Design and Development of Unmanned Ground Vehicle for Implementation in Urban Roads

Pingakshya Goswami
Department of ECE
Tezpur University, Assam

Swaraj Boishya
Department of ECE
Tezpur University, Assam

Rashmi Ranjan Sahoo
AEDS Lab
Jadavpur University,

M.K. Naskar
IEEE Senior Member
AEDS Lab
Jadavpur University

ABSTRACT
In the present day world, a lot of research has been going on in the field of unmanned vehicles, which includes unmanned ground vehicles (UGVs), unmanned aerial vehicles(UAVs) and unmanned water vehicles (UWVs). The UGV technology grows rapidly. Generally UGV is developed for military purpose. Now a days, many universities and research institute are researching in new UGV technology for commercial use such as transportation service. The UGV system generally consists of four parts such as vehicle control system, navigation system, and obstacle detecting system and traffic signal monitoring and detection system. In this project, we have designed and implemented a prototype of UGV using PIC 16F57 microcontroller on board the Board of Education (BOE) development board designed by Parallax Inc, USA. This UGV prototype was used to simulate acts performed by a real car on urban environment which include navigation, obstacle avoidance, collision avoidance at intersection of two roads, traffic signal detection and feed backing the current co-ordinates of the vehicle to the control room for tracking of the vehicle.

For implementing the navigation part of the robot, a novel algorithm was adapted which we named as *r*-θ *(r-theta*) algorithm which is explained latter in this paper.

Keywords: digital map, path planning, *r*-θ *(r-theta*) algorithm, UGV

1. INTRODUCTION
Since computer systems were introduced in the world, a lot of scientists have been engaged in creating autonomous robots or vehicles to explore unknown worlds. Some UGVs have been used for space exploration – exploring planets like the Saturn and Mars. Since 2004, the Defense Advanced Research Projects Agency (DARPA) of USA introduced a competition for autonomous vehicles, still continuing to hold this competition until today. This makes people highly motivated to create UGVs, and, considering that driverless vehicles are required more in our life in many fields, each DARPA Challenge is a great opportunity to develop unmanned vehicle systems [1,2].
In India very few research institutes are doing researches in the field of real time UGV although prototypes are developed in many labs. Recently in September, 2010 Defence Research and Development Organisation (DRDO) India in association with Defence Research and Technology Office (DRTO), Singapore, have together developed an unmanned ground vehicle that can be used during nuclear blasts and wars for a reconnaissance to help find injured people. The vehicle was developed using a Honda CRV sports utility vehicle [3].

Although UGVs for defence applications are developed in India, however development of UGVs for applications on real road in urban environment is very limited. In this project we have developed and implemented the prototype of an UGV for its applications in real urban environment using a Parallax Inc. manufactured mobile platform named BOE BOT. We have proposed a novel path planning algorithm, *r-θ algorithm* for efficient navigation of the vehicle from source to destination.

2. HARDWARE OVERVIEW
In this project we used Parallax made Boe-bot mobile robotic platform. This is a three wheeled platform which has a PIC16F57 microcontroller embedded on its board. For communication with other robot as well as with the PC, we used an EmbeddedBlue500 (eb500) Bluetooth module.

3. PATH PLANNING OF ROBOTS USING DIGITAL MAP
Path planning is an important issue as it allows a robot to move from point A to point B. Many people have been doing research in the field of path planning and many algorithms have been proposed. Anthony Stentz, from The Robotics Institute; Carnegie Mellon University proposed an algorithm known as D* - algorithm in which if the robot does not know the exact environment can also moved from the source to destination [4]. Some scientists have also proposed algorithms namely Lazy Theta* for shortest path calculation [5].

In this project we have implemented a novel algorithm which was named as *r-θ algorithm*. In this algorithm, if we provide the source(S) and destination (D) to the robot, it will scan the path from S to D and will generate a piecewise linear digital map is generated in terms of (r_i, θ_i) parameters, where r_i is the length of the path the robot traverses at *i th* go and θ_i is the angle the robot has to turn before covering r_i distance. Fig. 1 below shows the original path given to the UGV to traverse from S to D in blue and the converted piecewise linear digital map is shown in pink.

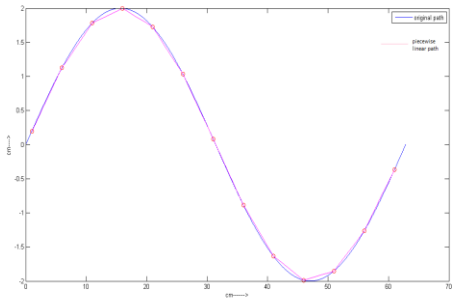


Fig. 1 Original path and piecewise linear path

Suppose $S(x_1,y_1)$ represents the source co-ordinate of the robot and D (x_3,y_3) be the destination co-ordinate of the robot. Fig. 5 below shows a simple map of the source and destination of a robot. As shown in Fig. 2 below, the path passes through three nodes S, P and D.

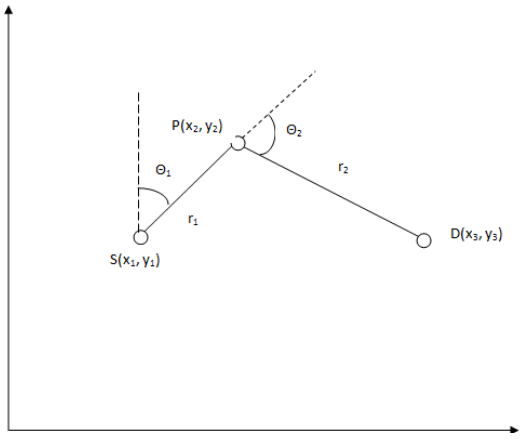


Fig. 2 Simple map showing source and destination of the robot

For solving this path through the r-θ algorithm the length of the path r_i and the turning angles θ_i are calculated by the following formulae:

$$r = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \qquad (i)$$

$$\theta_i = tan^{-1}\left(\frac{y_{i+2}-y_{i+1}}{x_{i+2}-x_{i+1}}\right) - tan^{-1}\left(\frac{y_{i+1}-y_i}{x_{i+1}-x_i}\right) \qquad (ii)$$

By using *(i)* and *(ii)* we will calculate the r and θ for a single turn. If there is N number of turns in the path, then the above formulae are repeated for N times for getting the entire digital map in r-θ format. In this paper every right turn is considered as positive while every left turn is considered as negative angle.
The (r_i, θ_i) co-ordinates are fed into the EEPROM memory of the UGV and it will travel from source to destination seamlessly provided there is no obstacle in the path. If the robot faces any obstacle in its path, a different approach has to be taken so that the robot continues to move towards its destination. This approach is discussed in detail in Section IV of this paper.

4. PATH PLANNING IN PRESENCE OF OBSTACLE

As already mentioned in the previous section, if the vehicle faces an obstacle in front of it, it cannot follow its original path. For the purpose detecting the obstacles, the robot is fitted with infra-red (IR) transmitters and receivers. When the robot faces an obstacle in front of it, then the ports of the BS2 microcontroller to which the IR receivers are connected becomes LOW (0). The IR receivers we used in this project is a HS0038 IR receiver module. In order to make this receiver work, we emitted IR wave of frequency 38KHz from the IR transmitter LEDs.

For our experimental purpose, we assume that the dimensions of all the obstacles placed on the path are same. We used boxes of 15cm x 20cm x 30cm as our obstacles. The robot was highly successful in avoiding the obstacles (even multiple obstacles) when it is placed both at the straight patch of the path as well as at the corners of the path.

4.1 Obstacle Avoidance in the Linear Patch of the Path

If the robot comes across an obstacle, the IR sensors detect the obstacle from a distance of about 5 cm. Here, we are referring the sensing distance as *s*. As soon as the IR receivers sense an obstacle, two of the input pins of the microcontroller become LOW. When the two particular pins become low, the onboard processor goes out of the main program to a subroutine which we named as GOSUB_TURN( ). Lumelsky [6] proposes a robot navigation algorithm in which the environment is initially assumed to be devoid of obstacles and moves the robot directly toward the goal. If an obstacle obstructs the path, the robot moves around the perimeter until the point on the obstacle nearest the goal is found. The robot then proceeds to move directly toward the goal again. But our approach is slightly different and effective from Lumelsky's algorithm. When the robot faces an obstacle, it moves around the perimeter of the object but instead of moving to the point on the obstacle nearest the goal, the robot searches for the original path. If it comes to the original path, it begins to follow the path and continues to move towards the destination through that path. In the GOSUB_TURN( ) subroutine, this step is done by the UGV. Fig. 3 shows the path followed by the robot when it faces the obstacle of above dimension for an ideal case.

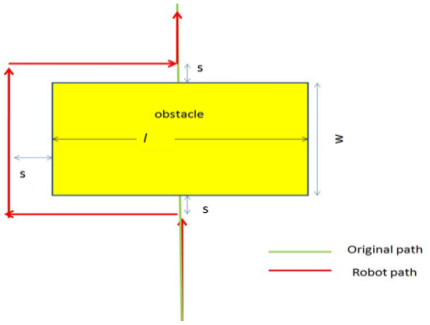


Fig. 3 Path followed by the robot in presence of obstacle in the linear path

4.2 Obstacle Avoidance at the Corner of the Path

If the obstacle is placed at the corner or at the turning of the path, the above convention, i.e. the approach for a straight is not followed. Here another question arises. How will the robot know whether the obstacle is placed at the linear portion or at the corner of the path? For this, a distance counter *(r-counter)* is running inside the onboard processor of the robot.

Suppose, *i* represents the current count of the distance counter and r_i represents the final count, i.e. the total distance to be covered in the *i* th turn. If at the time of obstacle detection, $i > r_i - (w/2+s)$, then the robot will assume that the obstacle is present at the corner. Here w is width of the obstacle and *s* is the minimum sensing distance from the obstacle. Fig. 4 illustrates this phenomenon diagrammatically.

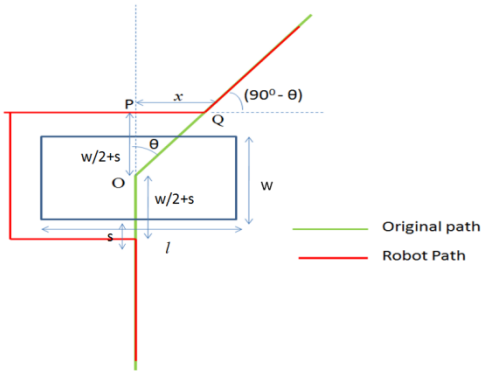


Fig. 4 Obstacle avoidance at positive corner (right turn)

As illustrated in Fig. 4, upto point P the vehicle follows the same path as the one discussed in Section 4.1. From this point, it traverses *x* distance before merging with the original path by taking a $(90^0-\theta)$ left turn. Considering our assumptions the distance *x* is calculated using the following equation:

$$x = (w/2+s)\tan\theta \qquad (iii)$$

After traversing this *x* distance, the robot takes a $(90^0-\theta)$ left turn to merge with the original path.

If the obstacle is placed at a negative angle i.e. at a left turn, following approach has to be taken to merge with the original path. This approach is illustrated in Fig. 5.

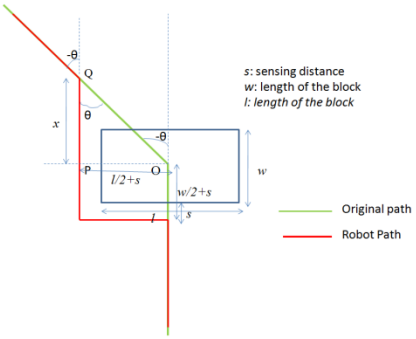


Fig. 5 Obstacle avoidance at negative corner (left turn)

As illustrated in Fig. 8, upto point P the vehicle follows the same path as the one discussed in Section 4.1. From this point, it traverses *x* distance before merging with the original path by taking a θ^0 left turn. Considering our assumptions the distance *x* is calculated using the following equation:

$$x = (l/2 +s)\tan\theta \qquad (iv)$$

After traversing this *x* distance, the robot takes a θ left turn to merge with the original path.

While experimenting with the three cases as discussed above, the robot was highly successful in overcoming the obstacles and reaching the destination.

4.3 Moving Obstacle Avoidance at the Intersection of Two Roads

Since the UGV which we have designed is modeled to operate on urban environment, one of the most important features which have to be made available to the vehicle is the moving obstacle detection part. This detection is performed using the same IR transmitter and receiver pairs used for the static obstacle detection part. Generally in the intersections of two roads, cars use to move in both directions in both the roads. In our UGV prototype, we have implemented a system in which if two cars are coming in both the roads then each of the cars senses the other car and then considering the priority of the roads and the cars. In order to detect the cars travelling in the other road, the cars are mounted with the IR sensor fitted a 45^0 to the car plane in both directions as shown in Fig. 6 and one is fitted at the center. By this way, the UGV is able to detect vehicles moving in both directions in the intersected road. Fig. 7 shows the positions of the two cars at four different instances of time and the response of the three sensors at each instance graphically.

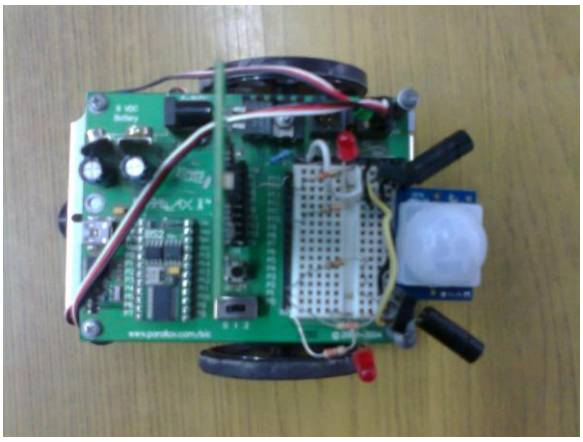


Fig. 6 Photo of boe-bot with sensors mounted at 45^0 to the bot

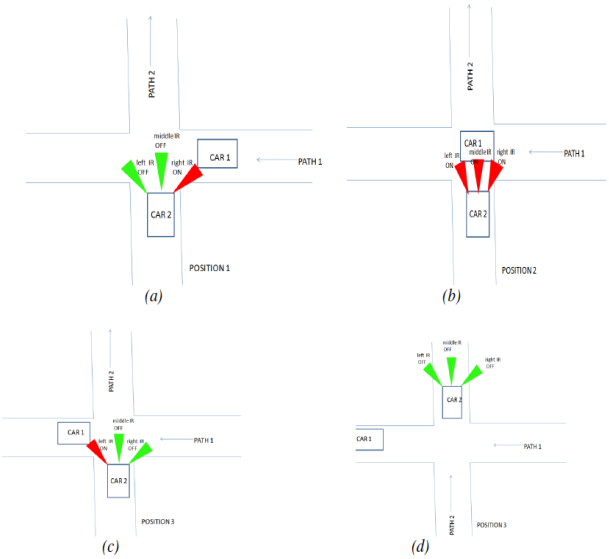


Fig. 7 Positions of the two cars at intersection at four different instances

In the above experiments, we consider that Path-1 has more priority than path-2. Hence any car moving in path-1 should be given the preference to pass through the intersection. Car-2 travelling in path-2 is mounted with three IR sensors. As shown in Fig. 7*(a)* if one of the corner sensors (right sensor in our case) of car-2 detects another car travelling towards it, i.e. it becomes HIGH, the car would stop there and allow Car-1 on Path-1 to pass. Now, how will Car-2 know whether Car-1 has passed or not? This is detected by the other sensor, i.e., the left sensor here. As soon as the left sensor becomes HIGH as shown in Fig. 7*(c)*, the on-board processor will signal the motors to start and the UGV will continue to proceed towards its destination. This process can also be applied to stop a vehicle on Path-2 when a platoon of many cars passes through Path-1.

The entire path planning algorithm of a UGV can be summarized with the help of the following flow-chart.

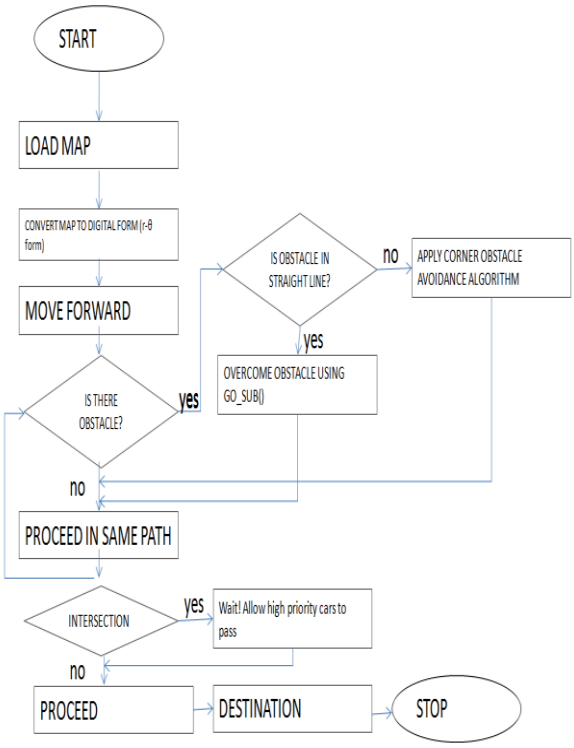


Fig. 8 Flow Chart showing the entire path planning using *r-θ algorithm*

5 POSITION TRACKING AND NAVIGATION CONTROL USING BLUETOOTH

In this project, we propose an algorithm in which the number of times the wheel rotates is sent to the control room computer using Bluetooth. From our experiments, we found out that in 243 counts the robot traverses 100cm. By applying simple unitary method calculations, the counts for any distance traversed can be found out. We sampled the distance counter after every 10 counts, i.e. after every 4.1cm using the on-board Basic Stamp2 module. This distance *(r)* along with the turning angle *(θ)* was send to the control room computer using the eb500 Bluetooth module fitted in the UGV. On the other hand, the control room computer is fitted with the eb600 adaptor (discussed in Section 2.4) in its serial port. In this module, the eb500 Bluetooth card is inserted. The control room computer Bluetooth module receives data send by the UGV. This data can be viewed in the Debug terminal of the Basic Stamp editor in the control room PC. Fig. 12a shows the screen shots of the Basic Stamp editor Debug terminal which shows the *r* and *θ* values. It can be seen from the debug terminal data (Fig. 12a) that the distance is sent after every 10 counts while the angle is sent to the control room whenever the Boe-Bot makes a turn greater than 15^0. In Fig. 12b, the angle turned by the robot and the total distance traversed after each turn is displayed, i.e. only r and θ values are displayed

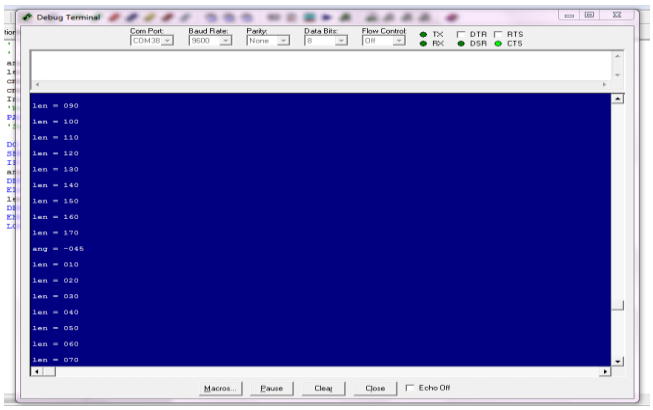
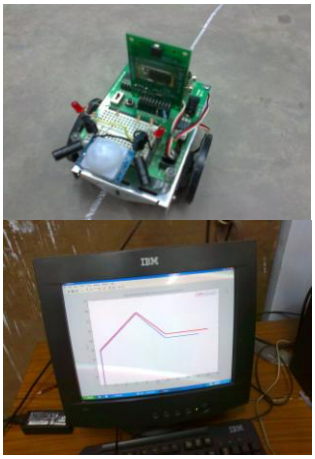


Fig. 9 Debug terminal showing r & θ values after 10 counts of r

5.1 Graphical Representation of Current Position of the UGV



Fig, 10 (a) Boe-bot sending current co-ordinate through Bluetooth

Fig. 10(b) Position is tracked in the control room PC

Fig. 10 shows the Boe-bot sending the current co-ordinates to the control room PC where it is displayed graphically. Since the receiver Bluetooth module is connected to the serial port of the control room PC, the data received by it can be easily accessed using Windows HyperTerminal as well by any COM port accessing software. The COM port can also be read using Matlab. The COM port values are read using Matlab and its co-ordinates are plotted graphically. A digital map of the area in which the robot will traverse is already plotted in the Matlab display panel. When the control room computer receives the r and θ values send by the remote UGV, the Matlab database gets updated and hence the corresponding point representing the UGV in the Matlab display panel moves accordingly. The outcome of this experiment is highly

successful and the results of this work is discussed and displayed in the result section (Section VII) of this paper.

The main purpose of position tracking of an UGV is to make the control room operator aware whether the vehicle is moving in its designed path or has deviated from its original path or it may face any danger. Sometimes it may also happen that the route in which the UGV is travelling is highly congested. In this case the operator may view the map of the UGV path and can change the path of the vehicle by sending some commands. In this project, we have also installed the facility by which the operator can control the navigation of the UGV in case if the vehicle arrives at the above mentioned situations. Since a Bluetooth connection is already exist between the UGV and the control room PC, the operator can send an interrupt request to the UGV processor. When the processor receives the interrupt request, it switches itself from data mode to command mode, i.e it begins to receives command from the operator PC. The operator then controls the navigation of the vehicle by sending commands and brings it to its desired location. After this the vehicle can again be switched from command mode to data mode and can be made to follow its original path after the correction.
By using this technique, we are able to control the motion of the Boe-bot successfully to a certain extent and can bring it back to its original path when it incurs any unusual situations.

6. SMART TRAFFIC CONTROL SYSTEM FOR UNMANNED AND MANNED GROUND VEHICLE

One of most important issue in research for unmanned ground vehicle is detecting signal lights at crossroads. In order to successfully ply in an urban environment, an UGV has to be equipped with some sort of technologies through which it will be able to detect the traffic signals at the intersection of two roads. Moreover the roads will also have to be installed with some signaling system which made it possible for the UGV for easy detection. Although many people are doing research in the field of traffic signaling for UGV, most of them use vision based system for detecting the traffic lights [8, 9]. Few groups have also implemented traffic signaling system using RF transmitter and receiver. But the main problem with real time traffic light systems using vision is that it sometimes detects the tail lamp of other cars as the traffic signals. Sometimes traffic signals at other junctions are also wrongly detected as the current signal. Also, a complex processor capable of performing filtering operations has to be installed in the UGV which becomes very much expensive. On the other hand, in case of RF based systems, same frequencies at the different roads of a junction will create interference at the car receiver.

In this project, we have designed a smart traffic controlling system for UGV which minimizes the above mentioned complexities. In our design, the traffic signals at the junctions will receive the signal from the control room (it may be manual or automatic) and convert it into both UGV readable and human readable forms. The traffic signal controller at each junction is equipped with a Bluetooth receiver. Here, we have used the Board of Education development board as our traffic signal controller. Whenever the traffic controller receives a stop signal from the control room, it turns ON a red signal (say an array of red LEDS) so that the human drivers can understand it indicates stop. On the other hand, for UGV purpose, the surface of the road near the junctions is fitted with IR LEDs. The IR LEDs are in turn connected to the Board of Education development board (the traffic controller). Whenever the controller receives a STOP signal from the control room, it turns ON the array of IR LEDs fitted on the road. The bases of the UGVs are also installed with IR receivers. When this IR receivers senses that the IR transmitters at the road surface are ON, it instructs the vehicle to stop and the UGV stops instantly. On the other hand when the IR transmitters are OFF, the UGV moves seamlessly.

This method of traffic signaling is a novel idea which is useful to both manned and unmanned vehicles. While doing the experiments, we find that the UGV stops as soon as it detects the glowing IR LEDs and continues to move when they are OFF.

7. RESULTS AND DISCUSSIONS

While doing this project, we used the Parallax made Boe-bot as our test vehicle. Due to the mobility constrains of the Boe-bot on rough terrain or on real roads, the entire experiment was carried out indoor only. However the basic concepts remain the same and hence it can be applied on real roads on urban environment. In our current work we have successfully developed a prototype unmanned ground vehicle capable of simulating the tasks perform by a real car on urban environment. The position feedback to the control room is still at experimental level and not fully implemented on real time cars.

The first experiment which we performed was the implementation of digital map. As discussed earlier, a path of the robot from source to destination is fed into the Boe-bot's EEPROM. The processor creates a digital map of the path in terms of r and θ and the vehicle traverses according to that path. In our experiments, we initially fed 330cm path into the Boe-bot's memory. Although the vehicle was successful in traversing from the source to destination, but there were some errors. It deviated from the actual destination by 6cm linearly and 3^0-4^0 angular deviation occurred. This error accounts to be 1.81%. Fig. 11(a) shows the path followed by the robot (shown in blue) in contrast over the actual path. We also done a path return test for the same path mentioned above, in which the robot moves from the source (S) to destination (D) and again returns back to the source(S) (S-D-S). In this case the robot successfully reaches D and retraces back to S. But here it deviated from the source by 13cm. i.e. 1.96% error takes place. Although the algorithm of the program is correct, the errors occur due to various mechanical and environmental reasons. One of the main reasons of the error is that the speed of the two servos are not equal. Calibrations were done to make the speed of both the servos equal. However some error existed due to small voltage difference at the servos. Moreover it was found out the diameter of the two wheels vary by 1mm. Sometimes the robot does not exactly turn to its programmed angle and varies by 4^0-5^0 due to the uneven floor surface.

The UGV was successful in detecting the obstacles placed both at the linear as well as the corner of the path. The Boe-bot detects the obstacles from a distance of 5cm. On detecting the obstacles, the processor goes to a subroutine known as GOSUB_TURN ( ). In this subroutine it performs all the processes required for overcoming the obstacles. After avoiding the obstacles the robot continues to move in its predefined path to reach its destination. Fig. 11(b) and (c) shows the path of the robot in presence of obstacles placed at the linear and corner portion of the path respectively.

In the third part of this project, we did the current co-ordinate feedback sending of the Boe-bot to the control room PC. The control room operator can view the current location of the vehicle and if the vehicle deviates from its original programmed path or faces some obstructions, the operator can correct the Boe-bot's path by sending commands over Bluetooth. The screenshots of control room PC are shown in Fig 11 below in absence of obstacle (Fig. 11(a)) and in presence of obstacle (Fig. 11(b) and (c)). In this experiment, the mobile Boe-bot sends its coordinate correctly upto 98%. So this experiment can be considered as a total success.

Finally, we designed the smart traffic control signal which signals both the manned and unmanned vehicles when to stop and when to go. The UGV was successful in detecting the traffic signals and follows the rule accordingly.

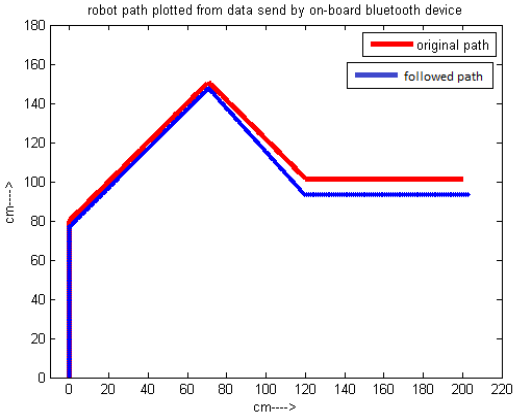


Fig. 11(a)

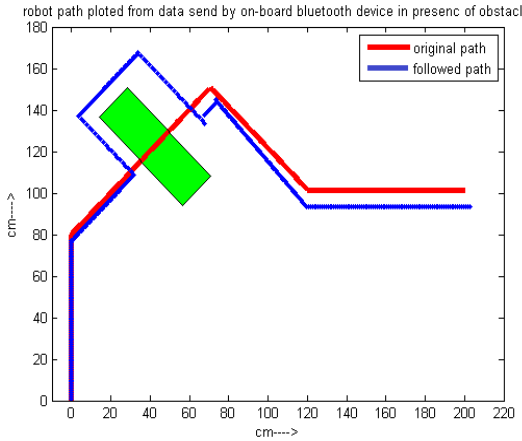


Fig. 11(b)

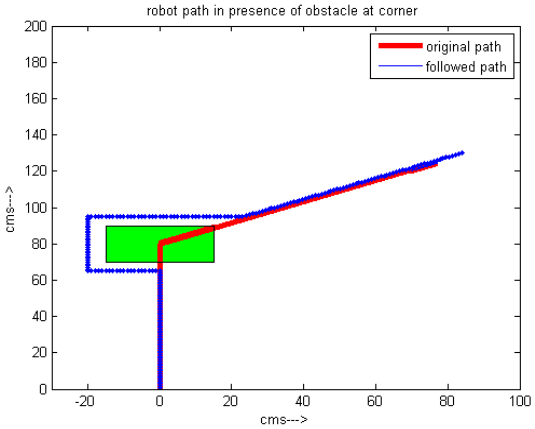


Fig.11 (c)

Fig. 11. Path followed by the robot (shown in blue) in absence of obstacle (a) and in presence of obstacle(b & c). The red line indicates the original path.

8. CONCLUSION

Through this project, we have developed a miniature UGV capable of performing the basic tasks a normal vehicle has to do. This includes navigation of the UGV from source to destination by creating a digital map in terms of r and θ. Moreover it is successful in avoiding static as well as moving obstacles in its path and then successfully reaching its destination. Although the project is done on a small model, it can be implemented on actual cars also as the basic concepts remains the same. For example, instead of transmitting data through Bluetooth, high frequency RF signal can be used to send the coordinates of the car to the control room. In this project, we have detected obstacles with the help of IR sensors. But camera or long range Lasers can be used to detect obstacles placed at longer distances.

This project is part of a main project which is started by Jadavpur University to implement a Vehicular Adhoc Network (VANET) in Kolkata, India. These small vehicles which we have designed will become the nodes of the VANET. Research is going on in our lab to implement the UGV on real road vehicles for testing in urban environment.

9. REFERENCES

[1] Seetharaman, Guna., Lakhotia, Arun., and Blasch, Erik Philip, "Unmanned Vehicles Come of Age: The DARPA Grand Challenge", *Computer*, Volume 39 Issue 12, December 2006.

[2] Dr. Peter Drewes, Brian Satterfield, and Heeten Choxi, "Sensor Technology and UGV Operations –Lessons Learned from the Urban Challenge".

[3] DRDO Develops Honda CRV Based Unmanned Ground Vehicle Prototype, *www.defenceforumindia.com/indian-army/15046-drdo-develops-honda-crv-based-unmanned-ground.html*.

[4] Anthony Stentz, "Optimal and Efficient Path Planning for Partially-Known Environments*", Proceedings IEEE International Conference on Robotics and Automation***,** May 1994.

[5] Alex Nash , Craig Toveyand Sven Koenig, "Lazy Theta*: Any-Angle Path Planning and Path Length Analysis in 3D*", National Conference on Artificial Intelligence - AAAI*, 2010.

[6] Lumelsky, V. J., Stepanov, A. A., "Dynamic Path Planning for a Mobile Automaton with Limited Information on the Environment", *IEEE Transactions on Automatic Contro*l, Vol.

[7] Borenstein, J., 1997, "Experimental Results from Internal Odometry Error Correction With the OmniMate Mobile Platform", *IEEE Transactions on Robotics and Automation*, Vol. 14, No. 6, December 1998, 963-96.

[8] Jin-Hyung Park, Chang-sung Jeong, "Real-time Signal Light Detection", *International Journal of Signal Processing, Image Processing and Pattern Recognition*, Vol.2, No.2, June 2009.

[9] Yun-Chung Chung et al, "A Vision-based Traffic Light Detection System at Intersections*", Journal of Taiwan Normal University: Mathematics, Science & Technology* 47(1), 67-86, 2002.