Design of an Autonomous Racing Vehicle

Oluwatoni Ogunmade, Jack Xu, Tsugumi Murata, Adrian Malanaran, Angela Gu, Brian Tran, Andrew Jin, Waleed Ahmed, Brian Kibazohi, Jun-Ha Jung

***Abstract*—This technical report describes the University of Waterloo's team approach to the International Autonomous Robot Racing Competition(IARRC). We developed a system capable of real-time autonomous driving utilizing off-the-shelf parts including three cameras, an Inertial Measurement Unit(IMU), a quadrature encoder and a Hokuyo planar Lidar. The vehicle is a modified 1:10th scale Traxxas RC car with electronics to support battery monitoring, power distribution, remote control and emergency stopping. A robust and modular aluminum superstructure was built as a retrofit system for the vehicle, to provide a reliable and accessible platform for developing autonomous cruising system. A cutting edge neural net based traffic light detection system was added to improve robustness. A new path planner was designed relying on a Random-exploring Random Tree(RRT) planner, a Model Predictive Controller(MPC) for longer range planning and a trajectory rollout algorithm for short range evasive maneuvers.**

In this paper we will focus on innovations in our overall approach by highlighting our computer vision, system architecture, software development practices, path planning and controls.

I. Introduction

The International Autonomous Robot Racing Competition (IARRC) is an annual event in which fully autonomous vehicles are tested in a race. Vehicles compete in two major races; a drag race focusing on high-speed control, and a circuit race focusing on safe maneuvering of a complex environment.

This report will focus on the mechanical, electrical and predominantly the software design of the robot. Advances over the designs from previous years will be demonstrated, along with the well-performing components which were retained. The following report is divided into the sections of problem definitions, mechanical design, design, software design, and conclusions.

II. Problem definition

The design of the robot stems from the list of engineering requirements and objectives identified in the rules and implied in the details of the competition. The requirements are absolutely mandatory features the vehicles needs to meet. The objectives are performance targets derived from the races which the vehicle needs to complete.

A. Race Requirements

- The vehicle must not exceed 75cm long, 55cm wide, 60cm tall.
- The vehicle must not interfere with other robot's operation, such as their sensing and navigation.
- The vehicle must detect the traffic light change reliably in varying conditions.
- The vehicle must be able to safely stop within 1 meter of crossing the finish line.

Contributors are from the Departments of Mathematics and Engineering at the University of Waterloo and the Department of Engineering at McMaster University.

- The vehicle must have a wired and wireless emergency stop button.

B. Objectives

- The vehicle should be able to achieve 10 m/s at drag race.
- The vehicle should be able to drive higher than average 3 m/s overall during the drag race.
- The vehicle should be able to detect other robots and avoid collisions.
- The vehicle should be able to detect obstacles and lines at 30 Hz
- The vehicle must be able to survive a collision and inversion at 10 m/s.
- All parts should be easily accessible; can be replaced in fewer than 10 minutes.

III. Mechanical Design

This year, a new mechanical retrofit system has been designed and built based on previous years' feedback. This year's mechanical system is much simpler, robust, modular, and rigid to provide a much more stable and easier developing platform.

A. Body

As previous years, the body of the robot is designed to be a retrofit system for a Traxxas RC monster truck. To make this year's chassis much more robust and rigid to protect on-board hard-wares, while keeping its simplicity and flexibility. The entire chassis is mainly made of aluminum angle extrusions, sheet, and 3D printed ABS components to provide a secure and light-weight system. The main body (as shown in Figure 1) can be composed with three major parts: a wire frame superstructure, a build plate, and other mounting components.

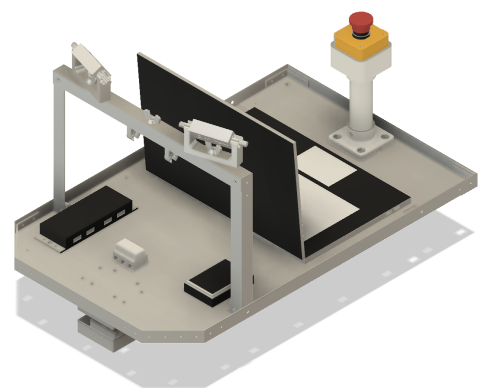

Fig. 1: The Fusion CAD model of the body

B. Wire Frame Superstructure

The wire frame superstructure is the core component that bridges and secures other components. The wire frame is mainly made of aluminum materials. It is composed with three parts: a retrofit mounting system, a protective cage, and a camera rig. For retrofit mounting system, it is built with two aluminum clevis with two angle extrusions attached horizontally. The protective cage is made of aluminum in a trapezoid shape. The base cage can provide far enough strength to prevent the build plate away from any torsion or shear stresses from external forces or impacts. Similarly, the camera rig is also made of aluminum angle extrusions to provide a stable and flat platform for cameras.

C. Build Plate

The main part of the build plate is a $\frac{1}{16}in.$ aluminum sheet, attached inside the protective cage securely. Since the build plate is entirely independent from the superstructure, it can be freely accessed and modified easily and independently without dissembling other structures. This can make the robot much easier to maintain and develop separately without complicated dis-assembly processes. On the bottom side of the plate, the lidar is also directly attached onto the sheet to remain protected and rigidly, while providing a clear detecting region. On the top side of the plate, hardware components such as a MCU (Micro Controller Unit) and a laptop can be safely mounted in place during the race.

D. Other Mounting Components

Other mounting components are mainly about the mounting system for mechanical E-Stop, and cameras. To keep the E-Stop in the required height and accessible area, a pillar with the combination of 3D printed mounting base and housing and an aluminum pipe was developed to elevate the E-Stop at the right height (as shown in Figure 2). In addition, the E-Stop pillar is installed right onto the wire frame superstructure to keep it securely and rigidly in place. For cameras, web cameras used in the previous competitions are kept to be used this year. To reduce the wobbliness of camera due to weak mounting mechanisms and sudden motion, a camera housing and mounting systems (as shown in Figure 3) are designed in Fusion and 3D printed with ABS filaments. At the surface of each joint a rough triangle friction disk is designed to fix the camera while allowing the team able to freely adjust the camera angles to optimal positions. Camera mounts can then be placed onto the camera rig and fixed with bolts and nuts.

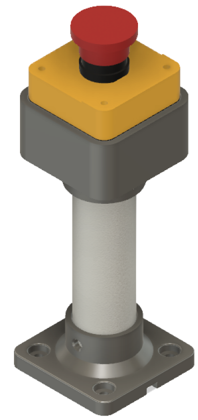

Fig. 2: E-Stop mount in Fusion

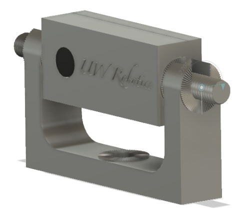

Fig. 3: Camera mount in Fusion

IV. Electrical Design

The electrical design of the robot is based on the previous years design with new features introduced. The innovations in our electrical design can be categorized into sections including sensors, actuators, and the Arduino Shield.

A. Power

Two 12V 5800mAh Li-Po batteries, connected in series, provide the main 24V power rail to the robot. This 24V is directly used by the brush-less motor. To monitor the status of battery, a battery management system is integrated on the Arduino Shield. The system monitors the voltage of the battery and it relays thatto the main computer.

B. Motor Controls

The motors used for powering the vehicle comprise of two steering servos and one brush-less drive motor. The drive motor was controlled by a Castle Creations Electronic Speed Control unit (ESC). These motors are all controlled by the Arduino via PWM servo commands.

In the event the Arduino PWM signal is neutral or missing, the ESC causes the drive motor to brake when powered.

C. Sensors

As described in the electrical design section, encoder counter is built from scratch using a 8-bit AVR MCU. The MCU which is programmed in low level C is capable to detect 512 encoder counts per revolution. The two encoder pulses are each detected as an external interrupt, and depending on the how the pulses change, the rotation direction and speed of the motor can be determined. The MCU communicates with the main Arduino via I2C, and returns the encoder count as unsigned 16 bit when the Arduino requests it. The calculation to convert from the encoder count to distance and speed is done on the Arduino side.

$$V_{car} = \frac{d}{dt}\left(\frac{Count_{encoder}}{256}\right) \cdot (R_{gear}) \cdot (\pi D_{wheel}) \quad (1)$$

The Hokuyo 2D lidar provides the vehicle with range data for any obstacles within 30m, in a 270°plane in front of the vehicle.

The vision system comprises 3 cameras, two for line detection and one for traffic light detection. The occupancy grid is used to represent obstacles around the robot and this is created from a combination of the lidar data and camera data.

V. SOFTWARE DESIGN

The improvements over last year's software fall into three main categories: Architectural, Software development and testing, computer vision improvements and optimizations to sustaining higher speeds. These improvements allowed the team to radically improve our throughput and code quality while simultaneously increasing the performance of our robot.

A. Software Development and Testing

A holistic approach was taken to identify the limitations of the teams software development velocity. The main culprits were poor code documentation and lack of accessible testing. To improve our code documentation the code base was completely refactored with a focus on clearer comments and aligning with a coherent coding style. This greatly improved readability and allowed new team members to get up to speed very quickly. To ensure that the new standards were carried forward, a code review was made compulsory before all code submissions. To improve our software testing, there we're two major improvements made over the previous year. Firstly a simulation of the race and our competition vehicle were made. This simulation was developed using gazebo and ROS. The tracks were simulated to closely resemble the race environment with traffic cones and lane markings. The car was simulated with all the sensors currently on board the physical vehicle. The simulations allowed the software team to test both individual features and the entire system while the physical vehicle was still being built.

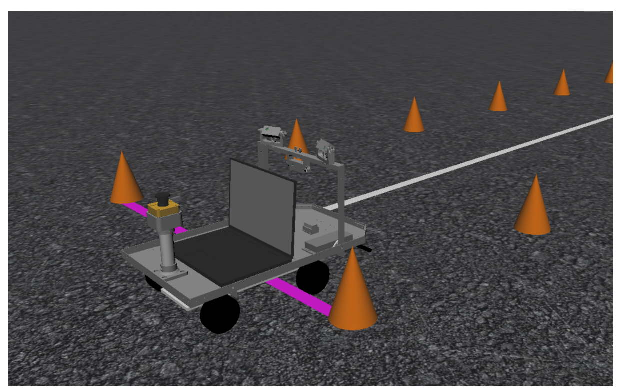

Fig. 4: Simulated Robot

Secondly, we relied on recorded data from the past years competition to validate our code. A repository was created, containing the sensor data from previous competitions. This was used to validate our computer vision software and demonstrate it's superiority to software from the previous years.

B. Architectural

In the previous structure each node was responsible for keeping track of the current state of the robot with no node responsible of keeping track of metrics or overseeing the operation of the vehicle. The new software architecture is illustrated below, with the addition of the supervisor node at the helm of the system. As seen in 5.

The sensing software consists of device drivers and ROS nodes that publish sensor data to other nodes to be consumed. Most of the software in this section is provided by vendors or the open source community. The path planning and controls sections are responsible for determining where to go and keeping the car on that path. The signal detection section consists of ROS nodes responsible for detecting important signals like a traffic light changes and the line signifying the end of the race. They communicate these events to the supervisor which then decides on the course of action. The supervisor node is a module designed to manage high level decisions and keep track of useful race metrics. It provides an interface for signal detection nodes to indicate that an event has occurred. Using it's knowledge of the current state of the robot, it decides when to start and stop the car. The supervisor node also keeps track of some useful variables; namely the race type, start and end times, lap count, average speed, and battery life. When the race is finished, a few of these variables are stored and saved into a text file. The race is started when a traffic light signal is received, and is ended when either the lap count is reached for that race, which is incremented by an end line detection signal, or if the battery life dips below 10

C. Computer Vision

This competition year huge strides we're made in improving the robustness and functionality of our computer vision software. To improve the robustness of our lane detection system a shadow removal algorithm was added as a pre-processing step on each of our cameras. To improve our traffic light detection in all conditions a machine learning classifier was added to help detect traffic lights in images. End line detection was improved considerably over past years as well, largely due to the addition of a new hysteresis filter on the camera stream.

1) Shadow Removal: In last year's competition, the robot had difficulty in reliably tracking lane lines under variable lighting conditions, such as when buildings or robots cast shadows onto the track. As a result, this year's team implemented a shadow detection and removal pre-processing step, based on work done by Murali et al[2] in order to address this issue. First, to detect shadows, the algorithm calculates the mean value of each plane in the Lab colour space and applies a threshold on the L and b values. The Lab colour space was designed to be uniform with respect to human vision, so it could better model visually perceived changes, such as variations in lightness that would indicate a shadow region. An example is shown in Figure 6.

To prepare for shadow removal, the shadow mask is dilated to smooth out holes and sharp edges and then separated into shadow regions. For each shadow region, the algorithm calculates the ratio of red, green, and blue values between the area inside the mask and the area just outside the mask. To remove shadows, the shadow region's RGB values are multiplied by these ratios.

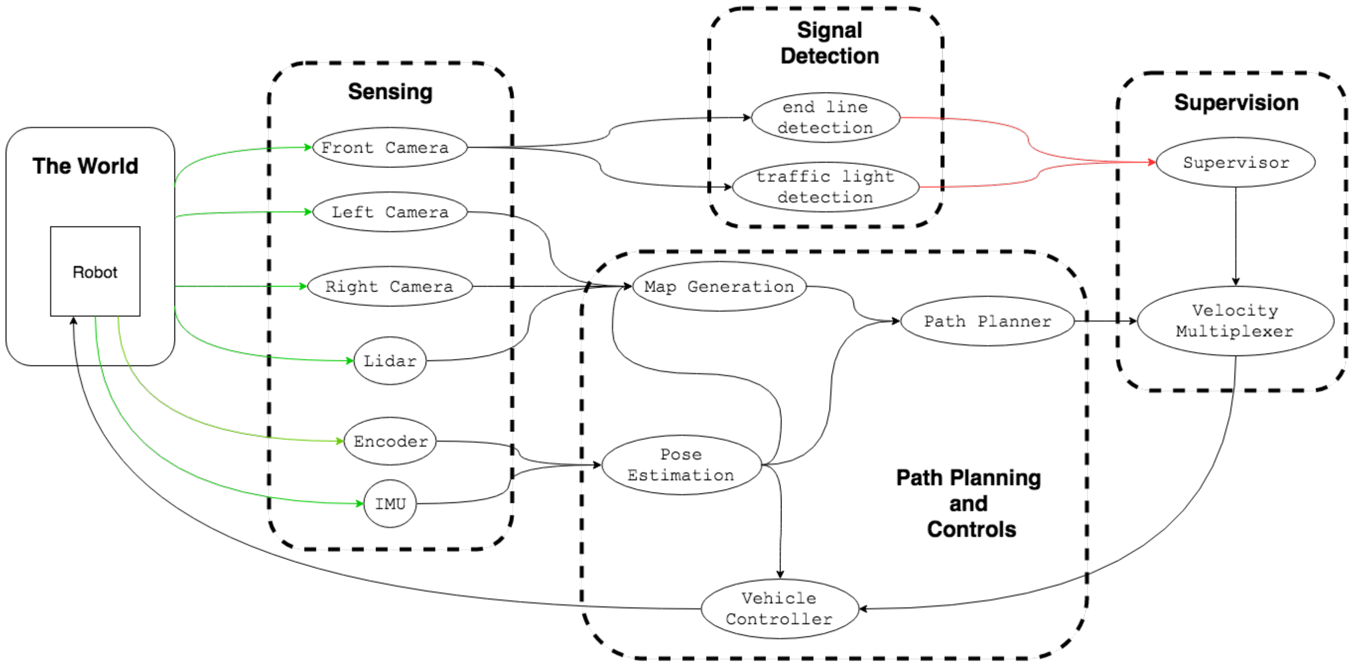

Fig. 5: The Software Architecture

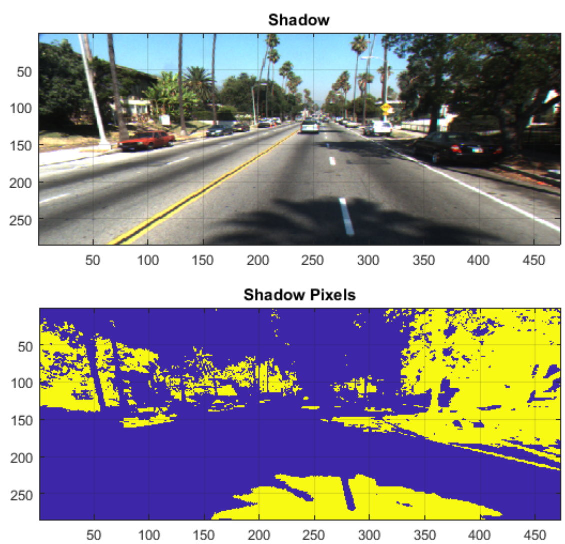

Fig. 6: Shadow Detection Using Lab Colour Space

The last, and slowest, step is to correct for over-illuminated edges in the shadow regions, since the outer edges of shadows tend to be brighter than the inner region. To do this, a median filter is applied on the edges of each shadow region.

There were two issues the team encountered with this shadow detection and removal approach. The first was that the over-illumination correction took the most time to run, but often blurred out lane lines through its median filter, especially if the shadow was spotty and had edges in its interior. However, the team did not expect to encounter this situation in competition and the resulting images were often more useful for line tracking than the originals. An alternative method to correct for over-illumination could be considered in the future. Second, since the shadow detection relies on a mean value for the lightness (L) over the entire image, a frame with extreme highlights can cause the algorithm to misidentify shadows and over-illuminate entire regions. Again, this was not expected to be seen in competition since this issue was most common with a blown-out sky region and the robot's cameras are directed towards the ground.

Figure 7 displays the final result, showing the lane detection before and after the shadow removal pre-processing step for a test image.

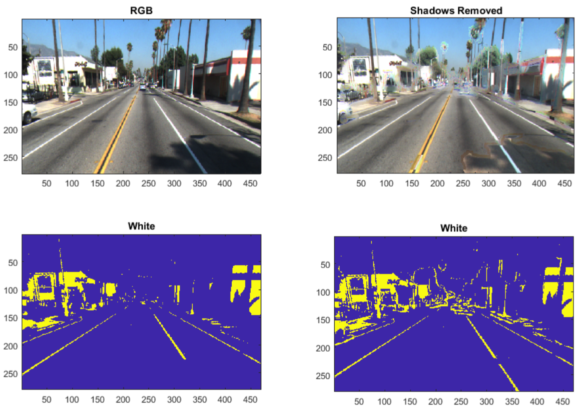

Fig. 7: White Line Detection Before and After Shadow Removal

2) Traffic Light Detection: The traffic light detection process is initiated starting with availability of image data supplied by the camera sensors, which is relayed in the form of a mutable message data structure to the system nodes. Detection of the traffic light object is performed using YOLO (You Only Look Once)[1], a neural net based object classifier which is integrated as a separate system node responsible for identifying and outlining the location of specific objects within an given image 8. As an example, Figure 1.0 shows YOLO detecting a traffic light object from camera image.

Once a traffic light object is detected, the camera image is cropped to the bounding box of the traffic light, which is then sent over to the traffic light processor node. Within this section of the system, the image, existing in RGB color space is converted into a hue, value, saturation (HSV) space. Gaussian blur is also applied to the image to smooth out irregularities and remove noise to increase analysis accuracy. The state of the traffic light is detected by applying a filter that isolates pixels that correspond to a certain color range to distinguish red and green colors

Fig. 8: White Line Detection Before and After Shadow Removal

in a diverse set of lighting conditions. The traffic light color state is determined based on of the prominence in either color value. Upon detection of a green traffic light, a service call is invoked, signalling the start of the race. This was able to detect traffic lights in 93% of the time in contrast to our previous algorithm which only worked on 75% of samples provided.

3) End line Detection: To detect the end line, the image is converted to the HSV color-space and filtered for Magenta. A Gaussian blur is then performed on the image to reduce small detection errors. Finally, blob detection is used to filter for minimum area and non-circularity. This detection algorithm is inserted between a hysteresis counter procedure, that increments when detection is true and decrements when detection is false. This prevents false positive detection of the algorithm. Once complete a service call is invoked signalling the completion of a lap.

D. Path Planning and Controls

The path planning framework was overhauled from the previous year. Whereas last year's path planner was entirely reactive, the new path planner incorporates a look ahead distance whenever lane markings are present. It uses these lane markings to pick a point ahead in space to plan a path to, this helps it sustain higher speeds for longer stretches of time. The path is continually reassessed and planned and a fairly low frequency of 1 Hz. If any obstacles are detected along the path closer than a certain threshold the path planner falls back to the trajectory roll-out algorithm which can run at a much higher frequency of up to 20Hz. To remain on the path that was selected the robot utilizes an MPC. This is updated with the robots estimate of its current position by a sensor fusion ROS node. The MPC computes the error between where the robot is and where it should be, often referred to as the cross track error and generates steering and throttle commands to follow the path.

1) Medium Range Planning: To plan a feasible path to a position farther ahead an RRT planner is employed. When provided a point along the track farther ahead of it. The planner takes the kinematic model into account and generates an obstacle free path for the robot to follow.

2) Close Range Obstacle Avoidance: The trajectory rollout algorithm from previous year is preserved and it is responsible for engaging in close range evasive maneuvers when unforeseen obstacles arise. These obstacles could be other cars on the track or humans that might cross the track.

3) Sensor Fusion and Controls: The velocity information from the encoder and orientation and acceleration information from IMU were fused together using an Extended Kalman Filter to provide the path planner with the vehicles pose. The MPC uses the vehicle pose and the generated path to generate steering and throttle commands for the robot. The MPC is superior to our previous sole PID controller because it computes commands with a knowledge of the kinematics of the vehicle.

VI. Conclusion

Thus far, the success of the current design can only be assessed relative to the performance of the previous vehicle. Many weak points were addressed, and significant performance improvements were made. The body was redesigned to be more durable, repairable and serviceable. The traffic light detection strategy was made much more robust, and has been tested successfully in varied environments. The path planner was improved to enable sustained high speed driving and retain vehicle responsiveness. The teams software development practices were updated and enforced.

Acknowledgment

The authors would like to thank members of the WAVE lab who provided hardware used on the robot. They would also like to thank Matthew Post for his mentorship and continued support of the team.

Additional note - All work done on changes to mechanical, electrical and software design was performed by team members who had no former experience with the competition, while returning members took on a mentoring role. The new members should be especially noted for their dedication, work ethic and ingenuity. They made this happen.

References

[1] J. Redmond, R. Girshick, A. Farhadi, and S. Divvala, *You Only Look Once: Unified, Real-Time Object Detection*. Washington, tech, 2015

[2] Murali, Saritha and V K, Govindan. *Shadow Detection and Removal from a Single Image Using LAB Color Space*. Cybernetics and Information Technologies, 2013.