Trinity College

SQUIRREL

Jin Pyo Jeon (jinpyo.jeon@trincoll.edu)

Basileal Imana (basileal.imana@trincoll.edu)

Barok Imana (barok.imana@trincoll.edu)

Bemnet Demere (bemnet.demere@trincoll.edu)

Submitted: 5/19/2017

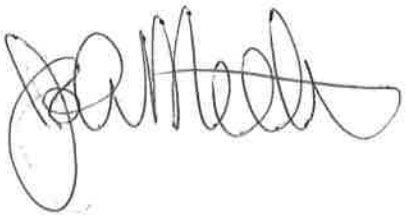
May 15, 2017

To: IGVC
From: Professor John Mertens, Trinity College

Dear IGVC,

The Trinity entry to IGVC this year is "Squirrel", a hybrid design using components from a new Trinity vehicle designed and built from scratch last year (named "E.A.R.L."), and an old mothballed robot from year's past ("Q"). The mechanical design of E.A.R.L. was flawed, and the imaging and navigation system was an advance, inventive, and imaginative design, but not fully realized. A team of Trinity College engineering students this year are hopefully incorporating a robust mechanical system from Q and completing the imaging and navigation system of E.A.R.L. to produce a successful Squirrel. We look forward to seeing you in June!

Best wishes,



Dr. John D. Mertens
Chair and Professor of Engineering
Trinity College
300 Summit St.
Hartford, CT 06106
(860) 297-2301

1. Introduction

This report outlines developments on the Trinity College Robot Study team's first iteration of SQUIRREL. It was built by combining the mechanical base and power supply of Trinity's veteran robot Q and the brain and sensors of the robot EARL which participated in IGVC in 2016. The team consists of seven undergraduate students, five studying Computer Science and two pursuing Engineering.

2. Innovations

This year, the robotics team's effort had largely been driven by two constraints: a lack of engineering students involved with the project and the engineering work required to make the base of team's previous robot (EARL) function. With these constraints, the team have chosen to integrate the software of our previous robot to the mechanical base of the robot called Q, with which we had competed in the years past. This required that the team interface the base of the robot with ROS, the main framework that the team are utilizing for the robot.

Primarily, the team developed a software driver that integrates the motor controller to ROS, enabling the retention of all the parts of robot not related to the motor control, such as sensors and algorithms. The driver was implemented with several core requirements in mind, such as response rate and allowance of both autonomous and manual control.

2. Design Process

a. Team Organization

The team consisted of four core members and several underclassmen organized into small groups based on their expertise and interest. Due to the decision that was made to take parts of the robots Q and EARL to build the new robot SQUIRREL, one group was dedicated to designing a way to utilize the existing power supply of Q to power the components that were previously used in EARL. Another group focused on improving the performance and accuracy of the navigation algorithm that was used in EARL in 2015. Each group was charged with solving a problem at and reporting with the team on a weekly basis, during which brainstorming and major decision making took place by the entire team. The entire team met for a minimum of three hours per week to work as a group and also held another mid-week meeting with a faculty advisor to assess the team's progress.

b. Design Methodology

The process started with a failure analysis of the performance of EARL, Trinity's robot which participated in IGVC in 2015. After understanding the faults of EARL, a detailed list of requirements was created. Based on these requirements, strategies were proposed and continually tested to reach a finalized implementation.

Figure 1. Design Methodology

3. Hardware

a. Chassis

The physical platform of SQUIRREL is a modified PerMobil Trax all-terrain wheelchair. This frame can support a payload of over 250 lbs [1] and has a small footprint of 40" by 26". It features a differential front wheel drive system – a pair of 500W Leroy Somer MBT1141S motors - and a pair of rear mounted casters. The motors are geared with a 25.8:1 ratio, providing 15ft-lb of torque. Additional sensor and payload mounting frames were constructed using 80/20 extruded aluminum channels. The use of these channels allowed for quick and easy component layout without compromising mechanical strength.

To avoid driving over lines, SQUIRREL's structure features a newly minimized turning radius of 17 inches. The GPS is centralized so that GPS readings are representative of SQUIRREL's true position. The payload rests on the front face of SQUIRREL for easy access, while the battery sits inside the robot. Electronic components can be easily accessed using a tray table that slides out of the back of the robot.

b. Control Panel

The hardware control panel allows easy control of the robot without the need of a software interface. The user initiates navigation by turning the power on, then turning the initialization, motor control (MC) and safety switches on. After pressing the GO button, the robot begins executing the navigation code.

The control panel also gives detailed dynamic feedback of SQUIRREL's operation. An array of LEDs provides the user with real-time status of the various sensors and the control program. A voltmeter also gives important battery life data. Finally, the emergency stop button sits in the center to terminate all processes.

c. Power Supply

The power system runs all electrical components (other than the motor) through a single power board. The new power board features separate DC-DC convertors for 24V, 12V and 5V power supply. They are rated at 100W, 75W, and 75W respectively. In addition, Switchlock circular connectors were added for all major components to allow easy removal and reconnection.

To secure sensitive electrical components from transients, high frequency feedback, and surges from the motors, a 10A Radius Power Filter model RP220-10-4.7 was added to the motor line. In addition to the filter, slow-blow fuses have been added for all major electrical components to secure them in the case of accidental short-circuits.

d. Processors and Sensors

A Dell Precision m4600 Laptop is used as the central machine. It is interfaced with two NVIDIA Jetson TK1 boards which handle all the image processing for the lane detection. The images are acquired using two PS3 eye cameras, processed and stitched together and sent over a WIFI link to the dell computer where the navigation software resides. Obstacle detection is performed using an RPLIDAR placed at the front end of the chassis. A Crescent A100 Differential GPS, with accuracy of 60 cm is used for positioning. The data from all sensors is processed on the dell computer to layout an optimum path. The path is executed via an AX3500 motor controller which controls the two DC motors.

4. Software

a. Overview

The process for autonomous navigation implemented a strategy that utilized a reactive model where decisions are made based on the current state of the robot and its surroundings. The process can be broken down into three stages: building a map and localizing the robot within that map, planning the best path to reach the destination according to that map, and converting the map to motor controls.

Data from the vision system is fused with the laser scanner to create the surroundings of the robot, and the odometry data calculated from wheel rotation is used to determine the relative position of the robot with respect to its surroundings. Once the map is built, a best path is generated from the robot's local position within the map, as well as global position with respect to the destination. Cost values are assigned to the lanes and obstacles such that the robot will avoid both in the planned path. Once a path is designated, it is then converted into motor controls. The speed and direction of each motor is calculated by the polar coordinates of the desired path vector.

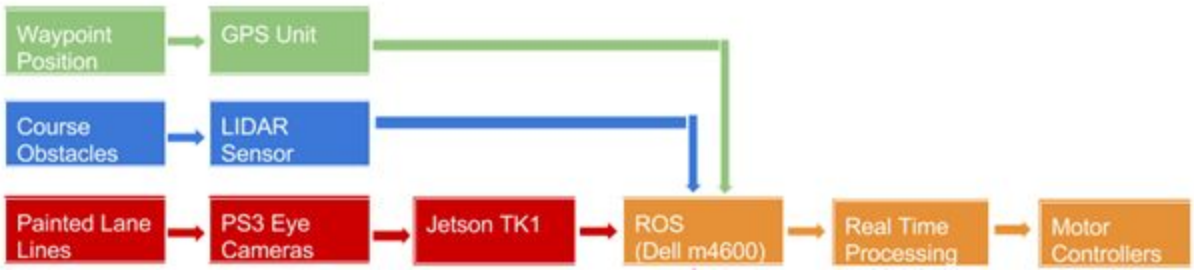


Figure: Software Architecture

b. Image Processing: Lane Detection

Image acquisition and processing is the first step in avoiding the lanes of the IGVC course. The vision system of the robot consisted of a dual camera (two PS3 Eye Toy webcams) in which

each image captures one half of the course, creating a panoramic view of the course. The line information would then be sent to the host PC to then be included in the generation of the map.

The choice to utilize two cameras was made since it created the widest field of vision possible without overly sacrificing depth of view. The team decided that it was most important to have a wide field of vision so that the robot could see the entirety of the designated 10 foot wide lane, as close as 1.5 feet with respect to the front of the robot.

Each camera captures video a resolution of 160x120 pixels. Lowering the resolution was a strategic decision to increase framerate. It was determined that for what the robot needed to see, this resolution was sufficient, as lines on grass did not call for a high resolution frame. The designed algorithm for detecting the line combines six steps of manipulation that consist of source acquisition, obstacle removal, mixed channel masking, binary thresholding, edge detection, and finally calculating a best fit line. Shown in the figure below is an example source image captured from a previous IGVC course.

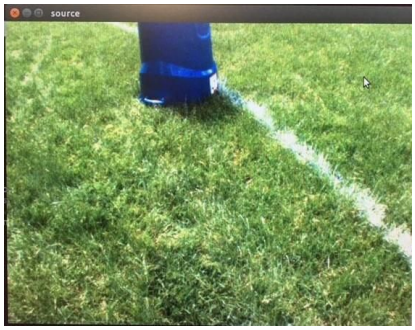


Figure: Example Source Image

The first step of manipulation involved removing any unwanted features of the source image. Specifically, this refers to obstacles (barrels, sawhorses, etc.). Since there is a separate subsystem to handle obstacle detection (explained later in this report), the robot's vision would completely ignore any obstacles and only focus on detecting the lanes. When designing the obstacle removal, it was known that the obstacles on the IGVC course are all either blue, yellow, or orange. Removing these colors from the image would, in turn, remove the barrel from the image.

The first step in doing this would be identifying all pixels in the source in range of published values of orange, yellow, and blue. The published values corresponded to the R, G, and B values of the pixel. Every frame was captured as a three channel, 8bit matrix (an RGB image with RGB values ranging from 0 to 255). If a pixel had its R, G, and B values all in range of the set parameters, it was identified, otherwise it was ignored. Identifying all pixels within the barrel created a mask that contained only the barrel. The mask was then subtracted from the source frame using bitwise subtraction (each pixel's R, G, and B value were subtracted). This left the pixels not corresponding to the ones in the barrel as is (the unidentified pixels in the mask has

RGB values (0,0,0)) whereas the images identified to be in the barrel were removed (the pixel was essentially subtracted from itself resulting in a final pixel value (0,0,0)). The process results in a new frame where the obstacle is removed from the source frame. Shown in the figure below is an example of the described process.

Figure:. Barrel Removal Example

Once the barrel is removed, the only data left in the image is the line and grass (and possible noise resulting from leaves). The next step is to create as much contrast between the line and grass as possible. The team's first notion was to perform a threshold, to create a single channel grayscale frame, where the line and grass were represented as binary values 0 and 1. Upon initial experimenting, this did not produce reliable or acceptable results. Upon further research it wa determined that the inaccuracies were due to pixel values having similar RGB values, despite looking different to the human eye. To compensate for this problem, a contrasting mask would create enough difference in pixel values to the threshold. A technique for accomplishing this was planar extraction. This entails converting the image to a single channel intensity matrix, where the pixel was represented as an 8 bit intensity value where the intensity corresponded to the specified plane.

Literature and testing showed that extracting the blue plane provided the best contrast between the grass and line.This produced fair results but in further testing, the removal of the green plane provided even better results. This was because the grass (without the line) is the most green part of the picture, so removing the green plane would drastically lower the intensity values of the grassy portions of the image. This mixed channel manipulation provided a reliable and usable contrasted frame. The process is visually represented in the figure below.

Figure: Mixed Channel Contrasting

With a more contrasted image, a thresholding could now be performed. The purpose of thresholding the image is to create a single channel image where all pixels are identified as either value 0 or value 1. This process is necessary in performing the further steps in the process. To accomplish this, a threshold intensity value is set and any pixel with a value higher than that intensity is set to 1, otherwise it is set to 0. The process is visually represented in the figure below.

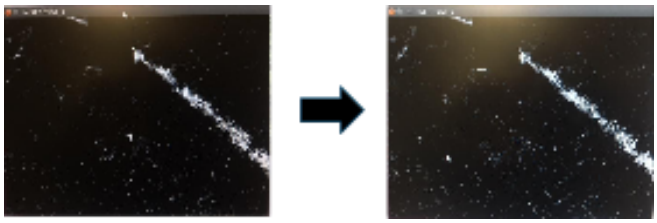


Figure: Binary Thresholding

With a single channel, binary image, the line extraction can occur in two step process. The first is to perform an edge detection. An edge detection is a process in which only pixels determined to be part of an edge are identified, otherwise the value is set to zero. The process begins by first performing a blurring filter to the image. This is because the edge detection is subject to be affected by salt and pepper noise within the image, so blurring with image will decrease the strength of the edges created by unwanted noise, whereas the true edges will be affected, but still identifiable by the algorithm.

After blurring, each pixel is compared to all of the pixels in its corresponding kernel (a definite nxn matrix containing adjacent pixels n/2 to the left, n/2 to the right, n/2 above, and n/2 below) and according to a predetermined set of rules will determined if the pixel belongs to an edge or not. This process involves comparing intensity values along the northsouth, eastwest, northeastsouthwest, northwestsoutheast axis and only identifying pixels in which intensity values indicate an edge. Once edge pixels are identified, the image is rethresheld to leave only the identified edge pixels. The process is visually represented in the figure below.

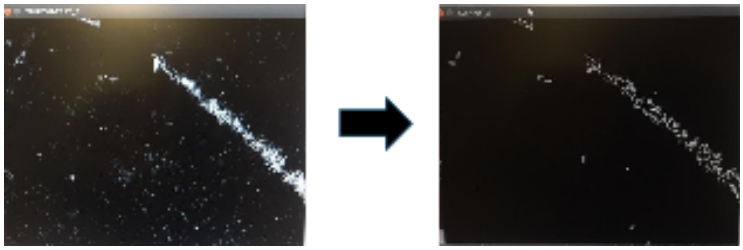


Figure: Edge Detection

With the edges now identified, the clusters of pixels still identified correspond to the pixels on the lane. The final step is to identify the pixels that belong to the line, by performing Hough Probabilistic Transform. The transform requires a single channel grayscale image and detects

features of that image according to a set of hard coded parameters. The transform scans for linear features that are made up of minimum number of n pixels, with a maximum allowed gap of m pixels (m and n are identified before running algorithm). This means that only features containing n pixels are detected, with features separated by a gap of m pixels or less are considered to be part of the same feature. The transform returns the two endpoints of the vector that make up the linear feature. The process is visually represented below in the figure below.

Figure: Hough Probabilistic Transform

The entire process from start to finish can be seen in the figure below.

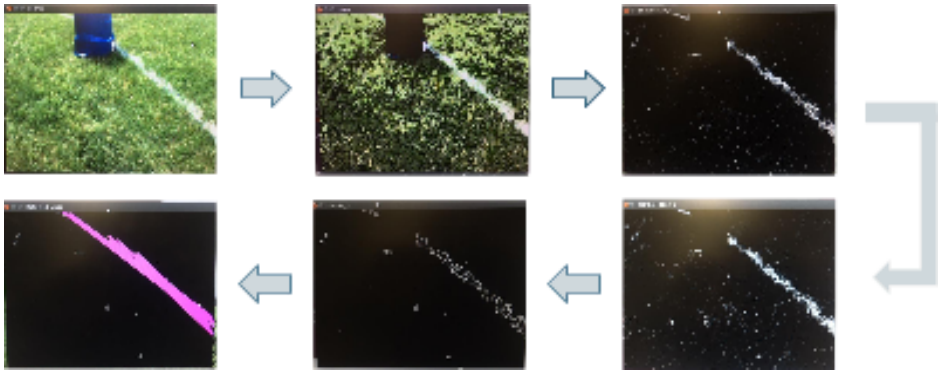
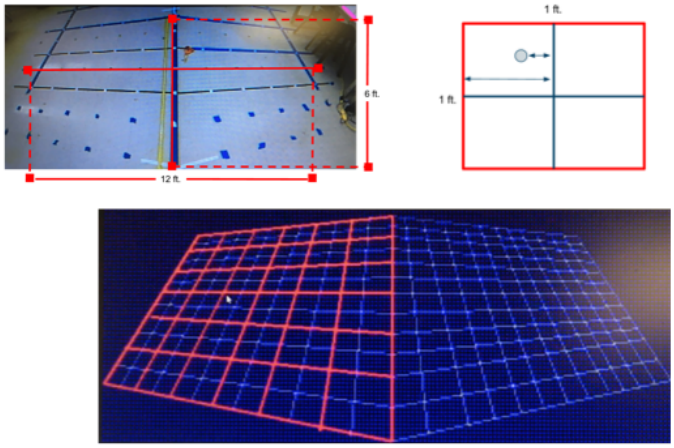


Figure: Six Step Manipulation Process

With each Jetson development board sending two halves of a processes panoramic view, with detected lanes, the images would then be stitched together locally and the final frame would be incorporated into the map.

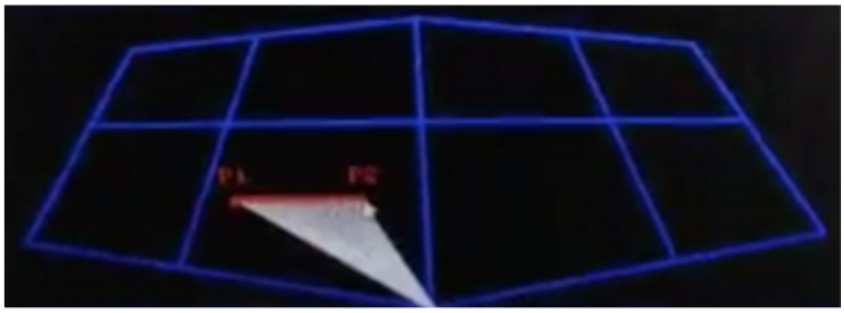
After processing, the line information containing coordinates of the endpoints of the lines segments that make of the entire lane are sent to the computer. These points allow the robot to calculate all pixels included on the specified line segment. These pixels are then mapped to real world (x,y) coordinates in relation to the robot. This is done by creating a vision mask that segments the vision into 1ft.x1ft. regions, and calculates the distance values by creating a ration of pixel displacement from known pixel to real world displacement distance.



Segmenting Region For Pixel Mapping

Figure: Pixel Mapping

When all points are mapped to real world distances, the coordinates are converted from cartesian to polar coordinates to generate a 2D Point Cloud that represents the line. This is read by the navigation algorithm as a laser scan, since this is the necessary data type for the map building algorithm implemented. The generated laser scan from the vision data is fused with the LIDAR scan programmatically to create one final laser scan that incorporates line data and obstacle data. This new laser scan represents the entirety of the surroundings of the robot within the 12 ft. (width of vision) by 6 ft. (depth from front of robot) that can be used to generate a map.



Generating Laser Scan On Vision Mask

c. Obstacle Detection

It was decided that the sensor that would perform the obstacle detection be a light detecting a ranging (LIDAR) sensor. See figure 28 below for image of a LIDAR sensor.

A LIDAR sensor is a 360 degree radial scanner that scans at a predetermined scan rate and generates a usable list, made up of range values with respect to the scanner, indexed by corresponding scan angles. The generated message is the same as the message described in the fake laser scan method of this report, as the fake scan message was implemented to mimic this laser message.

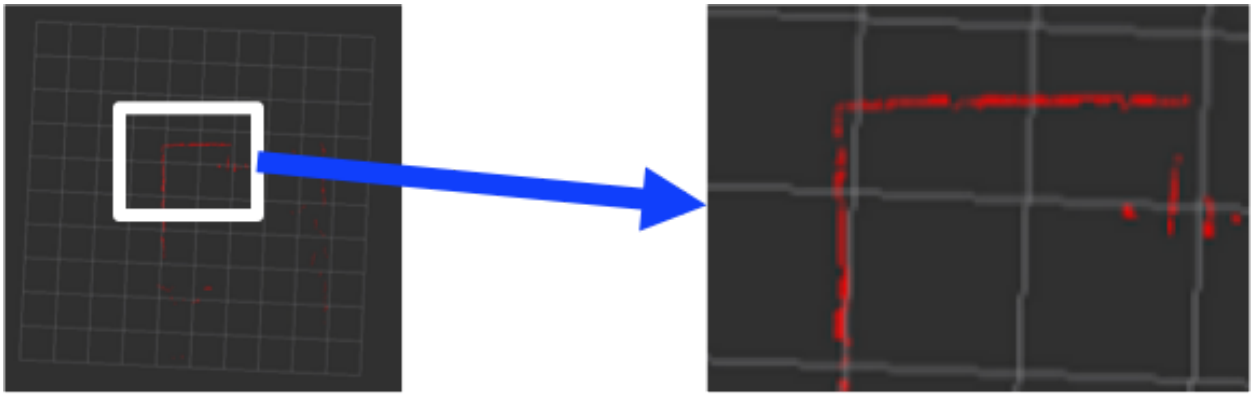


Fig 12. Plot Generated By LIDAR Scan

The scan fused together with the generated scan (method described in previous section) creates a comprehensive point cloud that represents the entirety of the surroundings of the robot within the 12 ft. (width of vision) by 6 ft. (depth from front of robot) that can be used to generate a map.

d. System Integration

To ensure smooth integration of various components, we utilized several methods in the design of our robot. First of all, the software that deals with each of the components are isolated into distinct parts called ROS nodes. This separation allows the components to be tested for bugs and capabilities separately. Furthermore, this distributed nature of the robot allows for the robot to undergo recovery procedure if one of the components were to fail.

5. Safety

Safety has been given the utmost priority in the design of SQUIRREL both for the electrical system and the mechanical system. Wires of gauge 10 were used to connect power sources to the motors and filter, and 16 gauge wires were used elsewhere. A circuit breaker was used for the entire electrical system. Slow-blow fuses were instantiated into the connections to each component from the power board to ensure that individual electronic devices did not receive too much power. A single phase dual stage power line filter was inserted to prevent transient current from the motors.

Three main motor safety measures have been implemented. They include the motor control board FETs, the physical emergency-stop, and the wireless emergency-stop. The motor control board FETs are now controlled by the program to disconnect power from the motor whenever the system is in an idle state. The physical and wireless e-stop features also work by immediately cutting power from the motor controllers as soon as one of them is triggered.

The wireless Emergency Stop is controlled by a Seco-Larm SK-919TD1S-UP transmitter that uses 315 MHz RF transmission with a one-channel receiver. The range was extended to 150 feet by installing antennae on SQUIRREL and on the transmitter.

6. Budget

Component List	Price (USD)	Cost Incurred (USD)
24V battery	$100	$0
Crescent A101 DGPS	$1500	$0
Caster Assembly	$150	$0
Chasis	$650	$0
Motors	$1000	$0
Power Supply Board	$150	$0
Remote Control	$180	$0
Roboteq AX3500 Motor Controller	$400	$0
Wiring/Electrical	$50	$0
RPlidar	$400	$400
PS3 Eye Toy Cameras	$12	$12
Dell Precision m4600 Laptop	$480	$480
Tegra TK1 Development Boards	$384	$384

7. Sponsors

Hemisphere

Trinity College Engineering Department

Trinity College S.G.A.