Midterm Progress Report

ARC - Autonomous RC

Senior Capstone Project

Oregon State University

Winter 2017

Tao Chen, Cierra Shawe, Daniel Stoyer

Version 1.0

February 17, 2017

CONTENTS

1 Project purpose and goals

The purpose of the Autonomous RC (ARC) project is to determine if it is possible to build an autonomous RC vehicle using commodity components, meaning components that are relatively inexpensive and can be bought at places like Radio Shack®, Best Buy®, or on Amazon.
Our goal is to make an RC vehicle navigate autonomous to a given waypoint/location, preferably at a high rate of speed. Stretch goals are to make the vehicle drift around corners and parallel park.

Fig. 1. Drifting example. Image from https://autorally.github.io/

While our main goal is to have a functioning autonomous RC vehicle we also hope that we can produce instructions that RC enthusiasts can follow to produce a functioning, consumer-grade autonomous RC vehicle of their own.

2 Cierra

2.1 Current Status

2.2 Left to do:

- Something to do...

2.3 Challenges

Problems	**Solutions**
Problems that impeded progress.	Specific actions to resolve problems.

3 Tao

3.1 Current Status

3.2 Left to do:

- Something to do...

3.3 Challenges

Problems	Solutions
Problems that impeded progress.	Specific actions to resolve problems.

4 DAN

4.1 Current Status

Much of the work of the first half of the term has been setting up the software environment that will run our RC vehicle. We have our operating systems installed and configured on our primary computers: the laptop which is the ground control station (GCS), the Raspberry Pi 3 (RPi3) and PXFmini which is the autopilot, and the NUC which is the companion computer that handles the raw calculations. We have ROS installed and running on all platforms.

We have been working on getting the RPi3 talking with the PXFMini autopilot cape. The PXFmini autopilot is used to integrate sensor data to send to the NUC and to integrate data coming from the NUC to the vehicle controls. The autopilot also automates calibration of the vehicle and performs other convenient features. We have a system image for the RPi3, from Erle Robotics (the makers of the PXFmini) installed and are able to get the PXFmini launched and armed. This means that the autopilot is ready to take commands.

We have also been working on communicating computer-to-computer using ROS nodes. We have run successful tests over direct ethernet connections talking between ROS nodes. These tests lay the groundwork for the NUC to be able to send instructions to the autopilot.

While the following areas are all incomplete at this time. However, our accomplishments in each of them pave the way for us to reach the goal of an autonomously controlled RC vehicle.

All told, problems with the PXFmini have delayed us from actual hardware testing by about 3 weeks.

4.1.1 Image Analysis

No significant progress has been made on image analysis. We still plan to use the ROS rtabmap_ros package to handle depth-finding and environment mapping. The work Tao has been doing with ROS and Gazebo (a simulation environment) has given our virtual vehicle (running the actual software) the ability to ”see” its environment, but we do not have the analysis part of it integrated yet.

4.1.2 Radio Communication

The status of radio communication is much the same as image analysis. We do not have the physical platform ready for developing and testing radio communication. The radio component of this project should prove to be fairly trivial since radio control and telemetry communication are completely standardized and there are many examples and tutorials, both in hardware and software, to follow. We will be using the MAVLink protocol for radio communication.

4.1.3 User Interface

We have a command line interface (CLI) up and running on the GCS, the RPi3, and the NUC. We are using ROS for the CLI. Successfully installed ROS on both the GCS and the NUC. This gives us our command line interface (CLI) for issuing commands to the vehicle. We have been using the CLI to send and receive individual commands to ROS topics (a ROS topic is basically the way that ROS sends and receives commands). We are able to publish and subscribe to ROS nodes, specifically MAVros nodes, which is also part of being able to monitor and control the vehicle. We have also run python scripts that have published to ROS topics.

4.2 Left to do:

4.2.1 Image Analysis

- Integrate data from lidar and cameras into the obstacle avoidance module.
- Perform testing and determine if rtabmap_ros will work for our needs.

4.2.2 Radio Communication

- Connect telemetry radios to the GCS and the NUC.
- Create/use ROS topics for reading the telemetry data.

4.2.3 User Interface

- Install and setup a GUI on the GCS for command and control of the vehicle. The most likely candidate at this time is ArduPilot, an open source package that allows fine vehicle control and waypoint selection.
- Some of the requirements of the GUI will be:
  - hooking in to telemetry
  - manual override of radio control
  - manual override of autonomous operation
  - ability to set waypoints
- Fill out CLI functionality. We still need to create custom convenience commands for CLI. Issuing commands to a ROS topic is pretty complicated right now.

4.3 Challenges

Problems	Solutions
Problems that impeded progress.	Specific actions to resolve problems.
Did not get RPi3 erle image until late (week 4/5).	Getting the image allowed us to attempt to set up the PXFmini on the RPi3.
The first erle image sent was corrupted, it took about a week waiting for a new image.	Erle sent a new image.
The environment from Erle seems to be very fragile/unstable. We have run into different problems each time we have re-installed the image they gave us. Our client is have yet a different problem in a separate project where they are trying to use the PXFmini.	This has not been resolved.
The PXFmini instructions/tutorials were inconsistent/incorrect. This has created quite a few problems, which has further delayed getting our autopilot running.	We have been working with our client to solve some of the issues with installing and running the PXFmini autopilot.
The RPi3 WiFii and ethernet were not configured to connect to the internet.	Researched how to set the RPi3 to connect to the internet via ethernet. It turned out that this problem was caused by how Erle set up their OS image. Found the proper settings to connect to the internet via either WiFi or ethernet. However, if WiFi is set to connect to the internet, external computers cannot talk to the PXFmini via WiFi, same for ethernet.
The PXFmini does not respond to commands via MAVros.	MAVros is the ROS api to MAVLink, the protocol for commanding the PXFmini. While we are able to publish (send commands to) and subscribe (get data from) the appropriate topics, the PXFmini does not respond as would be expected. The motors do not activate. We have found a forum posting stating that a certain variable (SYSID_MYGCS) needs to be set to 1 in order to override the RC in control to allow computer control of the PXFmini. It looks like the variable can only be set through a GUI, such as ArduPilot. We will try installing ArduPilot on the GCS and see if we can set SYSID_MYGCS to 1 and see if that makes any difference.
My personal laptop died during week 1	I received a replacement in 2 days and it took another 2-3 days to set my environment back up.
Installing Ubuntu 14.04 in dual boot did not go well. It took about a week to get installed properly along side Windows 10.	I wanted a native install of Ubuntu 14.04 (the OS we wanted to use with ROS) on my system to streamline development. Finally got the OS working properly