IARRC 2018: Fenrir

Sheila Afros, Andy Bao, Daniel Cheriyan, Michael Dennisov, Devrim Gunal, Taehoon Kim, Alexander Kitaev, Alexander Kropp, Adeeb Mahmud, Nicholas Pfeifle, Afzalur Rahman, Kailey Williamson, Anita Yang, Kevin Zhou

Viking Robotics
Technology, Science, and Business Departments
Waterloo Collegiate Institute
Waterloo, ON. N2L 3P2
Canada
wci.wrdsb.ca

***Abstract* --- Autonomous vehicles have become a popular and lucrative concept in the technology sector. With applications from forestry, to agriculture and transportation, commercial ventures are popping up daily. Viking Robotics' Fenrir aims to further the field through competition in the International Autonomous Robot Racing Competition. Built using commercially available components carefully interfaced and wired, the vehicle is able to maintain position in a given course while avoiding collisions and maintaining a high standard of safety. It is equipped with ultrasonic, proprioceptive, and visual sensors. These provide more than adequate data for finding and maintaining course position and speed. Working without mandatory external connections allows for higher processing speeds and allows for quicker roadway travel. In summary, this allows for Fenrir to concurrently travel at speed while maintaining course and high standards of safety.**

I. INTRODUCTION

Waterloo Collegiate's Robotics Team returns from a two year hiatus in high spirits. With members from the ages of fourteen to eighteen they vary in experience from nearly none to venerable masters. The Team from Waterloo, Ontario finds that the disparity is more a boon than a mal. Fresh radical ideas on programming, wiring, interfacing and more give a unique competitive edge. Competing at the International Autonomous Robot Racing Competition (IARRC) is a legacy experience for all members.

Building on previous years of experience allowed for a strong start in 2017. A greater understanding of all aspects of the vehicle was earned through manual construction of each component and hours of practice. Together these allowed for the focus to shift to the algorithms and proper setup of the vehicle itself. The IARRC provides a satisfying challenge to these highschool students, allowing for them to hone their skills while gathering valuable experience in the field of automation.

The IARRC is a competition run by the University of Waterloo to demonstrate the work of engineering students across the globe in their efforts to improve robotic automation and design. The entry by Viking Robotics strives to demonstrate that anyone can be successful in the field and that there is no exclusive cutting edge technology that is required. Using exclusively the resources available to a highschool team it is the paragon of innovation and resourcefulness.

II. Design Parameters

A. Requirements

- The vehicle must not exceed a maximum length of 75 cm, a maximum width of 55 cm, a maximum height of 60 cm, or a maximum weight of 20 kg.
- The vehicle must be propelled entirely by battery power.
- A mechanical E-Stop must be present with a red button at a minimum height of 30 cm.
- The wireless E-Stop must operate over a minimum range of 50 feet. The wireless E-Stop must stop the vehicle's motion within 5 seconds.
- The vehicle must not interfere with other vehicles' sensing or navigation abilities
- The vehicle must be able to drive in a straight line at a constant speed of 2m/s for 10 m and come to a stop within 5 m, limited to operate up to 10 m/s maximum.

B. *Objectives*

- The vehicle should be able to actively detect and avoid collisions with other competitors.
- The vehicle should consistently respond to starting signals in less than ten seconds
- The vehicle should be able to process video internally, and operate without requiring external starting commands or wifi connections

While some aspects of the vehicle were mandated by official others were considered to be good practice and beneficial to the overall result of the team. Collision avoidance provides the possibility of a much greater score and prevents component damage. Rapid responses allow for a further reduction in time, and paired with a requirement to operate "closed circuit" reduces the chances for external complications to propagate. By providing additional goals for the project a higher standard of excellence can be strove for and achieved.

III. MECHANICAL

A. *Chassis*

The chassis built around is a Traxxas Slash. This model was familiar and has a moderately low price point. This allowed for more risk taking in design as there was less of a sunk cost exhibited. Modifications to the chassis include a hockey puck taped on to act like a sacrificial bumper in case of any frontal collision and alteration of the gear ratio to limit maximum speed to regulation. The steering servo and the electronic speed control were disconnected from the original radio receiver. Note that the receiver was left on to diagnose any toe and camber problem that the car may suffer from and to make it easier to restore the car to its original condition for future use. Instead of the radio receiver, the ESC and servo were connected to an Arduino UNO, which sends signals to go forwards, back, or brake, and to steer left or right based on serial signals sent from the Beaglebone Black.

Fig 1 Stock Traxxas Slash Chassis

B. Further Modifications

With the chassis selected several modifications occurred to make Fenrir “race-ready”. The shell of the car was removed and replaced with a platform that was secured with inlaid metal clips. Here, the Arduino UNO, breadboard, Beaglebone Black, Crazyfire camera, USB hub and the emergency stop are mounted. A location to place electronics for easy maintenance proved useful for debugging, but left it vulnerable to collisions. Sponge and foam are located about to secure all components in place. Next reinforcing beams were attached to the rear to support the E-stop tower. Through the orientation of the wood work and force applied is carried through the metal superstructure to the wooden crossbar. From there it has been carefully ensure that the energy is near evenly transmitted to both the longitudinal and transverse directions of the platform. The lengths of these crossbars encourage the force to travel into the rear suspension when the E-stop is utilized, and any additional is mitigated through the flexible platform and front suspension. By providing a varying rigid surface it allows for some components in other areas, notably the camera, to be elasticised comparatively. The springiness of the platform and the camera’s mounting at the nose of the vehicle ensure that the heavy loads on the vehicle a centered around rigid sections, and allow for forces to propagate and mitigate as they travel. Due to the placement of loads on the platform this implies that under applied stress, or when momentum affects the platform minimal impact affects the sensors. Through modification Fenrir is given its distinct touches that create an environment rich in sensory stability and that mitigate any possible damages and forces acting upon it

C. *Interfacing*

A USB hub serves as the connection point between the camera, Beaglebone and Arduino UNO. It also provides power to all three components via a phone power bank. The Beaglebone was chosen due to its lightweight

Fig 2: Crazyfire Action Camera

and durable nature, in addition to prior experience in working with the system. It has the ability to run a serial connection to the Arduino, easing command transmission, while additionally connecting to the Crazyfire camera. Since all the components can take power over USB, this eased the connections and made interfacing a snap. (See Fig 8 below)

Fig 3: Insignia USB Hub

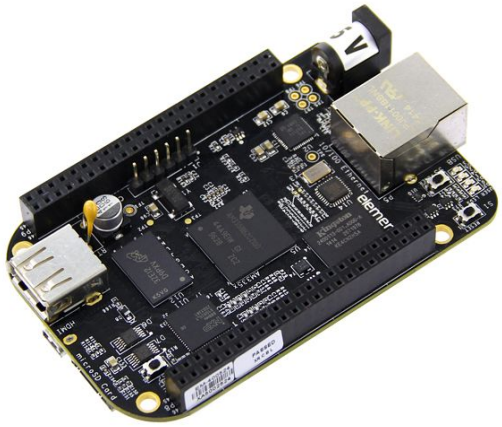


Fig 4: Beaglebone Black Microprocessor

IV. SOFTWARE

A. Parallel Processing

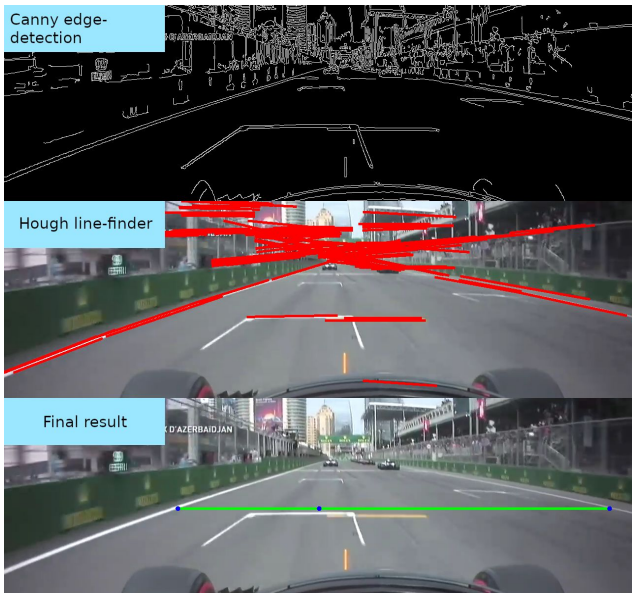
Due to our limited hardware, our software teams were required to design control software that was both horizontally and vertically scalable. The Beaglebone only includes a single,

low-throughput ARM core but real-time image processing often necessitates more faster hardware. Our club does however, have access to multiple Beaglebones which we can chain together or a single Raspberry PI that contains a quad-core processor. In order to write software the fully utilizes multiple processors and/or multiple cores, we had to architect our system to use parallel processing and have fully-serializable internal states. Therefore, we chose to write our software in Kotlin (a JVM-based programming language) using the Java OpenCV 3 bindings.

B. The Steering-Dot Algorithm

Our main steering algorithm uses a combination of the Hough line-finding algorithm along with a simple, custom algorithm to determine the direction to steer. A brief outline of the algorithm that runs on every video frame follows:

1. Convert video frame to a grayscale image
2. Run Canny edge-detection on the grayscale image
3. Run hough line-detection on the output of the previous algorithm
4. Draw a horizontal line across the image. The vertical position of the horizontal line may vary depending on camera/vehicle parameters.
5. Place a dot in the horizontal center of the image at the same vertical position of the horizontal line
6. Starting from the center of the image and sweeping towards the left, find the first intersect between the horizontal line and a line found during the Hough line-finding process. Place a dot there.
7. Repeat step 6 but sweep toward the right instead of the left.
8. The distance from the left-dot to the center-dot is the distance from the vehicle to the left side of the road. The distance from the right-dot to the center dot is the distance from the vehicle to the right side of the road.
9. Steer the vehicle depending on the two distances determined above.



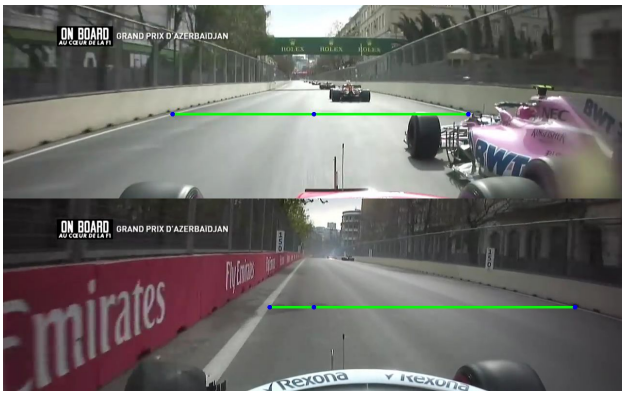


Fig 5: Full analysis of sample footage

Above, we visualize the algorithm using some example racing footage. The green line represents the horizontal line we draw in step 4. In the left image series, we show the video frame as it goes through various stages of image processing. In the above-right image, we show a scenario where no steering is necessary: the distances from the left and right dots to the center dot are equal. In the bottom-right image, we show a scenario where the robot should steer right: the distance from the right dot to the center dot is larger than the distance from the left dot to the center dot.

C. The Edge-Sweep Algorithm

In some cases, the steering-dot algorithm does not give accurate results or gives no result at all. There are many cases in which this can happen: a turn is too sharp/road zig-zags (the Hough-lines algorithm only works on mostly straight lines), the road lines are too faint to detect, a competing robot is blocking the road lines, etc… To ensure the robot continues to operate properly in these situations, we must have an alternative algorithm that the robot will fallback to if it is unable to find the left and/or right dots after running the steering-dot algorithm. Most of these issues only exist due to the inflexibility of the Hough-lines algorithm. Refer to the below example to find a scenario where even though Canny was able to detect an

edge on the right road-line, hough-lines was unable to detect a line at the same location.

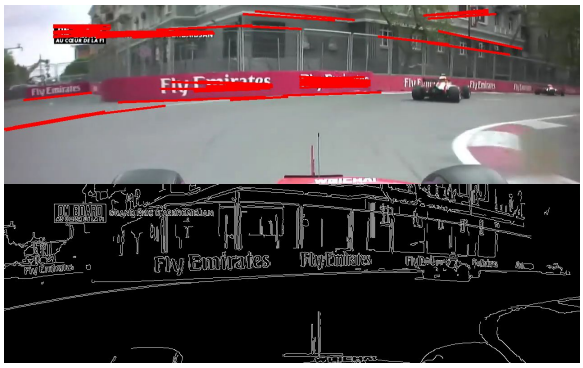


Fig 6: Hough-lines analysis

We reached the conclusion that we can trade off reliability and accuracy for consistency by running a variant of the steering-dot algorithm directly on the Canny edge-detection result instead of the lines detected by Hough-lines. Therefore the Edge-sweep algorithm was born, it is described below:

1. Convert video frame to a grayscale image
2. Run Canny edge-detection on the grayscale image
3. Draw a horizontal line across the image. The vertical position of the horizontal line may vary depending on camera/vehicle parameters.
4. Place a dot in the horizontal center of the image at the same vertical position of the horizontal line
5. Starting from the center of the image and sweeping towards the left, find the first intersect between the horizontal line and a white pixel found during the Canny edge-detection process. Place a dot there.
6. Repeat step 6 but sweep toward the right instead of the left.
7. The distance from the left-dot to the center-dot is the distance from the vehicle to the left side of the road. The distance from the right-dot to the center dot is the distance from the vehicle to the right side of the road.
8. Steer the vehicle depending on the two distances determined above.

To avoid undesired steering when the horizontal line intersects edges that do not belong to the road lines, the edge-sweep algorithm is only run when the steering-dot algorithm fails.

Also notice that the edge-sweep algorithm is able to avoid other robots while the steering-dot algorithm is not able to since the Hough-lines algorithm filters out other robots from the result.

D. Impulse steering prevention/outlier removal

In cases where the new steering direction differs too much from the steering direction determined in the previous frame, the software will instead select a reasonable compromise between the two directions to prevent outliers from causing the robot from making sudden turns.

E. Start signal detection

Basic template matching is used to locate the traffic light in the video frame. Then simple color calculations can be used to determine the state of the traffic light.

V. Safety Systems

In order to best operate within a strict parameter of safety the vehicle incorporates several safety systems into its design. The first of these are

Fig 7: Emergency Stop button used for Fenrir

manually activated, namely the Emergency Stops (E-stops). Physically located onboard the chassis is the E-stop tower. This metal construction sets the button at a height of around 45cm and is bright red to bring attention to itself. Upon pressing it cuts power to all components associated with the motors causing the vehicle to coast to a stop. Next is the wireless E-stop. Built off of the HC-06 Bluetooth transmitter/receiver it has an effective range of 65 ft (unobstructed view) and can be connected to a laptop or phone. When triggered it alerts the Arduino microprocessor to set the

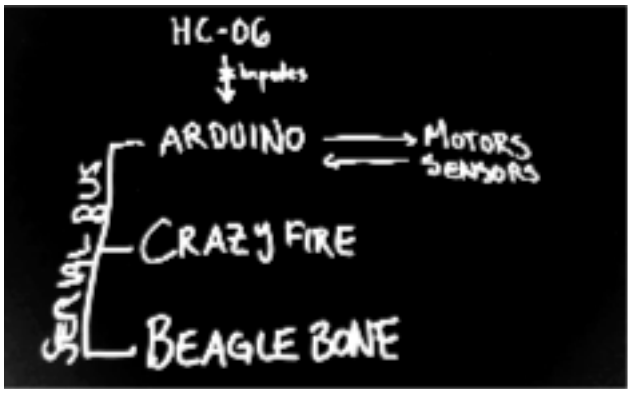


Figure 8: Diagram of Interfacing Setup

speed to zero decelerating until the proprioceptive sensors show this value. By surpassing the main processing unit and directly connecting to the Arduino the processing time of a command is reduced and can thus respond quicker in an emergency situation. However useful the E-stops may be they rely on a human to be observant and know when to "pull the plug". In order to more effectively control the vehicle, and to ensure minimal impacts to other vehicles other sensors about the robot are utilised. The camera and the methods of programming allow for an interesting by product. In the case where the vehicle is following another the closer it gets to the rear of said vehicle the more of the screen it takes up. As this will reduce the amount of road surface, and thus course lines, the certainty of direction will drop, and with it the speed. This allows for an automatic catch-up/follow system that ensures minimal contact and time disparity between competitors and the vehicle. Following this there are a number of ultrasonic sensors positioned strategically about the vehicle. Attached to the underside of the board along the front these serve the purpose of determining distances to other competitors and may help in making decisions on turning. The placement is not only beneficial from a collision avoidance point of view, allowing the vehicle to "steer clear", it does not account for any foreign objects behind it. This enables the crew and

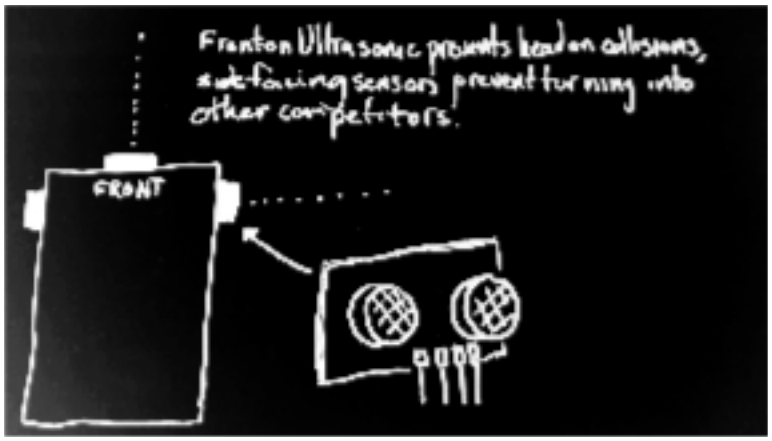


Figure 9:Diagram of Ultrasonic Sensor Placement

spectators to follow the vehicle safely around turns without triggering safety mechanisms, and only asserts that the vehicle is not at fault for

any accidents. Notwithstanding in the event of a collision foam padding lines the vehicle to dampen shocks and damages. Through the combination of onboard and manually controlled safety mechanisms Fenrir ensures that the highest degree of safety is maintained throughout its operation.

VI. MATERIALS

Item	Price
Traxxas Slash	$300
Traxxas Nickel-hydride battery	$80
Insignia USB Hub	$20
Arduino UNO	$15
Beaglebone Black	$80
Emergency stop button	$6
Powerbank	$80
Wiring and Tools	$20
Total	$601

Fig 10: Cost of materials

Through using exclusively consumer ready products Viking Robotics aims to show the world that automation does not require the same complexities in development that is so often assumed.

VII. Conclusion

Viking Robotic's participation in this years IARRC has been an exhilarating experience. From learning the ins and outs of vehicle automation, to working with experts on force propagation, image processing and more it has allowed for growth of the team as a whole. With many young and inexperienced members it was often times challenging. Yet from this apparent weakness came many wonderful and innovative ideas, methods of manufacturing, novel expertises and a passion for the work unbridled. Fenrir is the result of years working through trials and tribulations until finally we are confident in the competitiveness of this vehicle. Through development of new algorithms and hardware it has taken shape, and it is our sincere hope that through the efforts of the Robotics Team that other youths may come to realize their potential in the technical fields and find inspiration from their seniors as happened to us in the past.

Acknowledgment

Viking Robotics acknowledges the support and guidance of our mentors,

Dr. Michael Burns
Mr. Carlo Fusco
Mrs. Jill Harris
Mr. Arnold Henkel
Mr. Douglas Peterman

with special consideration to,

Mr. Ion Damian

as well as,

Tianyu Guo
Andrew Ilyas
and...
Yota Ohashi

for introducing us to the wonderful world of robotics.