Design and Implementation of the University of Ottawa IARRC Entry

Jimmy Deng
Team Lead
Ottawa, Canada
jdeng035@uottawa.ca

Yu Jiang
Team Co-Lead
Ottawa, Canada
bjian038@uottawa.ca

Rock Liang
Ottawa, Canada
rlian072@uottawa.ca

Tommy Deng
Ottawa, Canada
tdeng075@uottawa.ca

Adel Rashed
Ottawa, Canada
arash063@uottawa.ca

Miranda Holder
Ottawa, Canada
mhold049@uottawa.ca

David Nduka
Ottawa, Canada
dnduk048@uottawa.ca

Lucas Anderson
Ottawa, Canada
lande051@uottawa.ca

***Abstract*—This document is the submission for the International Autonomous Robot Racing Competition 2018 from Ottabotics, the University of Ottawa's robotics team. This document highlights the conceptual design and the various components of our autonomous racing vehicle.**

I. Introduction

This document is separated into three major sections: Mechanical System, Electrical System, and Software System. Each of these sections will explore the conceptual design of the subsystem, design decisions, and problems we faced.

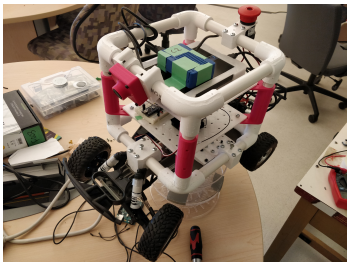

Fig. 1. Image of mostly assembled vehicle.

II. Mechanical System

For the mechanical system, instead of designing and fabricating the frame to custom specifications, we decided to extend the functionality of an existing RC car model. Our vehicle is built upon the frame, motor, gearbox, drivetrain, and steering system of the RC car, which is shown in Figure 2. We extended the functionality of the base RC car by designing and fabricating a roll cage and a multi-layered platform to house the electrical components.

Since this is our first vehicle for the International Autonomous Robot Racing Competition, we decided to design our vehicle with ease of prototyping and modularity in mind. The use of multiple layers of platforms, of which one is shown in Figure 3, within a rectangular roll cage allows the various subsystems to be modular and to be easily removed for testing and prototyping. For example, layer 0 is used for the drivetrain, power, and steering. Layer 1 is used for the electrical and safety systems. Layer 2 is used for anything that involves computer vision and higher level computing such as the mini PC. Additional components, such as sensors, can be easily attached to the roll cage which consists of cylindrical PVC piping and joints.

Fig. 2. Existing RC car model.

The following are a few of the problems that we encountered while designing and fabricating the mechanical system. First, the suspension was not designed for our use case and payload. The default suspension for our RC car frame was designed for off-road, low-load racing, and without modifications, it is not able to carry all of the necessary components for autonomous navigation. This was fixed simply by adding spacers to increase tension in the springs. Second, it was difficult to attach a custom roll cage onto the existing RC car frame because attachment points were not evenly positioned and were not all on the same plane. This was solved by designing and 3D printing custom anchor points/adapters, as shown in Figure 6 and 7, which mount directly onto the existing RC car to provide mounting points throughout the frame. These anchor points also feature embedded nuts to provide ease of mounting for the roll cage and any additional components.

A. CAD Drawings

Figures 3 to 7 show several CAD drawings for varying components.

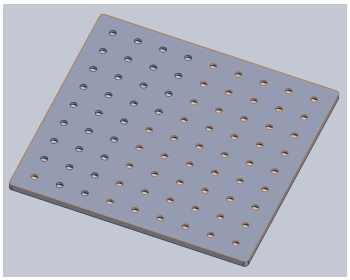

Fig. 3. One of the layers of the electronics platform.

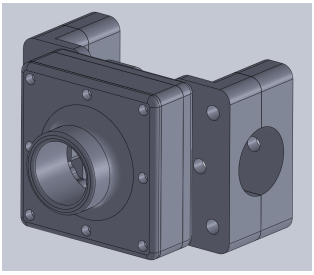

Fig. 4. Camera case.

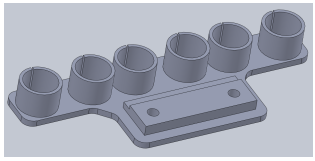

Fig. 5. Ultrasonic sensor array holder.

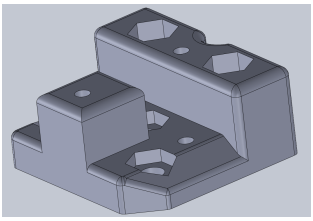

Fig. 6. Front anchor point.

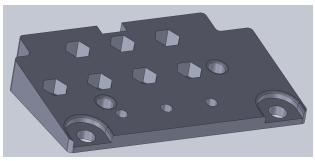

Fig. 7. Left anchor point.

III. Electrical System

The electrical system is centered around an Arduino Nano board. It handles two major inputs and two major outputs. The first input is a message from the mini PC over a serial USB line. This is used to communicate the desired steering angle, vehicle speed, and other miscellaneous messages. The second input is a digital signal from the Arduino slave board to trigger an interrupt in the case where an emergency stop must be performed. The first output is a 1kHz PWM signal to the speed controller to control the speed of the motor. The second output is a 31.25kHz PWM signal to a servo to control the vehicle steering angle.

A. Communication System

The Arduino master board receives serial messages through its USB port from the mini PC, which handles all of the computer vision and processing. Messages can be up to four characters. The most common message that is handled by the Arduino master board is a steering command which consists of the character 'a' and an integer between 0 and 180. Vehicle speed messages have a similar command consisting of the character 's' and an integer ranging from 0 to 255.

B. Safety System

The wireless emergency stop system is used to remotely stop the vehicle in the case where autonomous control becomes sporadic or possibly dangerous. It is configured to act like a dead man's switch where the vehicle will run normally if the switch is pressed, but stop if released. The system repurposes the RF transmitter and receiver from the existing RC car platform, as shown in Figure 9. The RF transmitter consists of two inputs controlled by the operator: a trigger and a rotary switch. The trigger is used to control whether the vehicle runs or stops. The switch is used to restart the vehicle and has two states, *Restart* and *Current*. The *Restart* state restarts the program while the *Current* state does not change vehicle's state. The two outputs of the receiver are connected to the Arduino slave board. The outputs send two PWM signals, *Signal 1* and *Signal 2*, in response to the trigger and switch respectively, and the Arduino board reads the amount of time the signals are *High*. For the vehicle to start, the trigger must be in its *Pulled* state, and the switch must be in the *Current* state. To stop the vehicle, the trigger must be *Released*. The vehicle will also stop if the connection between the transmitter and receiver is interrupted or lost. To restart the vehicle, the trigger must be *Pulled*, and the switch must be turned to its *Restart* state. After the vehicle restarts, the switch should be immediately turned to the *Run* state to prevent continuous restarting of the program. The various states of the system can be referenced in Table I.

C. Speed Controller

The original speed controller that came with the RC car, XL-5, was replaced with a Pololu High-Power motor driver. This new speed controller design allows for simple fault detection. The three faults displayed are short circuit across the motor, overtemperature and under voltage. When the short circuit fault occurs, the car stops until the reset pin is pulled low. This is resolved with a simple button switch between the reset pin and the grounding node.

D. Power System

Our power system consists of two independent power sources. An 8.2V battery is used for the drivetrain, which consists of a motor and a speed controller. A 5V battery is used for the vision and communication systems, which consists of the mini PC, Arduino Nano boards, and various sensors.

IV. Software System

The context of the problem involves localizing and lane detection to assist the vehicle to perform intellectual decision during driving. The breakdown of the software involves camera calibration, image preprocessing, object detection, and predicting angles of changes based on image and object avoidance.

A. Camera Calibration

A non-calibrated camera can result in false geometric perception rather than perceiving actual shape (known as

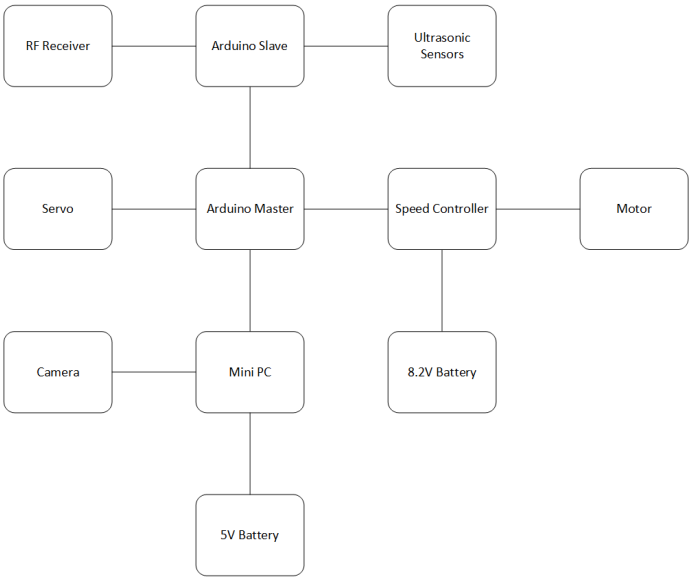

Fig. 8. Electrical diagram.

TABLE I
SAFETY SYSTEM STATES

RF Connection	Trigger	Switch	Signal 1 High Time (ms)	Signal 2 High Time (ms)	Vehicle State
Interrupted / Lost	x	x	x	x	Stops
Connected	Released	Run	$T < 953$ OR $T > 1113$	$T < 1948$ OR $T > 1113$	Stops
Connected	Released	Restart	$T < 953$ OR $T > 1113$	$1948 < T < 2008$	Stops
Connected	Pulled	Run	$953 < T < 1113$	$T < 1948$ OR $T > 1113$	Runs
Connected	Pulled	Restart	$953 < T < 1113$	$1948 < T < 2008$	Restarts

Fig. 9. Traxxas two-channel controller with the trigger and rotary switch indicated in red and green respectively.

distortion), thus calibrating the camera can prevent the vehicle from receiving false information and intrinsic parameters for localization. In photography, a distorted image can cause false geometric perception of the object and its surrounding known as perspective distortion. This changes the actual perceived length of the object. As Figure 10 shows, the square chessboard appears to be distorted into a spherical-squarish shape. In which the height and width of the object is not the same as our own perception. In Figure 10, the chessboard appears similar to an actual square. This known as perspective distortion; this is a transformation of an object and its surrounding when its perception significantly differs from its actual shape. OpenCV provided optimized solutions with function `findChessboardCorner` to extract the intrinsic matrix of the camera. Camera calibration is inevitable step for localization, which can help the car to accurately perceive vision data.

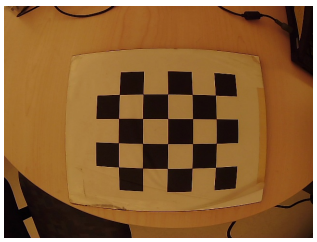

Fig. 10. Distorted image.

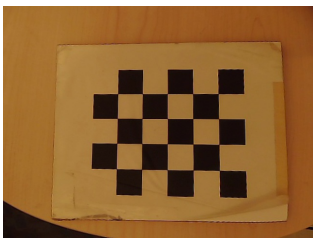

Fig. 11. Undistorted image.

B. *Traffic Light Detection*

The traffic light is detected by first isolating it within an image and then filtering the colour to determine the state.

Isolating the traffic light involves using the scale-invariant feature transform (SIFT) algorithm implemented in OpenCV to match a given reference image to an area in a different query image. If provided sufficient reference images, the process will return the portion of a different image that is most similar to the reference image that also meets an user-defined similarity threshold.

Once the traffic light is localized, the state of it (red or green for 'stop' and 'go') is found through filtering for prominent colours. Typically, images are stored in an Red-Green-Blue (RGB) based colour space which is not suitable for determining a range of colours (i.e. shades of green of varying intensities). For this reason, a Hue-Saturation-Value (HSV) colour space, which has properties that allows for specifying colour ranges, is used instead. Upper and lower bounds of colour values are manually defined for the red and green colours based on the reference images. The state is decided if there are enough red or green pixels in the image to pass a minimum threshold. See Figure 12 and Figure 13 for a visual example.

Fig. 12. Red traffic light template image.

Fig. 13. Red traffic light mask.

C. Video Streaming (Interframe compression)

To reduce packet size when streaming video from the vehicle to an external display, only the difference between the current and previous frame is sent. An initial 'soft mask' is found with OpenCV's `absdiff` function of which any non-black pixel is considered different and will be sent. Figure 14 shows the different stages an image undergos from the original to the reconstructed image.

Fig. 14. Stages of the original and reconstructed image.

D. Communication System

The communication between the mini PC and the master Arduino board is done through serial over a USB connection. The implementation on the mini PC side is done using PySerial, a Python library. Commands for vehicle speed, steering angle, and other miscellaneous commands are sent as one to four character packages to maximize throughput and latency. Acknowledgement responses from the master Arduino board are read using a separate concurrently running thread.

E. Pathfinding and Steering System

We are taking a purely vision based approach for the pathfinding and steering system. The first step, aside from any preprocessing, is to determine the boundaries, if any, in front of the vehicle. In the case of this competition, boundaries are represented by yellow and while lines of tape. Line segments are filtered out from the preprocessed video stream from the camera using OpenCV's `HoughLinesP` function which returns a set of line segments represented by start and end points. These line segments are then filtered by angle and clustered into groupings of similar angles. Finally, the resulting turn angle is determined by computing a weighted sum of the dominant turn angles from the bottom of the frame to the top.