ARC - AUTONOMOUS RC

Autonomous Obstacle Avoidance and Navigation

Project Introduction and Background

Introduction

Autonomous RC (or ARC) is a research project aimed at seeing if autonomous navigation, obstacle avoidance, and parallel parking can be achieved, at a 1/10th scale. A significant goal is to use commodity hardware that is obtainable through major retailers (online as well as brick and mortar), for a reasonable cost.

Our implementation follows the Robot Operating System (ROS) standards. ROS is a widely-used platform for robotics applications that provides consistent protocols and guidelines for data communication, which makes integration of multiple modules trivial.

Our implementation enables our the RC car to go from point A to point B autonomously while avoiding obstacles. The RC car is also able to parallel park if the surrounding environment is suitable.

ARC is designed to be open-source. With slight modifications, it should be able to run on different types of car-like vehicles. Students and hobbyists are welcome to contribute to the project by improving upon the software.

RC Car V2

Project Background and Motivation

Millions, if not billions of dollars are invested in autonomous vehicles each year. Most of these systems are cost prohibitive for the average user, looking to replicate them at a small-scale. Our motivation for the project includes the following:

- Seeing if commodity hardware can be used to help reduce the cost of autonomous systems.
- Provide an affordable test platform for scaled high-speed obstacle avoidance with GPS navigation.
- See if there are solutions that can be scaled up from the RC car, for consumer DIY projects or commercial products.

Project Description and Outcomes

Project Description

Using ROS and RVIS, we were able to simulate our software package before testing on the car. Simulation allowed us to move forward with our software implementation while we also worked on the hardware side of the project. Using the software simulation also allowed us to ensure that our code was working properly, and the car would have the expected behavior.

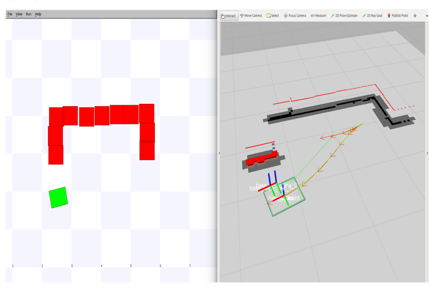
Software features:

- GPS waypoint following
- Obstacle avoidance
- Parallel parking
- Remote control

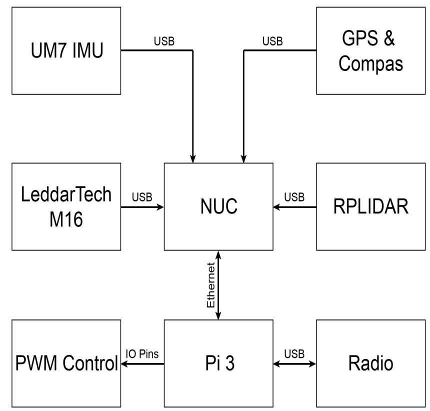
Hardware components:

- Traxxas© Summit RC car
- Intel® NUC Skull Canyon
- Rasberry Pi 3 LeddarTech M16 and RPLiDAR
- Sparkfun Venus GPS with TFDI Serial Breakout
- UM7 Inertial Measurement Unit (IMU)
- Voltage Boosters and Regulators
- 3DR SIK 915mHz Radios

We originally wanted to modify GT AutoRally project and integrate it with ROS to fit our software needs; however, as research progressed, we determined that the complexity of GT AutoRally made modifying to our project too complicated and time-consuming. We changed from integrating GT AutoRally, to strictly using ROS for our navigation stack.



Car Parallel Parking using Path Planning in RVIS



System Hardware Layout

Project Outcomes and Results

With our research and hardware choices, we determine that it is possible to build an autonomous RC with a budget of approximately $2000.

Throughout the entire development process, we focused on using open-source software packages for almost all autonomous features, with the goal of having a combination of ROS packages that would be simple to understand and modify.

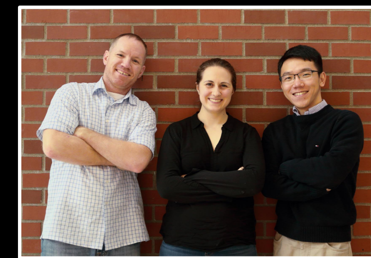
The most challenging part of our project was finding packages that would allow us to have flexible hardware options so that users would not be required to use our hardware configuration.

Another challenge is having the car navigate in sunlight, due to the UV saturation that causes our RPLiDAR unit to give incorrect data. Using the LeddarTech M16 allows us to have forward facing vision in direct sunlight, up to 150'; however, it does not provide a 360º view of surrounding the car needed for functionality such as parallel parking.

Environments that include a lot of glass are very challenging to navigate in (such as Kelly Engineering Center) as the car is unable to receive stable signals from the Leddar and LiDAR sensors.

We also have no implementation for low object detection for items such as curbs and drop-off detection for cliffs. The car can drive over a 4" curb; however, it occasionally will crash if the curb is between 6" and 8".

Computer Science Capstone
Group 44



Group Members

Daniel Stoyer
stoyerd@oregonstate.edu

Cierra Shawe
shawec@oregonstate.edu

Tao Chen
chentao@oregonstate.edu

Client and Affiliation

D. Kevin McGrath
Oregon State University

Conclusion

In conclusion, we implemented basic obstacle avoidance and path planning within our simulation software. Using the the obstacle avoidance, we are also able to parallel park the car by specifying the cars end position. The simulation has also been implemented on the hardware; however, the functionality is still very limited and has room for further improvement in the future.

Our research and documentation, along with our hardware, lay the foundation for others to further improve upon our software platform, by adding functionality, such as autonomous navigation at speed.

Oregon State UNIVERSITY