Use of Collaborative Technologies for Building an Autonomous Unmanned Ground Vehicle (UGV)

Faizan Kazi, Hassan Najeeb and Faisal Iradat
Faculty of Computer Science,
Institute of Business Administration (IBA),
Karachi, Pakistan
{faizan@faizan-kazi.com, hn.najeeb@gmail,com, firadat@iba.edu.pk}

***Abstract*—In this paper we present a model for an autonomous Unmanned Ground Vehicle (UGV). Similar to existing UGVs, the proposed UGV can be assigned GPS waypoints to navigate on a specified path to reach destination. However, the UGV is unique in the sense that it is intelligently controlled by an Android device together with its onboard sensors and a Microsoft Kinect depth sensing input device to navigate and avoid obstacles and a Microcontroller to control the motors. Simultaneously, long distance communications help relay commands to the vehicle and receive status updates for any number of vehicles around the world. To visualize the real implementation, a simulator has also been developed and presented. Due to design simplicity, the proposed UGV can be used for several military and rescue applications.**

***Index Terms*—UGV, Android, Kinect, Microcontroller, GPS, IMU.**

I. Introduction

To reduce life-risk factors during military and rescue operations, unmanned vehicles minimize the presence of humans in hostile and insecure environments. According to the U.S military [1], during the period of 7 years (2005-2012) 792 lives were saved by employing robots in combat, thereby demonstrating the effectiveness of the new technology. With advancements in unmanned vehicle technology, new methods for controlling and guidance techniques have to be implemented for their effective use.

Unmanned vehicles can operate either on ground or in air. Further, these vehicles operate without a human driver and are either controlled autonomously [2] or semi-autonomously [3]. Consequently, the objective is to establish a path for the vehicle to traverse. A semi-autonomous multi-modal approach [4] has been proposed where the path is determined employing user defined waypoints. Due to significant user interaction, the accuracy in determining the path is not satisfactory - approximately 63%. In unmanned vehicles, fully autonomous control may provide better accuracy if appropriately modeled. The fully autonomous approach has been significantly explored in the past in order to reduce or minimize response delays and errors. However, to the best of our understanding no one has yet been able to control a UGV in fully autonomous mode through an Android phone and Microsoft Kinect. Recently, Android and Kinect have been used for enabling user interactions primarily for gaming [5] and augmented reality based applications [6].

In this paper a fully autonomous controlled unmanned ground vehicle is presented using collaborative technologies for navigation and obstacle avoidance. The model presented is more economically feasible and requires minimal human interaction. The rest of the paper is organized as follows: In Section II, an overview of the UGV is presented followed by the central controller detailed in Section III. The mission controller is given in Section IV. In Section V the hardware controller is presented. A simulator is presented in Section VI. Limitations are discussed in Section VII. Finally, there is a statement of limitations in Section VIII and the conclusion in Section IX.

II. Overview Of The UGV

The vehicle comprises of some major components, each independent of the rest in the sense that the interior workings of the components can be changed, as long as the communications protocol remains the same. The components consist of Server Program, Android program and Motor control.

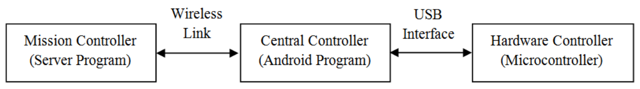

Mission Controller (Server Program)
Wireless Link
Central Controller (Android Program)
USB Interface
Hardware Controller (Microcontroller)

Figure 1- Modules with interrelationships

Each component can be considered as a “black box” i.e. each module can be modified and even replaced with an alternative solution, as long as the overall function and intercommunication protocol remains same.

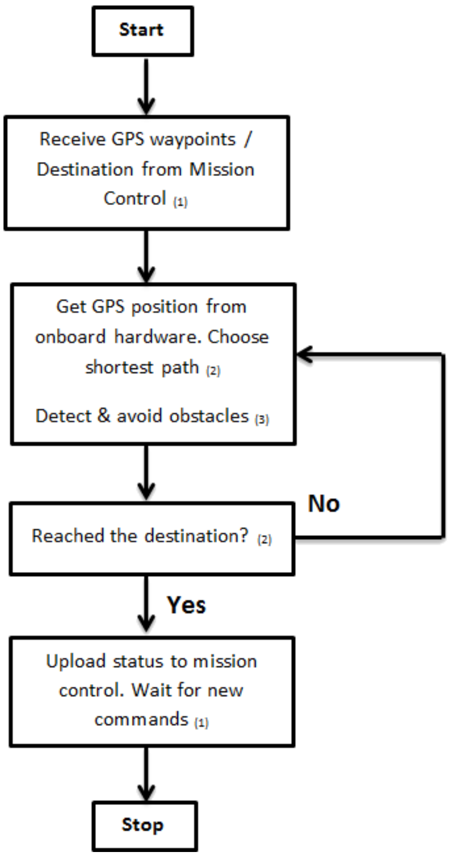
The mission controller (server program) allows us to monitor vehicles and assign routes and destinations. The central controller (Android program) resides in the UGV and uses sensor data to make decisions regarding navigation and obstacle avoidance. The hardware controller (microcontroller) controls the motors and external sensors such as ultrasonic range finders.

III. Central Controller (Android Program)

The Android device is the control center of our UGV. The Android device runs a native program written in Java. The program’s basic function is to control the vehicle and act as a communication hub.

The Android program does the following:

- Extracts data from the onboard sensors, including the Accelerometer, Gyroscope, Orientation and GPS data.
- Communicates with the Server and Motor Controller
- Intelligently decides how to reach the destination.


Start
Receive GPS waypoints / Destination from Mission Control (1)
Get GPS position from onboard hardware. Choose shortest path (2)
Detect & avoid obstacles (3)
Reached the destination? (2)
No
Yes
Upload status to mission control. Wait for new commands (1)
Stop

Flow Chart 1 - Overall Algorithm of the Android Program

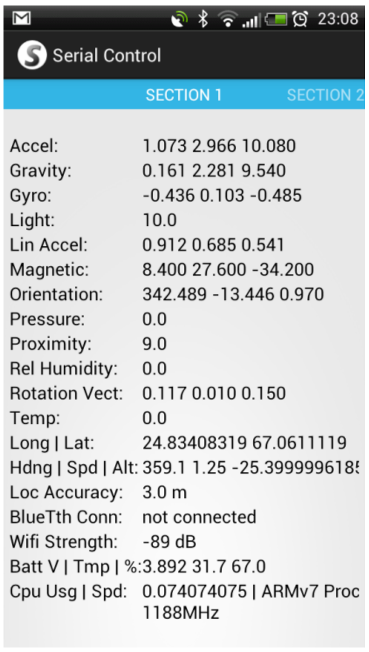
1) Wireless Communication with Server is over Bluetooth, GPRS based, etc.
2) Waypoint Navigation Algorithms
3) Obstacle Detection and Avoidance

A. Extraction of Sensor Data

The Android Device contains numerous sensors, of which we are only using a small subset relevant to our application. The sensor data that we have used includes [5]:

- Linear accelerometer which reports the acceleration force in m/s2, in all three physical axes (x, y, and z) excluding the force of gravity.
- Gyroscope which reports the device’s rate of rotation in radian/s around all three physical axes.
- Orientation which reports the device’s rotation around all three axes. These three items define Heading with respect to True North, Roll and Pitch. Roll and Pitch are useful in detecting chances of the UGV toppling, or attitude of a UAV in three dimensional space.
- GPS data which gives the Lat-Long (Latitude and Longitude) position of the vehicle, and the estimates of speed, bearing, altitude and most importantly the GPS information in meters.
- Depth sensor data including ultrasonic data and Kinect depth sensor data.

The screen shot of the retrieved data can be seen in Fig. 3 below.



```
Accel:             1.073 2.966 10.080
Gravity:           0.161 2.281 9.540
Gyro:              -0.436 0.103 -0.485
Light:             10.0
Lin Accel:         0.912 0.685 0.541
Magnetic:          8.400 27.600 -34.200
Orientation:       342.489 -13.446 0.970
Pressure:          0.0
Proximity:         9.0
Rel Humidity:      0.0
Rotation Vect:     0.117 0.010 0.150
Temp:              0.0
Long | Lat:        24.83408319 67.0611119
Hdng | Spd | Alt:  359.1 1.25 -25.399999618
Loc Accuracy:      3.0 m
BlueTth Conn:      not connected
Wifi Strength:     -89 dB
Batt V | Tmp | %:  3.892 31.7 67.0
Cpu Usg | Spd:     0.074074075 | ARMv7 Proc
                   1188MHz
```

Figure 2 - Screenshot from Android Program

This screenshot demonstrates live sensor values extracted from the Android Device’s onboard sensors.

B. Communications Intermediary

The Android program receives data from the server, including GPS coordinates of destination(s) and emergency manual override commands. It periodically sends data back to the server e.g. the sensor data and mission status information. Furthermore, it communicates final directional commands to the motors.

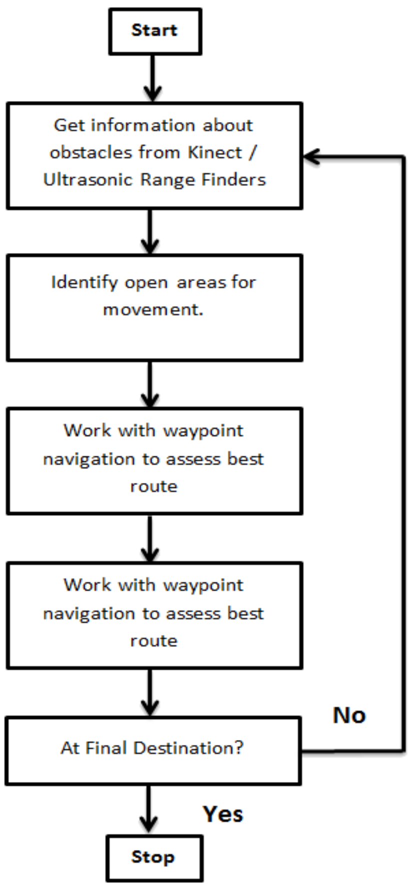
C. Decision Making Algorithms

The program uses the incoming data and computes the shortest path to take that has least resistance.

D. Path of Least Resistance

Currently, with our development in its initial stages, we are using a bump-and-go algorithm to detect the path of least resistance. For example, if the motors are pushing the UGV forward, and the linear acceleration sensors and speed calculations report zero values, then the vehicle is stuck, in which case it has to back out and try a new approach (track).

Once implemented, the ultrasonic depth sensors and Kinect depth sensor will provide real time depth data. The obstacle detection algorithm will then help pro-actively to avoid obstacles rather than running into them and then figuring a way out. Also, the gradient of ground surfaces will factor into choosing the path with the gentlest slope. The flowchart of the program is shown below.


Start
Get information about obstacles from Kinect / Ultrasonic Range Finders
Identify open areas for movement.
Work with waypoint navigation to assess best route
Work with waypoint navigation to assess best route
At Final Destination?
No
Yes
Stop

Flow chart 2 - Obstacle detection and avoidance algorithm

E. Navigation

This algorithm decides the expected heading between the current position and the destination. The purpose of this algorithm is to keep the vehicle travelling in a straight line. Ground vehicles tend to deviate from a straight line due to physical issues such as varying rotation of the individual wheels due to fluctuations in motor output, varying surface conditions such as bumps and differing traction on individual wheels.

IV. HARDWARE CONTROLLER (MICROCONTROLLER)

The microcontroller functions as a bridge between the Android program and motors and external electronics. The Android program provides the microcontroller the speed and direction to run the motors accordingly, whereas the microcontroller manages the electronic details.

The microcontroller controls the motors with the help of an amplifier circuit which we call the motor control board. The motors draws a lot of power while running, which the microcontroller is unable to provide. Therefore, to overcome this problem we use the motor control chip (L298N) dual motor driver (two motors can be attached) for each chip.

The ultrasonic depth sensors are also connected to the microcontroller which provides depth information to the Android program.

An Android IOIO Microcontroller is being used to implement the logic. The IOIO microcontroller communicates with the Android device over USB interface, with the ability to connect wirelessly via Bluetooth. The Android IOIO libraries are used to maintain communication with the microcontroller while it is plugged to the Android device.

V. MISSION CONTROLLER (SERVER PROGRAM)

The Server program is the equivalent of Mission Control, where the user can view the real time status of any number of vehicles in the field. The user also has the ability to control the destination, waypoints and routing of any particular UGVs.

The server collects data, charts it, logs it, offers emergency manual overrides and plots the position and status of vehicles on Google maps.

Data received from the remote Android program includes sensor data and mission status. The rate of data “packets” received per second is displayed. We are currently maintaining a data rate of more than 4 packets per second so that the data is grouped closely together. This will allow future data mining and in depth analysis of the data so that nuances and trends in sensor data readings can be observed.

Another page in the program shows us Google Maps, and display the real time position and heading of vehicles. It also helps visualize the accuracy of the latest GPS reading from a vehicle with a circle drawn around the estimated position. Included is the ability to track the estimated distance to the destination and time it would take to reach.

With the help of the Google Maps API [11], street directions to destinations can be retrieved, where road and routing data is available on the Google servers. This is useful during initial testing to make sure the vehicle is road worthy and gets to the destination with the best street route selected if one exists.

New Directions and Waypoints can be assigned to vehicles on the fly with a simple two click procedure by clicking on the map and assigning that point a new destination / waypoint.

The Server program also features emergency manual override to stop or redirect the vehicle as required. This is done with regards to standard UGV implementations [12].

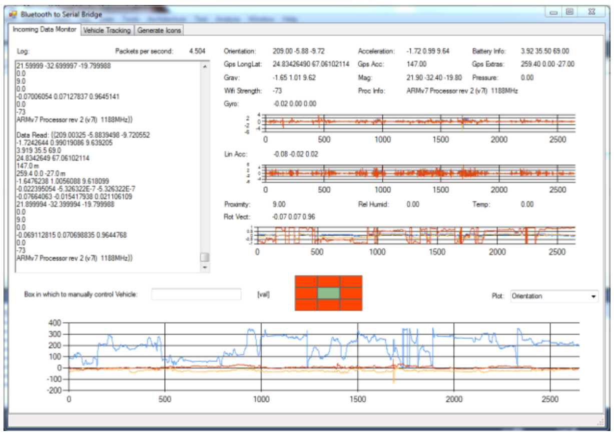
A. Logging and Charting of Sensor Data

The screenshot (see Fig. 3) from the Server Program shows raw logging of data on the left hand side. Packets received per second are displayed above the logging box. Parsed current data, along with line charts of selected items such as gyro, linear acceleration and rotation vector data can also be viewed on the right side.

The large graph at the bottom is configurable, and the data to be plotted can be selected from the dropdown above the chart on the right.

All the graphs contain sequential numbered data points on the x-axis, with their respective sensor values on the y-axis in terms of m/s^2 or radian / s.

An empty text box in the bottom half can be selected, and any key input from the left, right, up or down keys will make the Vehicle enter manual override mode. Similarly pointing the mouse over the orange boxes will make the vehicle move forward and left, forward, forward and right, right, back and right, back, back and left and left in clockwise order starting from top left. The green box at the center stops all motor movement. Keyboard number pad keys can also be used for gaining quick control over the vehicle.



Log: Packets per second: 4.504

```
21.59999 -32.699997 -19.799988
0.0
9.0
0.0
-0.07006054 0.07127837 0.9645141
0.0
-73
ARMv7 Processor rev 2 (v7l)  1188MHz])

Data Read: ([209.00325 -5.8839498 -9.720552
-1.7242644 0.99019086 9.639205
3.919 35.5 69.0
24.8342649 67.06102114
147.0 m
259.4 0.0 -27.0 m
-1.6476238 1.0056088 9.618099
-0.022395054 -5.326322E-7 -5.326322E-7
-0.07664063 -0.015417938 0.021106109
21.899994 -32.399994 -19.799988
0.0
9.0
0.0
-0.069112815 0.070698835 0.9644768
0.0
-73
ARMv7 Processor rev 2 (v7l)  1188MHz])
```

Orientation: 209.00 -5.88 -9.72
Acceleration: -1.72 0.99 9.64
Battery Info: 3.92 35.50 69.00
Gps LongLat: 24.83426490 67.06102114
Gps Acc: 147.00
Gps Extras: 259.40 0.00 -27.00
Grav: -1.65 1.01 9.62
Mag: 21.90 -32.40 -19.80
Pressure: 0.00
Wifi Strength: -73
Proc Info: ARMv7 Processor rev 2 (v7l) 1188MHz
Gyro: -0.02 0.00 0.00
Lin Acc: -0.08 -0.02 0.02
Proximity: 9.00
Rel Humid: 0.00
Temp: 0.00
Rot Vect: -0.07 0.07 0.96

Box in which to manually control Vehicle: [val]
Plot: Orientation

Figure 3 - Vehicle Data Logging and Charting

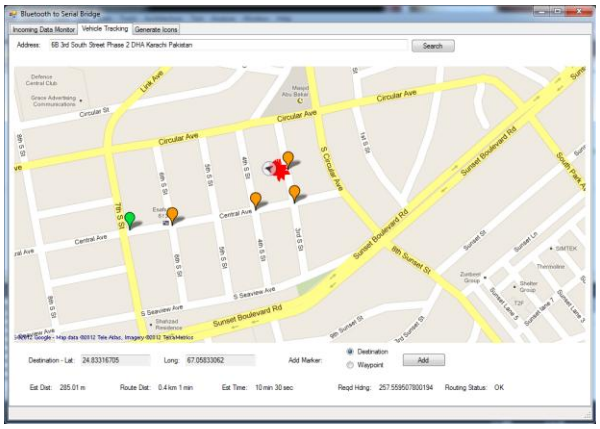
B. Real-time Mapping and Routing of Vehicle

The screenshot (see Fig. 4) from the server program shows Google Maps plot. Double clicking the Map updates the destination latitude and longitude boxes shown at the bottom left of the figure. The selected point can then be added either as a waypoint or destination by selecting the appropriate radio button and clicking the Add button.

In case the destination change, the Google Maps API is queried for street routing directions if available, and are used to help guide the vehicle. In this case, Google Maps routing was used, and Waypoints are shown in Orange, and the Destination in Green.

The position and heading of the vehicle is represented by the small black arrow. The circle around the arrow represents the GPS reading's accuracy. Meanwhile, the red area shows the area the vehicle has moved earlier.

Numerical data at the bottom represents the straight line distance from the current position to the destination, route distance and time from Google Maps, estimated time to destination at the current speed reported by GPS and required heading to destination.



Address: 6B 3rd South Street Phase 2 DHA Karachi Pakistan

Destination - Lat: 24.83316705 Long: 67.05833062 Add Marker: [✓ Destination] Waypoint

Est Dist: 285.01 m Route Dist: 0.4 km 1 min Est Time: 10 min 30 sec Reqd Hdng: 257.559507800194 Routing Status: OK

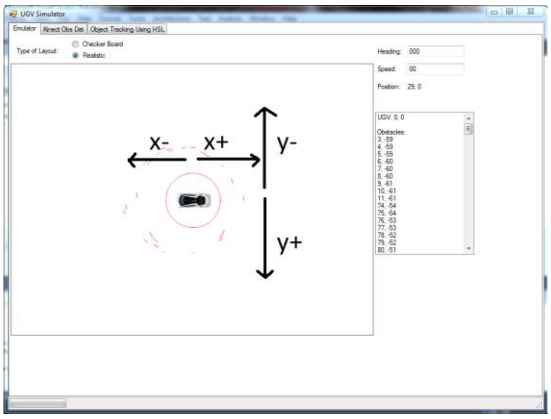
Figure 4 - Server Program showing Mapping

VI. SIMULATOR

The Simulator that has been developed is a very basic one. It serves to generate random obstacles, represented by lines of varying length, tangential to the radar / depth sensor circle around the UGV represented by a red circle. The objective is to test and virtually visualize obstacle avoidance and navigation taking place through simulator. This will enable one to evaluate performance of the algorithms from the logs.

The position of the UGV stays constant in the center of the screen while the obstacles are drawn on the screen as the UGV moves. As soon as an obstacle comes within the range of the sensors, the UGV detects it and uses an obstacle avoidance mechanism to move further without crashing into it.

The simulator is built on top of graphics libraries provided by the .Net Framework APIs. The screenshot of the simulator is given in Fig. 5.



Type of Layout: Checker Board [✓ Realistic]

Heading: 000
Speed: 00
Position: 29, 0

```
UGV: 0, 0

Obstacles:
3, -59
4, -59
5, -59
6, -60
7, -60
8, -60
9, -61
10, -61
11, -61
74, -54
75, -54
76, -53
77, -53
78, -52
79, -52
80, -51
```

Figure 5 - Simulator Program demonstrating randomly generated obstacles

VII. LIMITATIONS

Currently the UGV is using the bump-and-go method described in Section III-D and is relying heavily on Google Maps to reach its destination. Eventually once the Kinect depth sensor is implemented, real time depth data can be gathered. It will then be possible to detect obstacles and avoid them rather than running into them. The UGV would be able to move around in an unknown terrain using depth map data without crashing into any obstacle.

Although as mentioned in [7] the Kinect sensor cannot detect objects while it is moving itself. To address this issue, the UGV will stop or slow down to a very low speed at which we can record the depth map and then make an intelligent decision to move ahead.

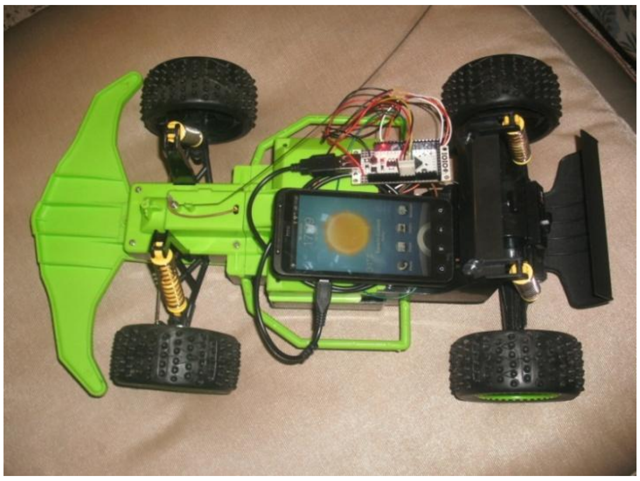


Figure 6 – Aerial view of our modified remote control car based test bed.

VIII. Future Work

Our current research includes integrating Kinect depth sensor with the UGV. The Kinect sensor is used to detect obstacles in real time. Kinect is a cheap sensor that provides us with depth map images, with a field of view of 57 degrees and maximum range of 3.5 meters [10].

The UGV will use the Kinect to obtain depth data while performing a mission. The depth data will be processed to obtain an obstacle-free path for the UGV to travel [7].

IX. Conclusion

In this paper we present a small scale project to test the idea of using an Android device to operate an unmanned vehicle autonomously. The proposed UGV will be able to perform risky tasks of rescue, supply, surveillance and military operations (exploring unknown terrain including mine-filled areas) and would be cost-effective and efficient as well. Future work includes development of an Unmanned Air Vehicle (UAV) or an autonomous drone that would be very cost effective for a number of applications where human reach is not possible due to a hostile and insecure environment.

References

[1] US Military, “Robotic Systems Joint Project Office (RS JPO).” 2012.

[2] S. Parsons, “Autonomous Robots: From Biological Inspiration to Implementation and Control by George A. Bekey, MIT Press, ISBN 0-262-02578-7,” Knowl. Eng. Rev., vol. 20, no. 2, pp. 197–198, 2005.

[3] M. M. Rohde, V. E. Perlin, K. D. Iagnemma, R. M. Lupa, S. M. Rohde, J. Overholt, and G. Fiorani, “PointCom: semi-autonomous UGV control with intuitive interface,” p. 69620G–69620G, Apr. 2008.

[4] S. Mortezapoor, M. Taale, and S. Soleimani, “A Multi Modal Interface Approach to Control an Unmanned Aerial Vehicle,” University of Fribourg, Switzerland, 2012.

[5] R. Meng, J. Isenhower, C. Qin, and S. Nelakuditi, “Can smartphone sensors enhance kinect experience?,” in Proceedings of the thirteenth ACM international symposium on Mobile Ad Hoc Networking and Computing, Hilton Head, South Carolina, USA, 2012, pp. 265–266.

[6] S. Warade, J. Aghav, P. Claude, and S. Udayagiri, “Real Time Detection and Tracking with Kinect,” in Proceedings of the ICCIT 2012, Bangkok, Thailand, 2012, pp. pp. 86–89.

[7] A. Khan, F. Moideen, J. Lopez, W. L. Khoo, and Z. Zhu, “KinDectect: kinect detecting objects,” in Proceedings of the 13th international conference on Computers Helping People with Special Needs - Volume Part II, Linz, Austria, 2012, pp. 588–595.

[8] B. Wei, J. Gao, K. Li, Y. Fan, X. Gao, and B. Gao, “Indoor mobile robot obstacle detection based on linear structured light vision system,” in Proceedings of the 2008 IEEE International Conference on Robotics and Biomimetics, 2009, pp. 834–839.

[9] C. Zheng and R. Green, “Feature Recognition and Obstacle Detection for Drive Assistance in Indoor Environments,” University of Canterbury, Christchurch, New Zealand.

[10] http://kotaku.com/5576002/here-are-kinects-technical-specs, accessed 26 October 2012

[11] https://developers.google.com/maps/

[12] Intelligent Ground Vehicle Competition Rules http://www.igvc.org/DRAFTIGVCRules2013DRAFT.docx