Control of a UGV Equipped with GPS and Sonars in a Virtual Environment

Kelvyn Guzman and Sabri Tosunoglu
Department of Mechanical Engineering
Florida International University
10555 West Flagler Street
Miami, Florida 33174
(305) 348-2000
kguzm002@fiu.edu, tosun@fiu.edu

ABSTRACT

Lately Unmanned Ground Vehicles (UGV) are the focus of many research projects for both civilian and military applications. UGVs are robotic platforms with the ability of operating outdoors and over a wide variety of terrain with no human interaction. Performing hazardous tasks without endangering human lives is now possible with this technology. Since such systems are expensive, they are out of reach of many developers as they require a substantial amount of money to be invested.

In this paper, we describe a functional UGV prototype in a virtual environment which is proposed specifically for developers. A simpler method of dealing with sensors and the environment is presented to give the opportunity to test a system in an easily adaptive environment. Configuring and assembly of sensors on a real UGV requires skills in many areas including electronic, mechanical and computer which makes it difficult for an engineer specializing in one of these areas. In this work, a new approach for the solution of this problem has been developed utilizing the Virtual Environment Software. A pathfinding algorithm is used to work as a GPS system and provide the vehicle with information for the path to follow - as the real GPS system does. The path is recalculated every time the vehicle detects an obstacle on its way. Sonar sensors are also developed virtually to detect obstacles. Performance of a UGV is demonstrated in a realistic urban environment.

Keywords
Unmanned ground vehicles, virtual environment, sonar, sensor, autonomous vehicles

1. INTRODUCTION

Unmanned ground systems have been part of a significant technology push over the past several decades. As early as World War II, unmanned ground platforms have been experimentally evaluated for military missions such as minefield breaching. The first major mobile robot development effort was Shakey, developed in the late 1960s to serve as a testbed for the Defense Advanced Research Projects Agency (DARPA) at Stanford Research Institute. Shakey was a wheeled platform equipped with steerable TV camera, ultrasonic range finder and touch sensors, connected via an RF link to its SDS-940 mainframe computer that performed navigation and exploration tasks [6].

A sophisticated stereo system was implemented in the Stanford Cart Project at Stanford University AI Lab from 1973 to 1981. This technology was employed to navigate and avoid obstacles by using a single TV camera which was moved to each of 9 different positions on top of its mobility base, and the resulting images were processed by the offboard KL-10 mainframe. Feature extraction and correlation between images allowed reconstruction of a model of the 3-D scene, which was used to negotiate cluttered obstacle courses (planned an obstacle-free path to the destination) taking about 5 hours [6].

In late 1980s, Shakey reemerged as the DARPA Autonomous Land Vehicle (ALV) using the DARPA’s Strategic Computing (SC) program, whose goal was to provide a realistic task environment for technology research. The ALV was built on a standard manufacturing eightwheel hydrostatically-driven all-terrain capable of speeds of up to 45 mph on the highway and up to 18 mph on rough terrain.
Since the Gulf War, an urgent need has surfaced to transform the army from the one characterized by heavy armor and firepower into a lighter, more responsive force that is at once more lethal and survivable. For that purpose, several programs including the Defense Advanced Research Projects Agency (DARPA) started receiving funds from the Army to research basic and advanced intelligent systems and lead developments in crew automation technology [1].

Computer technology played a key role in the development of unmanned technology, and DoD agencies were aware of that. DARPA’s role in fueling the information revolution has been pervasive and enduring. DARPA has been credited with between a third and a half of all the major innovations in computer science and technology.

Unmanned ground vehicles (UGVs) have many valuable attributes in military applications that will aid, complement, extend soldiers’ capabilities and reduce risk on the battle field.

Later, there was a need for using this technology for civilian applications, to find ways of reducing traffic accidents and make vehicular traffic more predictable. In 2004, DARPA Grand Challenge was established, where fifteen autonomous ground vehicles attempted to complete a 142 mile course in the desert. It was the first long distance competition in the world for driverless cars.

Later on, the competition was taken to the city, and the name was switched to DARPA Urban Challenge. This requires autonomous ground vehicles maneuvering in a mock city environment and obeying traffic rules by merging into moving traffic, navigating traffic circles, negotiating busy intersections, avoiding obstacles, hitting pedestrians and collisions.

Unmanned ground vehicles consist of many systems such as navigation, obstacle detection, vehicle control and communication [2].

Obstacle detection can be performed using a wide variety of sensors. LIDAR is a vision system capable of detecting static and moving obstacles, identifying the drivable road surface, and location road markings.

Unmanned ground vehicles are linked to many complex systems such as navigation, obstacle detection, communication and vehicle control.

Vehicle Control involves three categories which are Maneuver, Accelerate, and Brake. Maneuvering can be implemented using Steer-by-Wire System. For acceleration an electronic board can be used to enable the use of a computer to set the gas command and a system of cable and pulleys can be implemented to activate the brake pedal [3].

Computer modeling and simulation have the capability to play an important role in designing and analyzing large-scale complex systems in a safe and cost effective way. A large-scale complex system can be defined as a system of systems consisting of a number of components (such as humans, machines, technical systems, etc.) interacting with each other in complex ways to accomplish their goals.

These tools have played a key role in the evolution of the UGV technology. This has enabled realistic training and led the evolution of controllers and algorithms for advanced civilian and military applications. Numerous organizations and agencies have created and designed simulators for training purposes without the need of having an actual equipment, which allow them to reduce accident risks and save in training costs. The resultant feedback that this virtual tool provides to system designers and planners has accelerated the rate of improvement. Unmanned systems have a very promising future as proved by the commitment and continuous contributions of two multi-national US companies; namely, Microsoft and Google. Microsoft has created and developed a software called Microsoft Robotic Developer Studio which allows the creation of robotics applications across a wide variety of hardware. And Google has introduced a software named Google SketchUp which can be used to create 3D models for a variety of applications including robotics simulations [5].
2. MODELING OF THE VIRTUAL ENVIRONMENT

An urban city is a complex environment where a vehicle must interact with pedestrians, other vehicles, traffic rules and different kinds of situations.

In order for the urban virtual environment be created, a set of characteristics had to be defined. The need of finding a 3D modeling software, a graphic library, a 3D engine and a programming language was extremely important for the adequate development of the software. The 3D modeling software needed to provide the necessary tools to create all the models (buildings, roads, vehicles, etc.) required for the environment, and export in the appropriate file extensions so it can be used appropriately.

3D World Studio is a 3D software which was designed to provide a fast object manipulation and drawing environment. This is a tool intended for mass production of the media industry, by using fewer number of mouse clicks, high-quality results can be generated. Also the ability of performing stretches, positions, mirror, flip and rotate meshes manipulations makes it an incredibly useful tool. An important feature it has is the ability of exporting to .x, .map, .wmp, .dbo, .b3d and .dif files. The file extension .x which stands for DirectX is the one that is used in this work for the simulation purposes.

3D Engines are software systems designed for the creation and development 3D environments where a set of intelligence is needed [5].

The engines are developed with different kind of features:
- Graphics engine
- Complex models
- Artificial intelligence (AI)
- World physics

The characteristics for the ideal 3D Engine for this project are listed as follows:
- Fast compilation times.
- Low cost.
- Able to support libraries such as DirectX.
- Provide adequate built in commands for simulation purposes.

Dark GDK is a 3D Engine and was developed in C++ and based on some graphic libraries like DirectX. The compilation times are fast in this engine, it is free to download for personal use and a fast rate speed is achieved because of the Microsoft Visual C++ compiler.

After the tools that will be used for the simulation are defined, a set of concepts should be understood. For graphics to work either in 2D or 3D environments, it is required that a way of describing the position of any object in space is clearly defined. This requires a baseline, which is usually described through a coordinate system that is more accustomed to the Cartesian coordinates in which a point can be expressed as a distance across two (x,y) or three axes (x,y,z) mutually perpendicular intersecting at point (0,0) for two dimensions, and (0,0,0) for three.

The current PC monitors are two-dimensional devices and in order to convert it to three-dimensional, a set of tricky operations must be performed; the first step is the orthographic projection. In the orthographic projection rays are drawn parallel to the plane of projection and the image is formed by those rays that intersect the object, as described in Figure 8.

Another method used is the perspective projection in which objects undergo deformation by distance, as objects move away from the plane of projection, the size is reduced similar to when an object is moved away from the human eyes. In the Figure 9, one can notice how two images with different sizes, seem to have similar dimensions in the projection plane due to different distances. For that reason,
this perspective is used in simulations, so it can provide the sense of realism.

In the case of orthographic projection, the view volume is shaped like a rectangular box, so it can easily be determined by 6 steps: near, far, top, bottom, left, right and in the case of projection, truncated pyramid-shaped rectangular base, which can be determined by the field of view, the aspect ratio, near and far.

Figure 6. Perspective Projection

As described in Figure 10, both volumes are calculated from the observer’s position. Also the dimensions of the plane closest to the observer may be higher or lower than those of the window.

3. VIRTUAL GPS

The Global Positioning System (GPS) is a satellite-based navigation system made up of a network of 24 satellites placed into orbit by the U.S. Department of Defense. The implementation of this technology in the virtual world was challenging, because we do not have real information that the modeled GPS can receive; every information, object and interaction must be simulated.

The approach taken to model and create the Virtual GPS was to utilize the A* Pathfinding algorithm. The A* algorithm is one of the best-first graph search algorithm that finds the least-cost path from a given initial node to one goal node. Using a distance-plus-cost heuristic function to determine the order in which the search visits nodes in the tree.

4. RANGING SONAR

Ranging Sonar is an obstacle detection sensor that works by continuously transmitting an ultrasonic burst (sending a pulse to hit the obstacle and receiving it) and providing an output pulse that corresponds to the time it took for the echo to return to the sensor. With this information the distance can be easily calculated.

Sonar sensing is ubiquitous on mobile robots due to its low cost, the simplicity of the required processing, and the rapidity with which it can return results reflecting range measurements over a large region of space [8].

The virtual representation of this sensor into the simulation project was approached using a collision detection method commonly use in 3D environments.

Spherical collision detection describes perfectly how this method works. In this implementation, the volume of every object is represented by a sphere with a radius equal to the maximum distance between any of the vertices of the 3-D model and the model’s center. Collisions occur whenever the distance between the center of two objects is less than or equal to the sum of the radius of two objects. The collision normal is the normalized vector going from the center of one object to the other.

A more advance collision detection method can be applied which is more complicated but at the same time more accurate. This method instead of using a sphere as a
volume, it uses a set of Oriented Bounding Boxes (OBBs) [9]. Using these collision methods the sonar was virtually developed. A small 3D object was created and used it to simulate the ultrasonic burst that the ranging sonar produces. Since the visibility of the object is not necessary, it was hidden. The virtual ranging sonar is continuously sending the small object and every time it collides, it will return to the vehicle, providing the distance information.

5. SENSOR CALIBRATION

Another positive characteristic that the simulation software has is the ability of allowing the user to make relatively significant changes in seconds. If the user does not consider that the sensors’ configuration is adequate for the application, then it can be reconfigured by simply clicking the mouse a few times. The calibration of each sensor that has been implemented in the system is integrated into the simulation software.

The configuration on a real UGV requires skills in many areas including electronic, mechanical and computer which makes it difficult for an engineer specialized in one of these areas. Also the time that the sensor calibration might take is then translated to costs. The time companies, institutions, or organizations take to develop their technologies will determine the success or failure in the industry. Most of these organizations are funded by a military agency, and the economical resources obtained are to provide results in a specific period of time which might lead them to a big issue if the concept researched and implemented in the development of the technology was not the appropriate one, and some reconfigurations must be applied.

6. MANIPULATION OF THE ENVIRONMENT

As previously mentioned the 3D engine that will be used is Dark GDK, and the programming language is Microsoft Visual C++. The built in commands that Dark GDK provides will reduce significantly the programming and computational time.

The objects are placed strategically using (x,y,z) as coordinates position to build the urban environment, where “y” is going to have a fixed value since the simulation will be created in a flat surface. Objects are loaded to the environment using the following command:

```
dbLoadObject ( char* szFilename, int iObject )
```

Where `szFilename` is the actual name of the file that will be imported to the simulation, the file extension should also be included with the name i.e., building.x. `iObject` is the ID number that will be assigned to the object being loaded.

After the objects are located properly; with use of some 3D commands the object can be scaled, rotated, hide, etc. The following command is used for scaling:

```
dbScaleObject ( int iObject, float fX, float fY, float fZ )
```

The creation of the environment is the first step; the second one is to know how the motion will be given to the vehicles.

To provide a realistic motion of the vehicle (turning and rotating the wheels) some tricky steps must be performed.

The vehicles are created having different limbs. Each limb moves independently from the others i.e., the arm or leg of a human being. The illustration bellow provides more detail of the vehicle limbs.

```
x3ds_BODY
x3ds_SUSP_fr
x3ds_WHEEL_fr
x3ds_susp_br
x3ds_wheel_br
x3ds_suspend_br
x3ds_wheel_bl
x3ds_suspend_fl
x3ds_WHEEL_fl
x3ds_UNDERNEATH
```

Figure 10. Vehicle Limbs

The picture shows that the vehicle has 10 limbs, the way this limbs are called inside the programming language is by numbers. In this particular case the `x3ds_BODY` limb should be called with the ID number of one (1) and in the same order `x3ds_SUSP_fr` with the ID number of two (2). To rotate a limb, in this case a wheel the following command should be used:

```
dbRotateLimb ( int iObject, int iLimb, float fX, float fY, float fZ )
```

Where `iObject` is the ID given for the vehicle when it was loaded, `iLimb` is the ID number of the limb that wants to be used and `(fX, fY, fZ)` are the angles of the limb.

7. VEHICULAR TRAFFIC

Urban traffic is an important part of urban environments. The traffic congestions generated and frequent traffic accidents are the focus of many research industries. The need of reducing or eliminating these issues has been explored for many years. This is one of the key factors that cause urban environmental pollution.

The creation of virtual traffic environment becomes the perfect resource to do all the tests with autonomous vehicles without putting civilian lives in risk while the testing stage is performed.

Virtual traffic environment includes virtual roads, virtual traffic sign, Autonomous Virtual Vehicle (AVV), virtual
buildings and so on. AVV is one of the most important dynamic and intelligent components of driving simulator system, which is also the crucial factor for obtaining reliable research data from driving simulator. The AVV possesses autonomous and intellectual characters, and can simulate various behaviors of the vehicle in traffic environment i.e., avoiding obstacles, lane changing, overtaking, etc.

The simulation method for AVV autonomous overtaking behavior can be divided into several phases: firstly, vehicle changes to adjacent driveway; secondly, after arriving at the adjacent driveway, vehicle speeds up for a period of time, and then overtakes the front car; finally, the vehicle goes back to the original driveway. From this perspective, it is illustrated that the relation between lane changing and overtaking is close [10].

The Figure 11 illustrates this concept.

8. DEFINITION OF RULES FOR THE AUTONOMOUS VEHICLE

The most important aspect in an Unmanned Ground Vehicle is the decision-making. It can have the best controllers but if it does not have a good decision-making algorithm, the vehicle will not react or perform in an adequate manner.

In reality a good driver is measured with its ability of making the right decision at the right time. That decision might be the difference between a terrible fright or a fatal accident.

To achieve the decision making control the Fuzzy Logic method has been implemented. Fuzzy rule-based models play an important role both in control and decision-making [12].

Fuzzy logic is a concept based on the logic functions AND, OR and NOT which are used to construct logical rules. This one differs from the Boolean logic where only binary inputs and outputs values are used; in fuzzy logic values such as VERY LOW, LOW, OKAY, HIGH and VERY HIGH can be assigned [11].

Let’s assume that only a ranging sonar sensor is being used in the autonomous vehicle. In this particular scenario the collision mitigation system will be active. The rule for this situation will be the following:

IF (vehicle position) ≤ (distance from object detected) THEN Stop

Where STOP will has a set of parameters to make the vehicle to brake.

In the previous case just one sensors was used and for that reason there was not any decision to make. If we used more than one controller a decision making algorithm must be implemented, which has already been defined previously.

Some of the rules implemented in the developed software are described as follows:

- IF (vehicle is at starting point) THEN calculate the path with the GPS
- IF (object detected) > (specific distance) THEN keep following GPS Path
- IF (object detected) < (specific distance) THEN recalculate path with GPS
- IF (object detected) < (specific distance) THEN turn left and avoid the object
- IF (vehicle is in an intersection) THEN turn either left or right depending on the path
- IF (vehicle is at end point) THEN stop

9. RUNNING THE SIMULATION

Once the vehicular traffic, the controllers to be used in the UGV, the start point, the end point, the decision-making rules and the environment are specified the next step is to run the simulation.

In a general case study before the vehicle starts running, the ideal path must be calculated. Using the virtual GPS mounted on the UGV, the autonomous vehicle determined the most adequate course to take.

10. CONCLUSIONS

Computer modeling and simulations play important roles in the evolution of complex systems in the robotic industry. This work describes how a simulation software package for unmanned systems was created and developed. The proposed software package is intended to evaluate and test new controllers as well as to train personnel in virtual scenarios without the need of investing in expensive hardware.

The developed software provides an adequate environment for the creation and testing of new control algorithms. The output provided by the software gives information to the user to determine the weak aspects of the vehicle’s controller configuration. The results obtained can be used to further research and improve the controller that is under development and testing.

The environment has the necessary features to analyze the performance of a designed controller.

The software offers a tool for developers to convert an ordinary vehicle to an autonomous one and see how it interacts with the vehicular traffic. Developers have the opportunity to merge into a field where usually a substantial amount of money must be invested in order for anyone to explore it.

Future improvements must be introduced in the software to cover more areas into this field. The inclusion of more sensors will help the platform mobility, providing it with more necessary information for the interaction with the surroundings. Also the addition of different decision making algorithms will provide flexibility and opportunities to explore different ways to target a problem.
11. REFERENCES

[1] Committee on army unmanned ground vehicle technology and national research council (2002). Technology development for army unmanned ground vehicles.


