Purdue University

Design for an Autonomous Vehicle Racing System

Anthony Goeckner
Benjamin Huemann
Zachary Perry
Harold Smith

CS 407 – Senior Design
Team 21 – Autobot Racing

January 26, 2018

Contents

1 Purpose

Autonomous driving is an area of active research, and the Autobot Racing team aims to contribute to the growing body of knowledge in that area. The goal of this project is to create an automated car racing system, with centralized control of all vehicles. While similar systems already exist, this new Autobots system will implement advanced behaviors such as lane-keeping and passing capabilities, along with collision and obstacle avoidance. This will be achieved with one sensor: an overhead camera tracking all vehicles.

1.1 Background Information

As autonomous vehicles become more prevalent in today's society, researchers need to understand the complexities and difficulties of developing algorithms for these systems. The large size and cost of actual vehicles makes full-scale research setups cost prohibitive. Therefore, a cheap, small-scale environment is needed to conserve money and allow for easier research and testing. While this system is targeted towards researchers wishing to pursue autonomous vehicle development, the Autobot Racing development team will also be performing its own research using the framework it establishes, including development of autonomous driving algorithms. The team will thus serve as both developers and as researchers. After the project is completed, the framework will be made available for future research by other interested parties.

While similar systems have been created, including a well-developed one at ETH Zurich, this system aims to be cheaper to build, use fewer custom components, and be easier to set up. Unlike the system built by ETH Zurich, which involved custom PCBs with microcontrollers and digital radios on each car, Autobots will use off-the-shelf RC cars, requiring no modification to the vehicles themselves. This significantly decreases hardware costs.

2 Design Overview

The Autobot Racing system will consist of four major logical components: the vehicles, the camera system, the research framework, and the control system, which extends the framework. The computer vision system captures the location of each vehicle and any obstacles on the track, as well as course boundaries. Using the framework (essentially the core of the system), this data is passed to the researcher-created control system, which determines adjustments that must be made to the vehicle's position or heading. These adjustments are then

Figure 1: System deployment diagram.

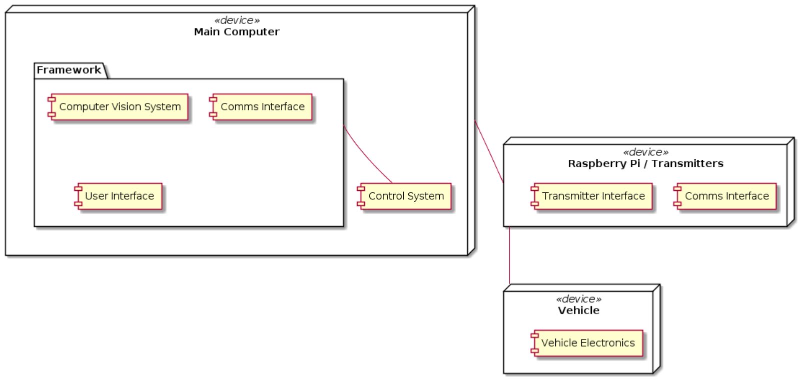
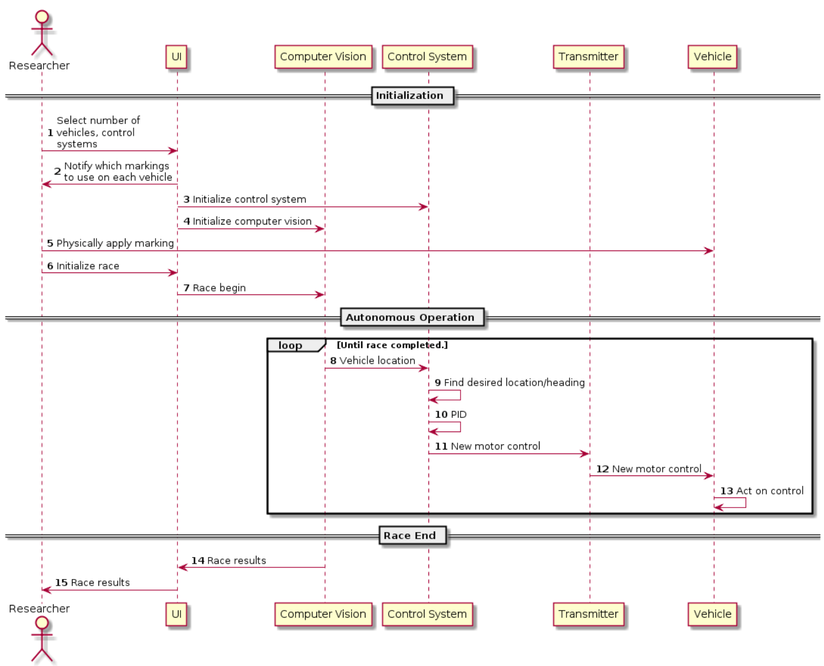


Figure 2: Overall race sequence.

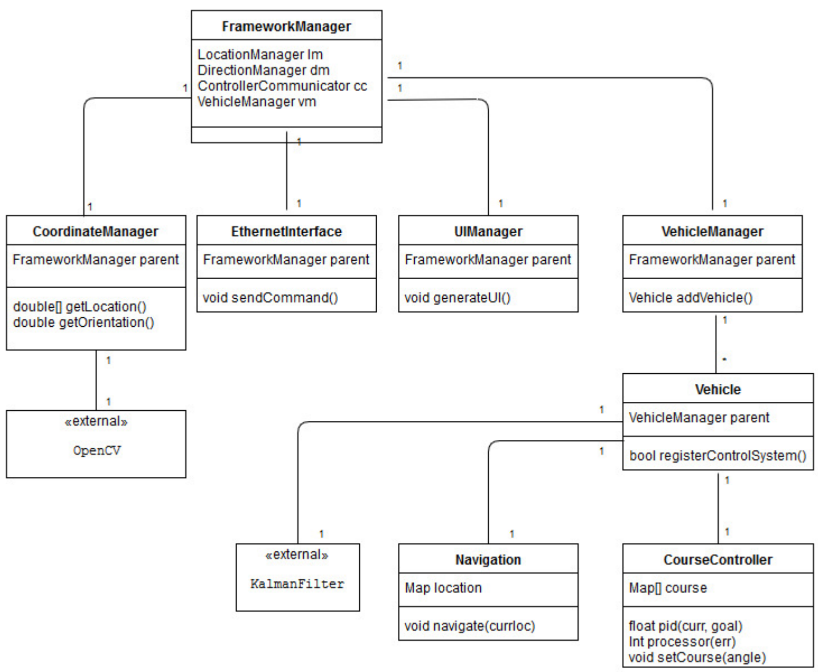


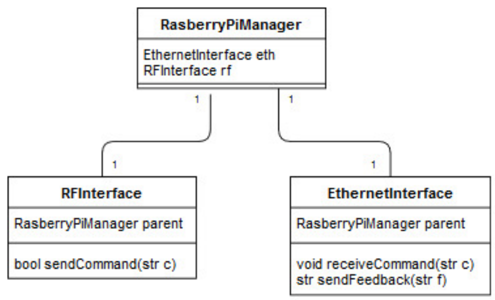
passed to the Raspberry Pi for transmission to the vehicle's original electronics. This structure is visible in Figure 1. Basic operation of the system by a researcher will follow the sequence diagram seen in Figure 2.

For computer vision purposes, a single overhead camera will be used. This choice might at first seem unrealistic, but it is the most cost-effective way of studying autonomous driving theory using real hardware. Placing a camera and full autonomy on each vehicle would increase costs dramatically, as each car would need at minimum a powerful processor and a camera for computer vision, potentially along with additional sensors and a custom integrated circuit to replace the vehicle's original electronics. By performing this image processing and autonomy remotely, the cars can be left in their stock, off-the-shelf configuration, drastically reducing complexity and cost. The cars will instead be controlled by means of a Raspberry Pi providing electrical inputs to the original RC transmitters, as further described in section 3.

2.1 Overall Design Detail

The below diagrams help to demonstrate the overall class structure of the system.





3 Hardware Design Detail

The hardware requirements for Autobot Racing are designed to be minimal and as cost-effective as possible. For a full list of materials required, see Appendix A.

3.1 Vehicles

The R/C cars which will be used are Velocsis 1/32 scale buggies with a 2WD electric motor, operating on a 2.4 GHz frequency. See Appendix A for more information.

3.2 Control Interface

Each vehicle will be controlled by its original RC transmitter. However, the transmitters will be modified to accept electrical inputs from a Raspberry Pi, rather than from the original buttons. This Raspberry Pi will be connected to the main computer using an ethernet interface.

3.3 Camera & Mount

If available, a 1080p high-definition web-camera from the CS Department will be used. If such a camera is not available, a basic high-definition web-camera will be purchased. The camera will be mounted on a boom above the track, and will be able to see the whole track including the cars on it. The cars will have dot patterns on the tops of the cars, and the camera can see and track them using these patterns. The camera will be held over the track, looking down, using a boom to see the course from the edge.

3.4 Race/Test Track

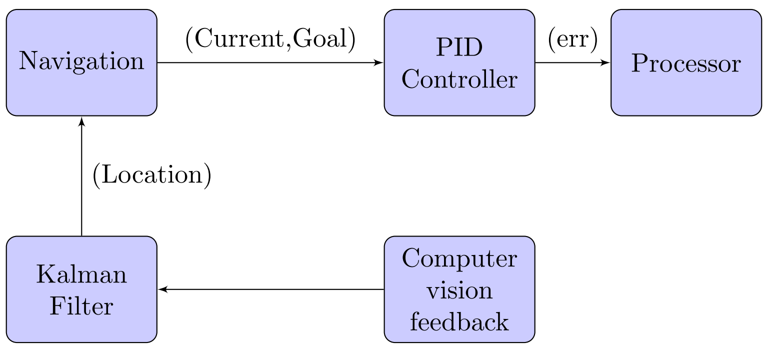
A modular RCP 96 track will be used, allowing for more than one layout. This track is also wide enough for cars to pass each other.

See Appendix A for more information.

4 Control System Design Detail

Autobot functionality and control will be driven by a feedback control system. This system is illustrated in figure 3.

Figure 3: Control system diagram.



The system will utilize a PID controller, processor, Navigation, Kalman filter, and computer vision acting as sensory input. The PID Controller, short for proportional - integral - derivative, is a mechanism for controlling systems with need for constant modulated control. It works by taking two inputs and calculating a percent error. This percent error indicates far the vehicle's heading is from its intended goal. The percent error will be fed into a processor, whose job is to adjust heading based on error. For example, zero would indicate no error; anything greater means the heading is too far to the right and needs to be corrected left; anything less and the wheels must be turned to the right. Computer vision will provide continuous real-time data about the RC car's orientation, position, and environment. This data is passed through a Kalman filter to eliminate noise. The filtered data is then passed to the navigation module to determine the best action to take in order to clear obstacles and stay on track. The navigation outputs the current and goal location for the PID controller.

NOTE: The Autobot Racing team will NOT be implementing the Kalman Filter. This algorithm is too complex and too time-prohibitive to create a custom implementation.

5 Computer Vision Design Detail

Computer vision is the primary system used to determine the shape of the course, the location and orientation of each vehicle, and the location of any obstacles on the track. Autobot Racing will use the proven, open-source OpenCV library for all computer vision tasks. For compatibility with the Autobot framework, the Python implementation of OpenCV will be used. Research indicates that performance impacts caused by using the Python rather than C implementation of OpenCV will be negligible.

5.1 Vehicle Tagging

Vehicles will be tagged with unique dot patterns using high-visibility paint dots. The dot patterns will be such that they can be unmistakably recognized as vehicle identifiers, and such that they can be used to determine orientation of the vehicle. These patterns will be applied to dark note cards and attached to the top of each vehicle, allowing for interchangeability between cars.

5.2 Boundary and Obstacle Tagging

Course boundaries will be defined by the edge of the track, and will also follow an easily identifiable color scheme. Obstacles placed in the track will not be tagged. Instead, any unmarked object detected on the track will be classified as an obstacle.

6 Framework/API Design Detail

6.1 Framework

The framework of the program serves to manage all the backend components. This will include handling communication with the Pi, tracking vehicles and their performance, and creating and updating the user interface.
Below is a brief description of each class and its purpose in the program

Class Descriptions:

1. FrameworkManager
   The framework manager class will be the class that manages all classes within the framework

2. CoordinateManager
   The coordinate manager class manage the coordinate system

3. EthernetInterface(Framework)
   The controller communicator class will handle sending all directional commands to the rasberry pi to drive the car

4. UIManager
   The UI Manager class will handle creating and updating the UI on the computer running the program

5. VehicleManager
   The Vehicle Manager class maintains the list of vehicles on the track

6. Vehicle
   The Vehicle class maintains all information pertinent to the vehicles performance

7. RasberryPiManager
   The RasberryPi Manager class will manage the two classes that connect communication between the controller and the program

8. RFInterface
   The RF Interface will handle sending all commands to be executed on the controller

9. EthernetInterface(Pi)
   The Ethernet Interface will handle receiving commands from the framework and sending feedback back to the framework

6.2 Control System Programming Interface

The framework will expose an interface allowing researchers to use their own in-house control systems on individual cars. There will also be several functions in the program that a researcher can use to input their desired commands and retrieve information about a cars performance.

7 Design Issues

7.1 Fully-autonomous vs. centrally controlled vehicles

An early decision was on the level of autonomy of the vehicles. There were two options:

1. Fully autonomous vehicles, with their own onboard cameras, processors, and decision logic.

2. Centrally-controlled vehicles, with one overhead camera and a central decision system directing every car.

Decision: The choice was made to use a centralized control system, rather than fully autonomous vehicles. This decision is more fully described in Section 2, including the rationale for the decision.

7.2 Vehicles: Find a car with a suitable size

A major design issue was finding a car large enough that it can still be seen on the camera, but small enough to navigate a relatively narrow course with other cars present on the track. This is also referenced below, in section 7.3.

1. 4pcs Velocis 1/32 2.4G RC Racing Car Multilayer in Parallel
2. Remote Control Car Off Road 2.4GHz Karting RC Car BG1503
3. Rabing RC Car F1 High Speed 20km/h 1:16 4WD Electric Power 2.4GHZ Drifting Radio Remote Control

Decision: Option one presented the best solution. The cars are relatively cheap and durable, have good reviews, and meet all of the requirements.

7.3 Track: Find a track with a suitable size

Finding a track small enough to fit in a room for presentation purposes, and also be modular for different designs was difficult to find.

1. Build a track
2. Buy a track
3. Project track on floor

Decision: Option one presented the best solution. Building the track will allow for flexibility for changes, should they become necessary. Building the track will also be more cost effective and was the recommended solution from an R/C guru.

7.4 Radio Control

Most 2.4 GHz transmitters available for controlling RC cars are extremely expensive, and are not readily available as individual electronics components.

1. Buy transmitters
2. Reconfigure controllers to allow a Pi to send commands (See section 3.2 for more information.)

Decision: Option two presented the best solution. This option presented the most cost effective solution. This will the simpler route to transmitting directions for the cars to move.

7.5 Vehicle Tagging

Creating a system for vehicle tagging on small RC cars may prove to be challenging. The cars will need to be uniquely identifiable and easily noticeable by the camera.

1. Place high-visibility dot patterns on top of cars
2. Use QR codes for each car
3. Use infrared reflectors on each car

Decision: Option one presented the best solution. The QR codes were not sufficient as they do not easily give orientation information. Infrared reflectors were not feasible because of the large cost of infrared cameras.

Appendices

A Bill of Materials

The following is a list of materials required for successful execution of the project.

- Cars: https://www.banggood.com/4pcs-Velocis-132-2_4G-RC-Racing-Car-Mutiplayer-in-Parall html?rmmds=category&cur_warehouse=CN
- Track: http://www.kenonhobby.com/RCP-Tracks-Mini-96-Kit_p_43466.html
- Camera: TBD, based on available CS hardware and funds.
- Boom: https://www.amazon.com/LimoStudio-Tripod-Photography-Studio-AGG2487/dp/B075MM1PX5/ref=sr_1_1?s=electronics&ie=UTF8&qid=1516811630&sr=1-1&keywords=tall+camera+boom