An Autonomous Off-Road Robot Based on Integrative Technologies

Orlando J. Hernandez
Electrical and Computer Engineering
The College of New Jersey
hernande@tcnj.edu

Yunfeng Wang
Mechanical Engineering
The College of New Jersey
jwang@tcnj.edu

Abstract

This paper presents an autonomous off-road robot designed by converging highly reliable integrative technologies. The mechanical, electrical, and computing platforms of the robot are highly stable and easy to configure and can be used as a general robotics prototyping infrastructure for unmanned ground vehicle. Part of the design is an interactive graphical simulation interface that proved to correlate well with field trials. Field experiments have shown that the system is highly efficient.

1. Introduction

The subjects on autonomous mobile robots have been studied for several decades. It is only until recently that their practical applications for civilian and military purposes have become highly possible and regain people's attention due to the dramatic improvements on computing and sensing technology [1]. To accelerate research and development in autonomous mobile robots, The Defense Advanced Research Projects Agency (DARPA) of U.S.A initiated Grand Challenge in 2004, which is an annual autonomous-ground-vehicle race with $2 million prize for the winner [2]. Significant results and improvements have been achieved and demonstrated under the stimulation of this Challenge. As proved by the failure race of 2004 DARPA Grand Challenge, building a functional autonomous mobile robot involves many challenging issues on design, sensing, localization, navigation, planning, control, and decision making. Many scholarly work have been focused on the algorithmic issues on these problems such as sensing [3], control [4], navigation [5, 6], decision making [7], and localization [8]. Case studies of successful autonomous mobile robots are well documented in the book [9], and the detailed failure analysis on the unsuccessful cases is conducted in [10]. However, there is lack of documentation on the design and implementation aspect, which is the target area of this paper.

This paper presents the design and implementation of an autonomous off-road robot based on integrative technologies and uncomplicated algorithms. The developed robot is able to autonomously traverse a given route and avoid obstacles it encounters along its path. Based on the functional analysis, simple configuration and easy to be adapted features, the system architecture has been developed as shown in Figure 1.

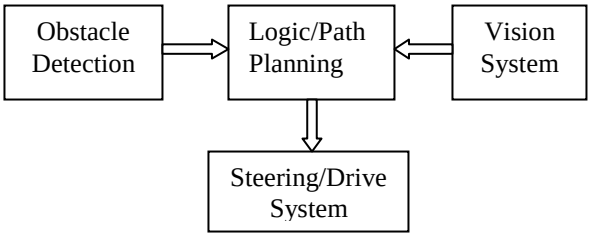

Figure 1: System architecture

The detailed design and implementation of each system are elaborated in the following sections.

2. Vehicle Chassis and Driveline

Chassis

The mechanical design of autonomous mobile robots always starts from the chassis design. The chassis structure determines the whole physical setup of the robot. It lays the foundation for developing a functional autonomous mobile robot. For the robot presented in this paper, four major criteria are considered for the chassis design: lightweight, ease of assembly, compact, and adequate space for housing sensors, batteries, mechanical and electrical components. A CAD model of the final design is shown in Figure 2. The overall length, width and height of the chassis are 40", 29" and 60". It is made of aluminum tubing which is optimal in terms of weight, ease of fabrication, and strength.

Suspension System

To reduce vibrations and absorb shocks caused by surrounding environments, a suspension system is integrated to the developed robot. Through the standard impact analysis, a suspension system with the stiffness of 29 lb/in, and damping coefficient of 16.5 slugs/s is designed. Using A-brackets, gyroscopic eye-bolt connectors, and other various connection components,

this system was attached to the chassis. Figure 3 shows a picture of the fully assembled suspension system.

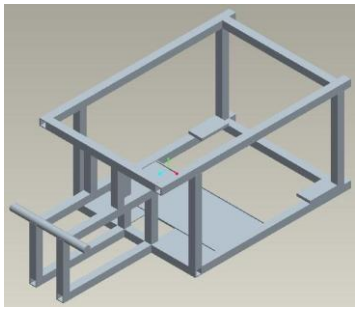

Figure 2. The Chassis' CAD Model

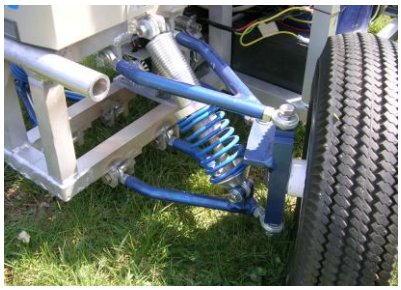

Figure 3. Fully Assembled Suspension System

Steering and Drive System

The steering system makes use of Ackermann Steering Geometry and adopts the rack and pinion structure. Two optical encoders were used in conjunction with this steering mechanism. The first was configured to provide rotation information with an encoder wheel affixed to the axle of the steering motor. The second was attached to provide centering information by connecting a centering L-bracket to the rack of the steering system. Half the L-bracket was removed, enabling the computer to determine whether the car was turned left or right and turn the steering accordingly to center the vehicle. Figure 4 displays the completed steering system.

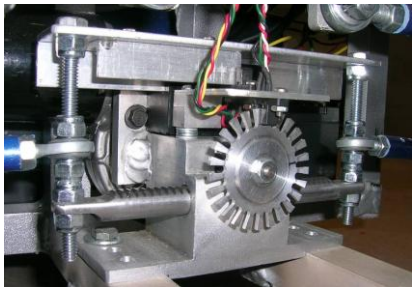

Figure 4. Fully Assembled Steering System

The motor used to drive the entire vehicle is a UNIPAC Drive System Motor (model MK35HE). This motor is an entirely sealed system, with lifetime lubrication. A Curtis PMC model 1208C motor controller is used to control the drive motor. To attain accurate data corresponding to the actual vehicle speed, a feedback system is designed for the driving mechanism. This feedback system is implemented via an encoder wheel, collar, and opto-switch. The encoder wheel was specially manufactured and bolted to a collar that was subsequently placed directly onto the rear drive axle, rotating in unison with the axle. An optical encoder was attached over the teeth of the encoder wheel to permit measurement of the rotational velocity of the rear axle.

3. Vision System

This system is primarily responsible for acquiring image data, as well as providing the path-planning logic with the location of the boundary lines that designate the edges of the obstacles. The imaging signal pipeline consists of the following steps:

1. Acquisition of image data from hardware
2. Processing and filtering of image data
3. Projection of data into 2D coordinate space

The camera used for image acquisition is a Basler 302fc video camera. The camera produces RGB images at a standard resolution of 640 x 480 pixels, which provides an adequate level of a detail while eliminating the need to down sample the image due to speed concerns. These images are acquired via a C++ software development kit (SDK), which provides a high-level interface for quickly accessing the IEEE 1394 bus on which the raw data is transmitted to the computer.

Using C++, software algorithms were developed for detecting lines and potholes in the images acquired from the camera. These algorithms operate based on training data that is supplied in the form of sample images of the desired detection surface, which are provided at runtime. This allowed creation for multiple training sets derived from an image database of lines painted on grass, with training sets corresponding to different lighting and grass conditions. The first pass of the algorithm uses the training data in the form of the equation (1).

$$PixelScore = \frac{\left|\mu_{sample} - CurrentPixel\right|}{\sigma_{sample}} \quad (1)$$

Once scores are calculated, pixels with scores lower than a specified threshold are marked as white in a new 8-bit monochrome image. Pixels with scores higher than the

threshold are colored black. These results in the effect demonstrated in Figure 5.

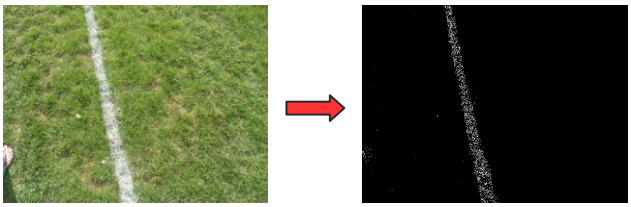

Figure 5. First Pass of Image Processing Algorithm

After the first pass of the image processing algorithm, the image must be filtered to reduce noise. This is accomplished by utilizing a square-radius median filter to eliminate isolated white pixels and connected white pixels that are grouped closely together. The effects of the filtering are demonstrated below in Figure 6.

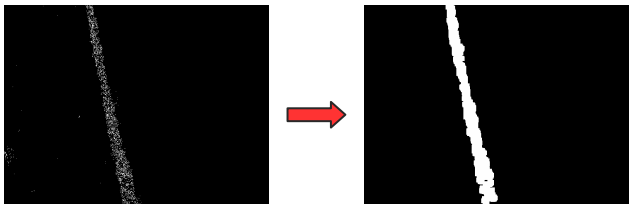

Figure 6. Second Pass of Image Processing Algorithm

To properly map filtered pixels to 2D coordinates, it was necessary to calibrate the camera for the specific orientation and lens being used. This calibration was accomplished by securing the camera into its mount on the vehicle, and then analyzing a single image taken of tile grid. This image can be seen below in Figure 7.

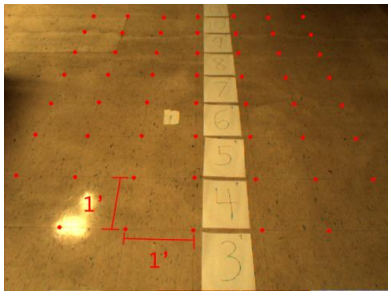

Figure 7. Camera Calibration Image

As seen in Figure 7, the pixel locations were recorded at 1' intervals along the image. From this data, an equation for mapping pixel Y values to real-world Y values was devised by fitting a fourth-order polynomial to the calibration data. An equation for calculating real-world X values from pixel X and Y coordinates was also devised by observing a linear relationship between pixel Y values and the amount of horizontal pixels required to represent 12 inches. Figure 8 demonstrates the results of the mapping system when used to detect a red line, with rightmost image representing a 2D overhead map of the area directly in front of the vehicle. The black area represents the trapezoidal field of view of the camera, while the white pixels represent the results of the image processing

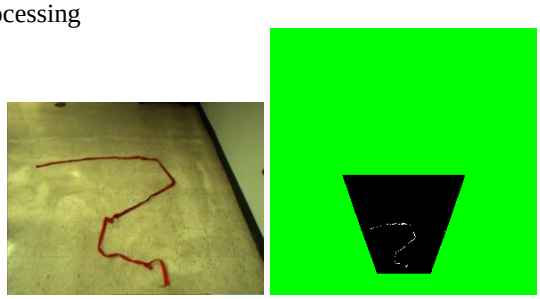

Figure 8. Mapping of Pixels to Real-World Coordinates

The image-processing algorithms and software interface for the vision system was implemented entirely in C++. C++ was ultimately chosen for efficiency reasons. C++ allows for generation of machine-native code that executes quickly, and also allowed for convenient interfacing with the path-planning software and camera SDK. The C++ implementation of the vision system was timed for multiple successive runs, and the average processing time was determined to be 35 milliseconds.

4. Logic and Navigation Systems

This navigation system was designed with discrete subsystems, which could be activated or deactivated as needed. The sensor used for the system to detect obstacles is a scanning laser range finder, SICK LMS 291-S05. This sensor was mounted to the front of the vehicle using a pair of adjustable brackets. The slotted holes allow the unit to be adjusted from level to 15° above horizontal. This adjustability allows the unit to avoid sensing hills or terrain up to a maximum grade of 15° that may appear to be an obstacle otherwise. The SICK LMS communicates with the laptop via a simple serial connection, RS-232. Data is read by the laptop using the C++ based software. The unit reads 181 values that correspond with 0°-180° in 1° increments. Erroneous distance values are eliminated using a simple filter. The filter removes any distance measurements whose values are significantly different than the values of both adjacent data points. A full sweep is performed and processed approximately three to five times per second. Mapping of objects is performed using the distance measured to each point. Objects are represented as a thin impassible boundary for the vehicle. An

example of how the unit maps solid objects from its surroundings is illustrated in Figure 9.

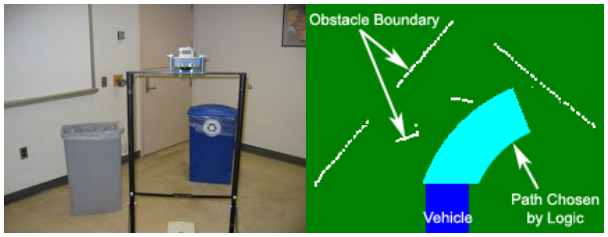

Figure 9. Actual Obstacles (left), Obstacle Map (right)

The fundamental structure of the path-planning algorithm created is based on an arc generation and selection process. To choose a best path, a series of five arcs representing possible paths for the car to take are generated. The number of arcs is chosen based on the resolution needed for the algorithm.

The arcs extend from the current location of the car at various angles out in front of the car. Figure 10 shows a diagram of this arc generation process.

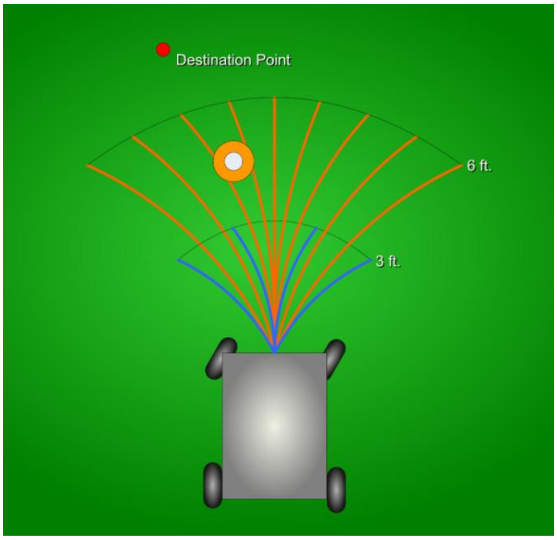

Figure 10. Arc Generation Process

After generating the arcs in front of the car, the laser range finder is polled to determine relevant obstacle data. Note that it is assumed that the environment being explored contains obstacles, and the destination point is always directly forward of the car at the top center of the mini-map of Figure 10. Also, the Image Processing subsystem will be checked to determine the location of any potential obstacles. Finally, based on the destination point and obstacle information, and the current angle of the car wheels, a weighted score is generated for each of the arcs. The equation for this weighting is shown below in Equation (2); where A, B, and C are heuristically determined weighting coefficients for the destination, wheel angle, and obstacle sub-scores, respectively.

$$Score_i = \sum \begin{Bmatrix} A \times \left(Dest_{x,y} - Current_{x,y}\right)_i + \\ B \times \left(Arc\theta - Current\theta\right)_i + \\ C \times Obstacle_i \end{Bmatrix} \qquad (2)$$

This process is then repeated several times a second. Because of this rapid pace of path selection, the movement of the car remains fluid despite the finite and discrete set of arcs being selected from, and this fluidity is maintained both in simulation and with experimental testing. In its most simplified form, the path planning algorithm has only two primary methods: UpdateMap() and CalculateScore() which are called for each execution of the main program loop. UpdateMap() gathers data from the sensor systems and computer vision system to place obstacle and line data on the possible paths map. CalculuateScore() then examines this map and assigns scores to each of the generated arcs based on obstacle, computer vision, and wheel angle.

5. Off-line Simulator

In order to verify the navigation algorithm before a working hardware and software platform were completed, a software simulator was developed. The simulator, which was coded entirely in C++, utilized the CPrimaryLogic class to make path-planning decisions based on vision and sensor data provided by the simulator. Output is provided by a graphical rendering system, which utilizes the Microsoft Direct3D API for generating polygonal models of objects in the simulated environment. An image captured from the graphical output can be seen below in Figure 11.

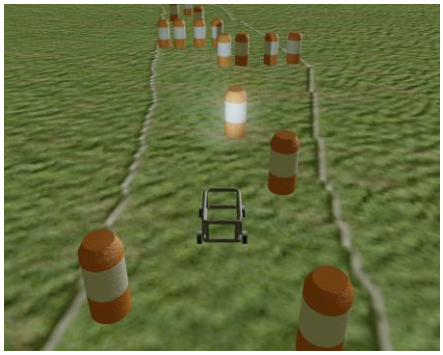

Figure 11. Graphical Output from Simulator

As seen in the figure above, the graphical output allows for observation of the vehicles behavior as the simulation progresses. A map representing the internal 2D map used by the CPrimaryLogic class is also present in the bottom-right corner, allowing for additional observation of low-level path-planning behavior. A software class diagram of the program can be found below in Figure 12.

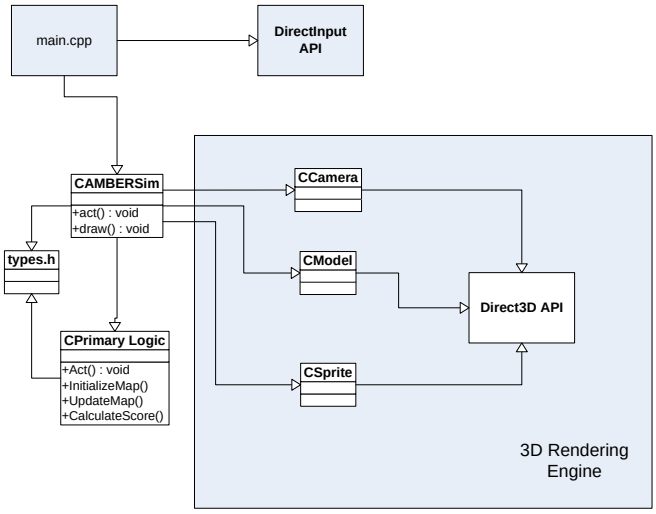

Figure 12. Simulator Class Diagram

6. Field Experiments

A comprehensive field test plan was developed for ensuring proper functionality of the various vehicle systems. These scenarios included tests of the vision system, obstacle detection system, navigation system, and terrain capabilities. Throughout testing, several modifications were made to improve performance. Examples of such changes included implementing edge detection in the vision system and adding multiple length arcs in the path-planning.

To further examine its robustness, our off-road robot attended the 15th annual Intelligent Ground Vehicle Competition (IGVC) held in Michigan on June 8th-11th, 2007 [11]. Our robot competed in the Autonomous Challenge. This Challenge requires the robot to maneuver through an obstacle course containing physical obstructions which must be avoided while the vehicle remains within boundary lines as shown in Figure 13. Obstacles on the course consists of various colors (white, orange, brown, green, black, etc.), five-gallon palls, construction drums, cones, pedestals and barricades that are used on roadways and highways. Our robot has successfully avoided various obstacles along its path in the IGVC as shown by the snapshots in Figure14. Our robot cannot complete the whole course since it deviated from the given route when it encountered the sand area of the course. This failure was because the vision system of our robot doesn't contain the case of sand. It can be easily solved by including the sand information in the image database of the vision system. Over twenty teams competed in the 15th IGVC. None of them finished the whole course due to its complicatedness. Despite of the failure due to sand, our robot has demonstrated excellent navigation of various obstacle configurations, and navigates even difficult situations such as switchbacks.

Fig. 13. Competition Course for Autonomous Challenge

Fig. 14. Snapshots of the Robot Navigating in the Competition Course

7. Conclusions

As the field of robotics expands and progresses, the need for rapid prototyping robotics platforms that can be easily configured has emerged. Reliable integrative technologies can be used to this end. The robot presented in this paper supports highly sophisticated sensing and simulation capabilities that can be expanded upon. This results in a complete high performance system, and represents a major step toward making possible the development of new robotics knowledge. By Successfully competing in the IGVC, the presented robot has demonstrated its robust integration of several subsystems including propulsion and steering, power distribution, line recognition, obstacle detection, navigation, and logic based decision-making.

Acknowledgements

The authors kindly acknowledge the hard work and contributions of their student research team for the year 2007.

References

[1] S. S. Ge, F. L. Lewis, *Autonomous Mobile Robots: Sensing, Control, Decision Making and Application*, CRC press, 2006.

[2] http://www.darpa.mil/grandchallenge/

[3] C. Robl, G. Faerber, "System architecture for synchronizing, signal level fusing, simulating and implementing sensors," *IEEE International Conference on Robotics and Automation*, pp. 1639-1644, April 2000.

[4] P. Coelho, U. Nunes, "Path-following control of mobile robots in presence of uncertainties," *IEEE Transactions on Robotics*, Vol. 21, No. 2, pp. 252-261, 2005.

[5] S. Lacroix, A. Mallet, D. Bonnafous, G. Bauzil, S. Fleury, M. Herrb and R. Chatila, "Autonomous rover navigation on unknown terrains: functions and integration, " *International Journal of Robotics Research*, Vol. 21, No. 10-11, pp. 917-942, 2002.

[6] Kelly, O. Amidi, M. Happold, H. Herman, T. Pilarski, P. Rander, A. Stentz, N. Vallidis, R. Warner, "Toward reliable off-road autonomous vehicle operating in challenging environments," *International Symposium on Experimental Robotics*, Singapore, June 2004.

[7] S. Balarkisky, A. Lacaze, "World modeling and behavior Generation for autonomous ground vehicle,"*IEEE International Conference on Robotics and Automation*, CA, April 2000.

[8] A Georgiev, P. K. Allen, "Localization methods for a mobile robot in Urban Enviroments," *IEEE Transactions on Robotics*, Vol. 20, No. 5, pp. 851-864, 2005.

[9] D. Kortenkamp, R. P. Bonasso, R. Murphy, *Artificial Intelligence and Mobile Robots: Case Studies of Successful Robot Systems*, AAAI Press/MIT Press, 1998.

[10] J. Carlson, R. R. Murphy, "How UGVs physically fail in the field," *IEEE Transactions on Robotics*, Vol. 21, No. 3, pp. 423-437, 2005.

[11] http://www.igvc.org/