left	Move 2 Right side wheels Forward and 2 Left side wheels in Backward
	Turn Left	Move 2 Right side wheels Forward in full Speed and 2 Left side wheels in low Speed

Before giving commands to motor driver, the decision function then looks at current heading. The function makes sure the heading is always between -180 to +180 degrees. Rotation of the motors is controlled by constant time function which removes the requirement of encoders. What this function does is that the motors are allowed to rotate for a constant time at a constant speed and the robot will move at constant steps. This movement can be calibrated by the LabVIEW interface by giving X and Y increment of each step (forward step, Left rotation step or right rotation step). For the robot navigation during generation of occupancy grid is controlled by in built Leonardo arduino board of LattePanda. Figure 10 shows Part of Motor calibration.

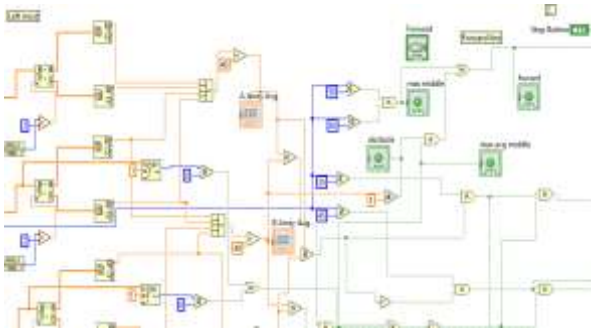

Fig. 10: Part of Motor calibration block diagram

The occupancy grid created can be used to generate the shortest paths using varies path planning algorithms. The main aim of the proposed solution is to plan the shortest path to the goal using an appropriate and efficient path planning algorithm. Once the 2D projection of map is completed, LabVIEW navigation stack will be executed according to the goal which is initialized by the user. Navigation stack will compute the shorted path using A-Star path planning algorithm. Here the path planning logic is an evolutionary approach since it has the ability to recalculate the path to the goal accordingly, after detecting dynamic obstacles.

The key advantages of this proposed solution is easy installation, human computer interaction and low cost. The final build package of the solution can be deployed/ installed into a single board computer using exe file.

4. Experimental Design

Most of the existing mobile robots are designed by using sensors like sonar and laser range finders. It's a known factor that, raspberry-pi circuit modules are increasingly used in robotics field. These modules support Ubuntu/Linux platform and Robot Operating System (ROS).

RTAB-Map (Real Time Appearance Based Mapping) library is used with its stand-alone application to create 3D cloud map. Point Cloud Library and RTAB-Map inherited vision based robots are available in the present. However, there are very few applications in robot development which support windows platform. RTAB-Map stand-alone application supports windows platform whereas RVIZ(ROS Visualization) stand-alone application doesn't support windows platform. Initially, implementation started with RTAB-Map approach. However, most of the libraries do not compatible with windows 10 platform. So that, LabVIEW was the next best alternative that can be used with Windows 10 platforms. LabVIEW is system engineering software which supports hardware and software integration. LabVIEW has a compatible version for windows 10 platform.

Therefore, the proposed solution will support windows platform and it uses the LattePanda single board computer instead raspberry-pi. Not only that, this robot is able to project moving obstacles and avoid those while navigation. And as additional feature, it has ability to access remotely to occupancy grid. This is a special feature and user can analyze the path by predicting extra obstacles on the way. Most of the existing systems use encoders. But here, instead of encoders, entire logic was implemented by using definite time controls. Most of the functions were controlled programmatically here. Therefore, the cost incurred for development of this robot is lesser than existing systems.

Inertial Measurement Unit consists, the accelerometer and the gyro sensor results. When gaining the results as the raw data considering motion angles and the acceleration forces, accuracy of the raw data cannot ensure since there are lot of noises happen with the electromechanical devices and the environmental influences. Therefore, the IMU raw data is experimentally calibrated into most accurate level. Following Figure 11 shows how to calibrate IMU raw data using the LabVIEW application.

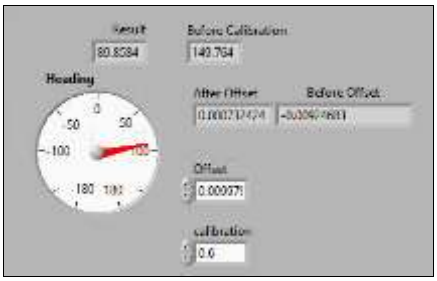

Fig. 11: 90 degree IMU calibration

The Table 1 shows sample of the data which analyze the experimental design by evaluating how calibration results going to match with the actual motion angles.

Table 2: Evaluation of the Accuracy in Inertial Measurement Unit

Motion Angle	Inertial Measurement Unit	Calibration	Result
90	149.764	0.6	89.8584
180	298.109	0.6	178.865

The calibration results will maintain the accuracy of the Inertial Measurement Unit according calibration factor manually inserted by the user through experimentation with different angles. This is an essential experiment which affects the overall robot's performance because robots orientation data solely depends on Inertial Measurement Unit data. This could be avoided using absolute encoders.

Here the main investigation has done for A-Star path planning algorithm. Following Table 2 shows two scenarios of A-Star Path Planning. First of Grid was created on a Text file. For both scenarios same grid is used.

Table 3: Evaluation On A-Star Path Planning Algorithm For Two Test Cases

A-Star Path Planning Algorithm Scenario 1	States	Path Time(ms)	Path Length (Cost)
Start –(3,3) Goal-(32,32)	362	0.7641527	56.3848
Start –(4,6) Goal-(33,12)	354	0.7709662	55.3848

The robot which was implemented for this research is shown in following Figure 12. It is a four wheeled robot which is in width of 30 cm and length of 30 cm.



Fig. 4: Front View of Robot

5. Conclusion

The robot was a success after many experiments and try outs. The most crucial development was the calibration of the Inertial Measurement Unit sensor. After many experiments and rigorous calibration it was able to minimize the noise. But since there is integration and accumulation of angles, the error of measurement is also accumulating. Generation of occupancy grid is very straight forward but depends of the accuracy of the Kinect sensor. Kinect sensor has a range which it can work properly for distances more or less than this range the robot cannot depend on the Kinect sensor. Because of that conflicts and collisions can be happened since Kinect's inability to track obstacles within first 10 cm of distance in front of it. Kinect range is about 95 degrees. Therefore, this is also a limitation of Kinect. And also its tilt motor can be rotated only vertically. The proposed solution is only for indoor environments with a flat surface. For outdoor environments Kinect is not suitable for capturing real environment. This kind of robot can be used to analyze a hidden place of a building or danger area where humans cannot reach. For an example, nuclear power plants etc. So that outsiders can view the unreachable environment and have a map. For future reference these maps can be saved and used by defining goals.

As an alternative platform based on windows the robot perform very well. It has all the advantages of the windows based system. With this solid foundation the robot can be improved much further.

6. Further Work

Currently the robots main inputs are the Kinect vision and the IMU data. The first step of further development would be to improve orientation data by having encoders alongside with IMU sensor. This can remove the rigorous calibration of the robot and the errors originated from the IMU sensor. Navigation can be improved by addressing different ranges through different sensors. For example Close range obstacle avoidance can be tackled through sonar or Inferred sensors. Medium range localization and visual navigation could be handled by the Kinect sensor. For long range Laser Range Finders could be used. By combining these sensors, navigation errors could be minimized. For the path planning algorithms, it can be clearly said that A-Star algorithm can deliver optimal path for the solution. However, A-Star is not efficient in all the scenarios. In such scenarios AD-Star is more reliable and efficient. The main constraint of implementing AD-Star is its complexity and required quality level of input feed to generate grid. By using stereo vision camera and Laser Range Finders input can be enhanced. In this proposed solution, LattePanda 32 bit version has used. To enhance performance it is better to use 64 bit version with a higher Memory. This can also improve the occupancy grid array size and also the ability to work with the other range sensors simultaneously to smoothen the process.

Acknowledgement

We hereby warmly thank everyone who helped developing this project and all the people who made the technologies which made this project possible.

References

[1] Hsu, C., Chen, Y., & Lu, M. (2012). Optimal Path planning Incorporating Global and Local Search for Mobile Robots, (1), 668-671.
[2] Bruno Siciliano, OussamaKhatib, FransGroen: Robot Navigation from Nature Simultaneous Localization, Mapping, and Path Planning Based on Hippocampal Models. Springer Tracts in Advanced Robotics Volume 41 ISSN 1610-7438. Springer-Verlag Berlin Heidelberg, 2008.
[3] TT Nguyen, IEEE Transactions on Evolutionary Computation, Vol. 16, 2012.
[4] Roland Siegwart, Illah R. Nourbakhsh, Introduction to Autonomous Mobile Robots, Massachusetts Institute of Technology, 2004.
[5] Fiorini P, Shiller Z (1998). Motion planning in dynamic environments using velocity obstacles. Int. J. Robot. Res., 17(7): 760-772.
[6] Open Kinect, 'Imaging Information', 2013. [Online]. Available: https://openkinect.org/wiki/Imaging_Information. [Accessed: 12-Oct- 2018].