Q

2015 IGVC Design Report

Team Members:

Binod Giri ’15, Benjamin Williams ’15, Philip Cho ’15,
Barok Imana ’16, Romero Board ’16, Basileal Imana ‘17

Faculty Advisor:

Dr. David J. Ahlgren

May 11, 2015

Trinity College
HARTFORD CONNECTICUT

Table of Contents

Trinity College
HARTFORD CONNECTICUT

May 11, 2015

Faculty Statement for Trinity College IGVC Entry

This is to certify that the Trinity College IGVC entry "Q" has undergone significant redesign in both hardware and software from last year's IGVC entry. The Q team members worked on the robot as an Independent Study project and received 0.5 credit (1.5 credit hours) for each of the two semesters in academic year 2014-2015. This project is significant, and has produced offshoot senior design projects for next year (2015-2016) in both Computer Science and Engineering.

Sincerely,

[illegible]

Dr. John D. Mertens
Chair and Professor of Engineering
Director of the Trinity College Robot Team
Trinity College
300 Summit St.
Hartford, CT 06106

DEPARTMENT OF ENGINEERING

300 SUMMIT STREET, HARTFORD, CT 06106-3100
TEL (860) 297-2517 FAX (860) 297-2531 www.trincoll.edu

1. Introduction

This report outlines developments on the Trinity College Robot Study team's ninth iteration of Q, an autonomous ground vehicle designed for competition at IGVC. The team consists of six undergraduate students, two studying Computer Science and four pursuing Engineering. Great focus has been taken on developing reliable GPS navigation. Additionally, the team continues to develop an effective vision system, create a more dynamic motor control system, and adhere closely to the guidelines for the Interoperability Profile (IOP) Challenge.

2. Innovations

In IGVC 2014, Q benefitted from a mechanical redesign and newly developed vision system. The focus of innovation this year was a reassessment of Q's GPS navigation system, which has operated poorly in previous competitions. Motor control has also operated inefficiently, resulting in jagged and abrupt movements from Q. This year Q will navigate more smoothly through the course using a more developed automatic control system.

Q's vision system, a continuous project, will also be heavily revised from previous years. While a Hough transform was effective last year to detect white lines on the ground, the introduction of white potholes would greatly hinder that algorithms performance. Measures to revise the vision system are under development.

Finally, we upgraded our JAUS component to support additional messages as required by this year's Interoperability Challenge.

a. GPS Navigation

In recent years GPS navigation has been functional but far from optimal. The primary fault in last year's competition was that Q would appear to aim for a GPS coordinate far away from the actual GPS coordinate given by the competition, resulting in Q driving off the course. This year the team ran full diagnostics of the GPS navigation system, determining that the GPS and compass (purchased in 2007) were responsible for the inaccuracies. This year the team has implemented a new GPS and compass to navigate the course.

b. Vision System

The principal purpose of Q's vision system is to detect the white lines that Q must remain inside during the competition. As such, extraneous information such as blue and white barrels on the advanced course and the new white potholes on both courses require redevelopment of Q's vision system. The previous system was specifically designed for detecting only lines, but the team is now developing a system that will identify and remove a pothole from the image being processed. The previous algorithm will then handle the image consisting only of the lane, while a new algorithm will process the white hole. Finally, Q will similarly remove blue and white barrels from its images, processing the barrels using a distance sensor in order to avoid them.

c. Motor Control System

Last year Q’s motor control system was revisited by developing encoders to verify Q’s speed during operation, which was especially critical to the IOP Challenge. However, Q lacked an effective motor control system when making adjustments to its path, resulting in undesired, jagged behavior. A more developed control system to smoothly control Q’s path is in progress.

d. IOP Challenge

Q has performed very well in the IOP challenge in recent years, placing second last year. One issue last year was the receiving of messages from two transmitters simultaneously. This year we can account for multiple transmitters. Additionally, we have built a tester program to simulate the transmission of messages from the judge easily and efficiently. This allows for quick verification of the implementation.

3. Design Process

a. Team Organization

Given the core of the team was three senior members, the 2014 fall semester mostly consisted of imparting information to underclassmen who were not experienced with Q. Members worked together on revitalizing the GPS navigation system during this time. By spring 2015, tasks were divided into the following:

1. Leadership: one senior member who leads the entire Trinity College Robotics Team and one senior member who leads the Q team. These members led weekly meetings and determined design focus and individual responsibilities for the robot.

2. Vision System: one senior and one underclassman tasked with developing vision software.

3. Navigation: one senior and one underclassman responsible for improving the motor control system and integrating navigation algorithm with vision system.

4. IOP: one senior and one underclassman responsible for the development and testing of the IOP implementation on Q.

Members were divided into these groups based on their expertise, interests and workload associated with each group. Most members contributed to more than just one group. All members were required to work together for a minimum of four hours per week as well as attend weekly meetings to give progress reports and discuss their projects.

Since Q is a multi-year project, key members graduate every year and knowledge that was once common to all members is lost. To combat the high turnover rate, each new member works with an experienced mentor on the team who passes on his or her expert knowledge. Furthermore, regular workshops were held on LabVIEW coding to help new members master Q’s programming language.

As Q is nearing its tenth year, much of the spring semester was devoted to developing a new organization of six juniors interested in pursuing a new robot for IGVC next year. The Q team has passed on substantial knowledge so that these juniors can design an entirely new robot for IGVC for their senior design projects next year.

b. Design Methodology

The team followed an iterative design process to improve Q. The process started with a detailed failure analysis of Q's performance in the IGVC 2014. After understanding the faults of Q and analyzing IGVC's new rules, a detailed list of requirements was created. Based on these requirements, strategies were proposed and continually tested to reach a finalized implementation.

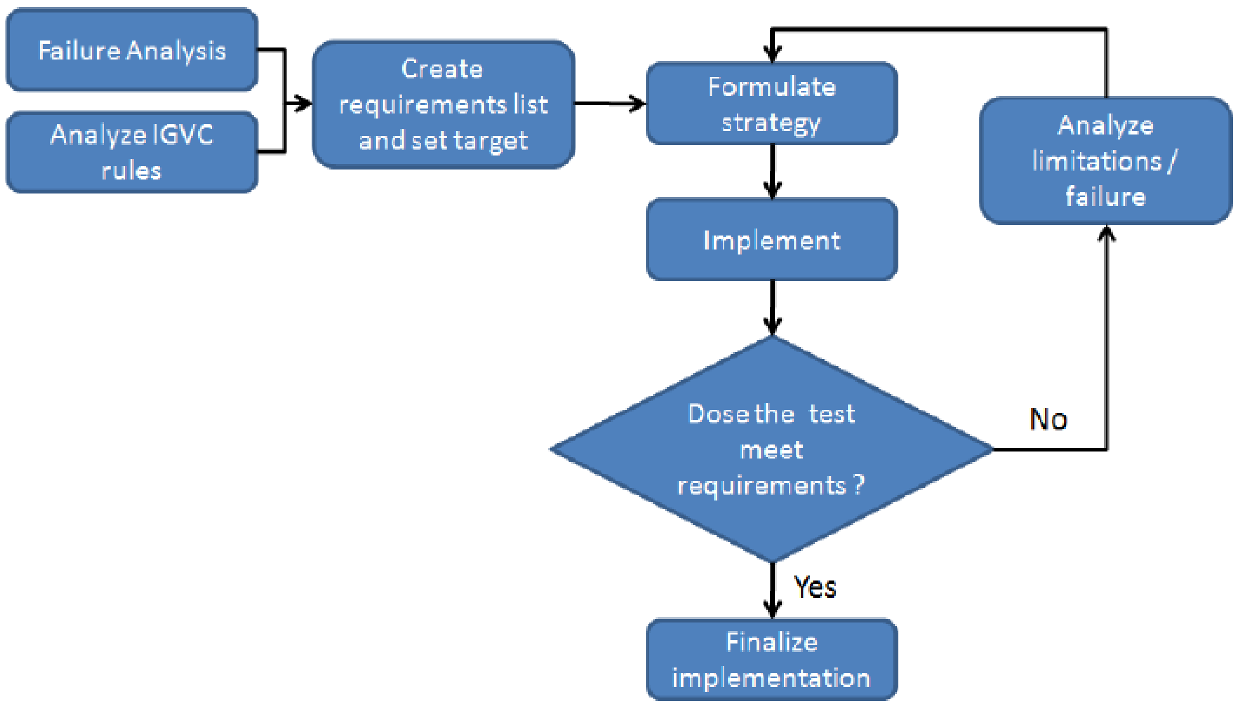
Failure Analysis
Analyze IGVC rules
Create requirements list and set target
Formulate strategy
Implement
Dose the test meet requirements ?
No
Analyze limitations / failure
Yes
Finalize implementation

Figure 1: Design Methodology

4. Hardware

a. Chassis and Drive Train

The physical platform of Q 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, Q's structure features a newly minimized turning radius of 17 inches. The GPS is centrallized so that GPS readings are representative of Q's true position. The payload rests on the front face of Q 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.

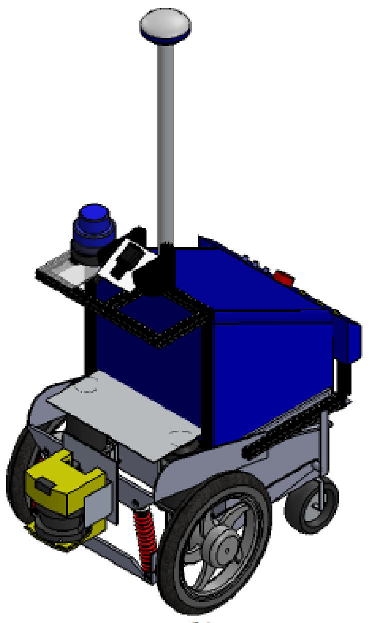

Figure 2: SolidWorks Model of Q

b. System Integration

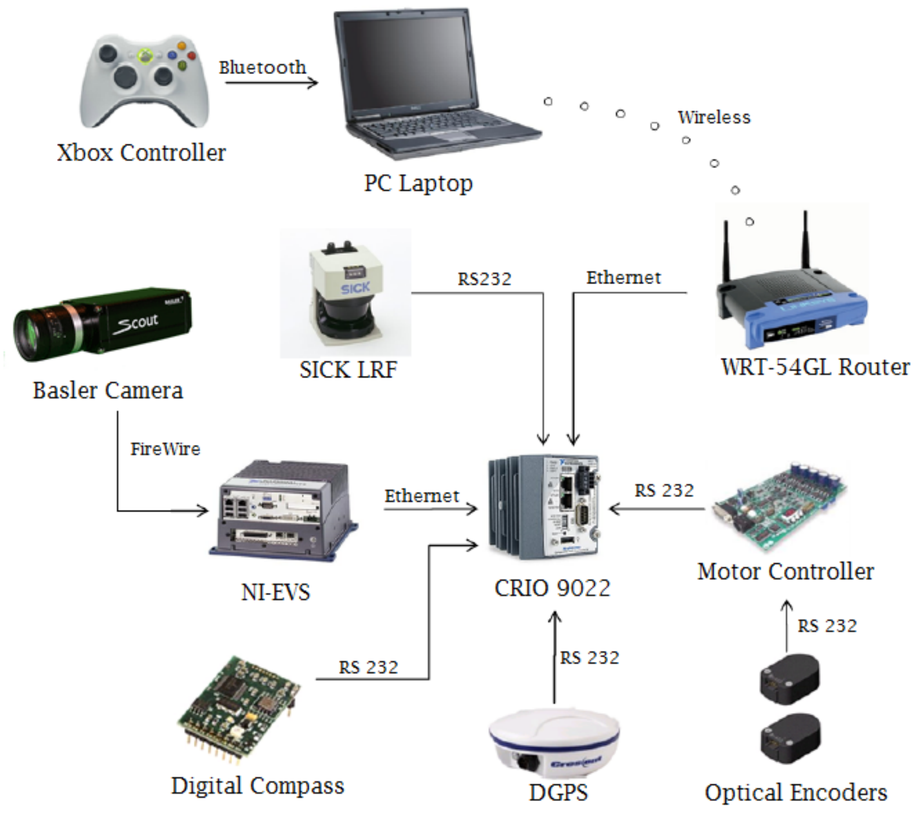

Figure 3: System Overview of Q

As shown in Fig. 3, the National Instruments Compact Reconfigurable I/O (cRIO) 9022 is the main processing unit of Q. All sensors send information to cRIO, which makes navigation decisions based on sensor data. Q uses a SICK laser range finder that provides a 180° sweep in front of the robot and generates a polar histogram of the distance to any obstacles within 80 meters at 1° intervals. A Honeywell HMR 3300 Digital Compass also provides roll, pitch, and yaw data. A Crescent A100 Differential GPS, with accuracy within 60 cm, provides Q with GPS coordinates. Additionally, E5 encoders embedded into the DC motors record speed data for JAUS applications. All sensors link to cRIO, which collects and processes data using a modified Vector Field Histogram (VFH) path planning algorithm. The algorithm determines speed and turn commands, which are delivered to the AX3500 motor controller to move the robot.

The National Instruments Embedded Vision System (EVS) 1464 performs image processing separately before reporting to cRIO. EVS acquires images via a Basler Scout Camera with a wide angle lens, which has a range between 0.3 and 3.5 m in front of Q. This vision system is used to keep track of the white lines on the grass. Once cRIO receives information about the lines from EVS in the form of polar histogram, it superimposes this information with the SICK data, which is then processed by the navigation algorithm.

cRIO is also connected to a Linksys Wifi Router, which allows cRIO to poll sensor information to a remote laptop using UDP packets. When not operating autonomously, an Xbox controller is connected via Bluetooth to the laptop, allowing users to relay motor commands to cRIO wirelessly and manually control Q.

c. 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.

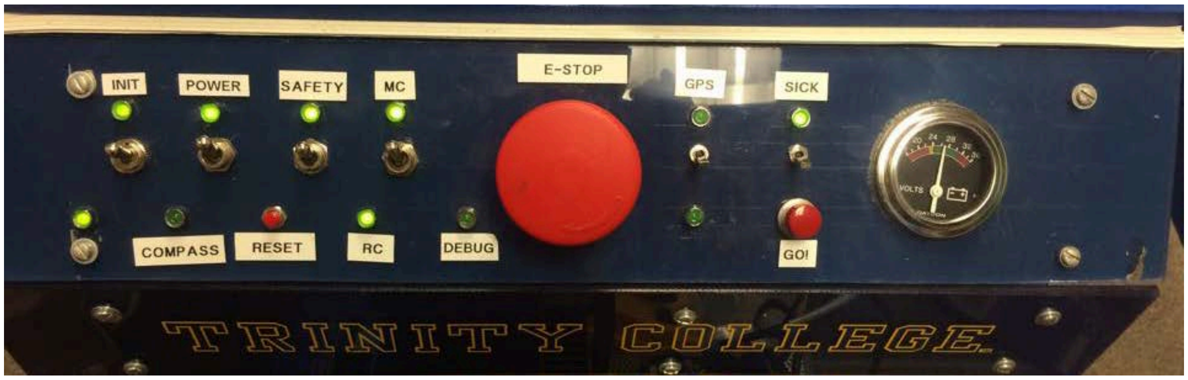

Figure 4: Q's Control Panel, located in rear of robot

The control panel also gives detailed dynamic feedback of Q'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.

d. 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.

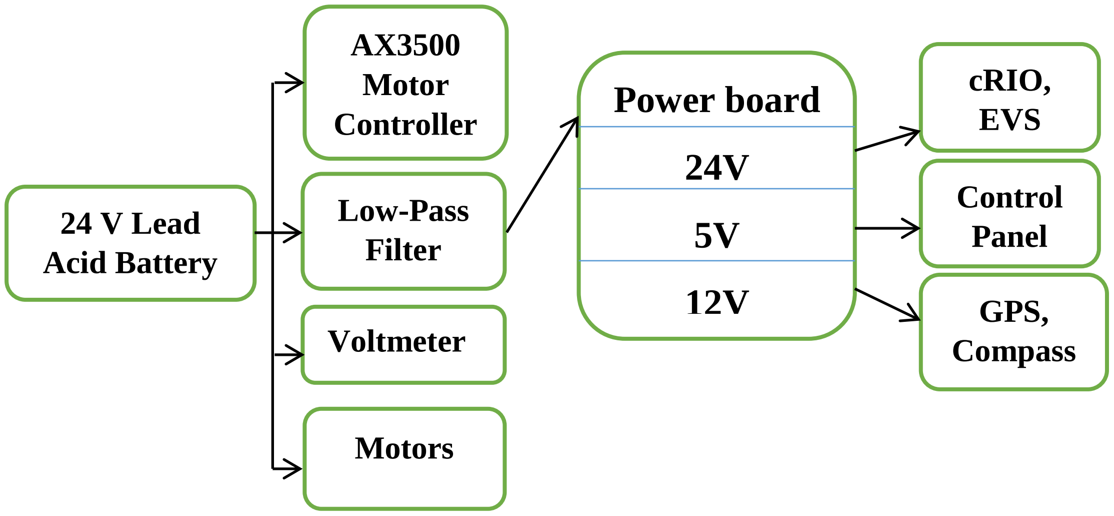

Figure 5: Power Distribution on Q

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.

5. Software

Software is written in LabVIEW and uploaded to cRIO. This provides a parallel processing design paradigm, which leads to high processing efficiency.

a. Software Architecture

The core of Q's control system is located in the cRIO and interfaces with most of the sensors and devices on the Q, including the motor controllers. All movements of the robot, autonomous or remote-controlled, are directed through this main software architecture, which is shown in figure 6.

Our goal in designing the main software architecture was not only to create a system with maximum performance and reliability, but also to make our software as intuitive and as modular as possible. To that end, our design is based on a parallel architecture, where each component runs relatively independent of other components. Our implementation of this architecture uses parallel loops, each containing code pertaining to one particular component of the robot. For instance, each of the sensors interfaced with the cRIO and the motor controllers is controlled by a separate loop. The sensor loops continuously update data buffers, while the motor controller loop waits for active commands from the main control loop. This is to ensure that there is absolutely no undesired movement of the robot.

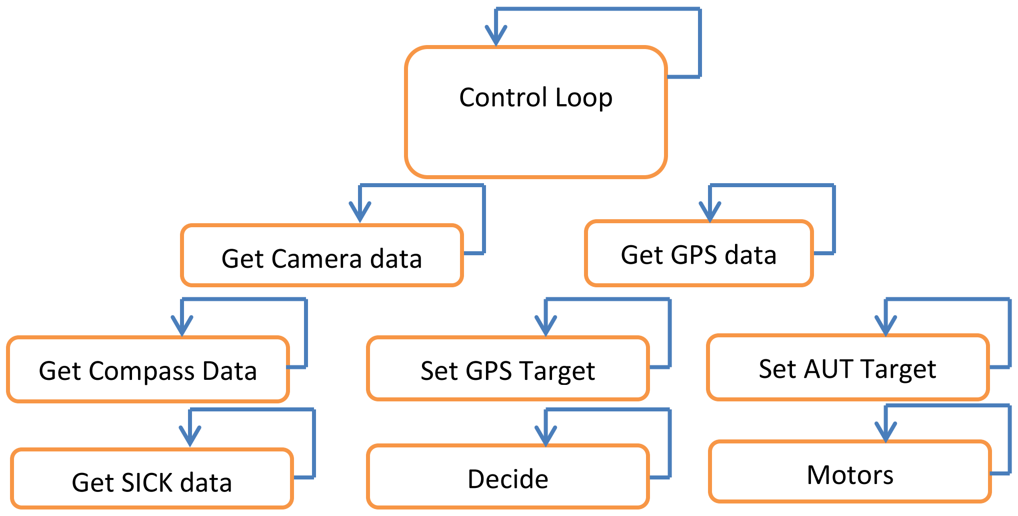

Figure 6: Parallel Software Architecture on Q

The main control loop directs data flow between loops, thereby controlling the behavior of the system. For example, when in the "remote control" state, motor commands received from the tablet PC are sent to the motor controller loop, and while in the "autonomous" state, the motor commands are computed by the VFH loop. Communication with the tablet PC and the Embedded Vision System are again achieved by using separate UDP loops.

The parallel architecture has several advantages, for instance, it allows the overall throughput of the system to be much higher, allowing for smoother control of the robot. Furthermore, the new architecture has proved to be much more intuitive than its predecessor, since the overall structure is much more natural to understand.

Having parallel loops has also allowed us to modularize the code and work on separate parts individually, increasing efficiency and reducing the amount of time spent on piecing the various sections of code together.

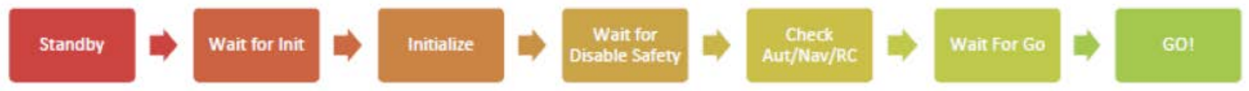

Figure 7: Control Loop State Machine

b. Software Interfacing

The camera is connected directly to the EVS's FireWire ports. LabVIEW provides library functions for capturing images from the camera. The GPS, compass, and SICK LIDAR interface with the cRIO through the FPGA. Each of them connects to one of the four ports on the cRIO RS232 module, which is directly interfaced with the FPGA.

The code to interpret data from the GPS runs on the real-time controller in the Compact RIO. The FPGA is programmed to act as a serial port. The GPS outputs longitude and latitude following the NMEA format. All latitude and longitude data are preceded by the string \$GPGGA, so the GPS interpreter searches for that string, and then parses the following string for the latitude and longitude, which are converted to double floating point values and transmitted to the Compact RIO.

To read data from the SICK LIDAR, some initialization commands must be sent. Once these commands are sent, the SICK sends 360 bytes preceded by a 6 byte header. Those 360 bytes are 180 16-bit distance values in centimeters - one value for each degree, starting at -90 degrees from the heading and going until 90 degrees from the vehicle heading. The FPGA code searches for the 6 byte header and then reads each pair of bytes into some of the dedicated memory in the FPGA.

c. Communication

There are several components in the system that must communicate for the purpose of debugging, monitoring, parallel processing, and remote control of the robot. Figure 3 shows how all the processing components in the system are connected.

The EVS and cRIO each have alternate Ethernet ports which are used for direct communication between the cRIO and the EVS. The primary Ethernet ports are used for the wired connection from the EVS and cRIO to the Linksys wireless router. The router is used for wireless debugging, software development, and monitoring of the robot. A LabVIEW application for real time debugging was created that displays the exact information Q is using to navigate, such as current GPS location, current heading, target heading, SICK array, compass reading, a waypoint checklist, battery voltage, sensor initialization, and the control loop state machine state.

d. Intelligence Algorithms: VFH+

A modified form of the Vector Field Histogram (VFH) algorithm was used for obstacle avoidance [3]. Using a SICK LIDAR sensor, a compass, and camera data, a polar histogram representation of the obstacles is created and paths are planned accordingly. The algorithm utilizes a cost function which considers the target direction, current heading, and previous direction. White lines are detected using the camera and other obstacles are detected using the SICK sensor. When there is more than one opening between obstacles, this cost function is used to select the best candidate direction. The opening with the lowest cost is chosen. The result is a heading which represents the best path. The SICK data is divided into three distance thresholds, with the nearest threshold (at one meter) receiving priority in motor functions. The further two thresholds, at two and three meters, are considered in path planning. They influence the cost calculation in the same manner as the first threshold, with robot direction and target heading taken into account, but have less weight than the threshold at one meter.

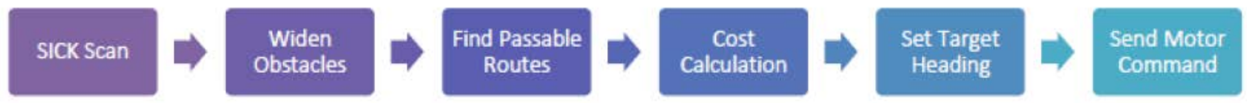

Figure 8: VFH Loop

e. Smart Path History

Q now uses a smart path history to calculate a target heading to help navigate and avoid turning around. Q uses GPS co-ordinates taken at time intervals to calculate and store past headings. A weighted average of this history gives a general heading that roughly follows the direction of the course. This assists Q in determining the general direction of the track. This is illustrated in fig. 9. However, using a simple averaged path history fails while traversing a complex chicane. While traversing a chicane, Q may be required to turn at a direction that is radically different than the general direction of the course. Using heading readings from inside the chicane to calculate an averaged path history may result in a target direction that leads Q into a dead end. Such a scenario is identified in fig. 9.

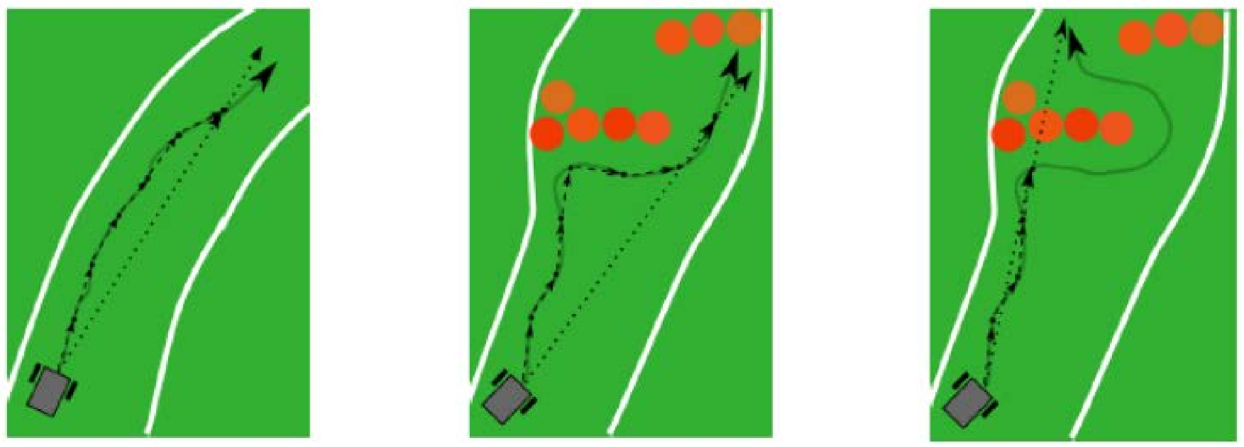

Figure 9: General path without obstacles (left), simple path history route – Q is driven into a corner (center), and environment aware path history route – Q traverses the chicane (right)

To overcome the challenge of traversing a chicane, Q uses an environment-aware path history. Q uses the concept of an obstacle field to select an alternate path. Here, an obstacle field is defined as an environment where there is a large density of obstacles such as in a chicane. When Q is not in an obstacle field it continuously logs path history to calculate a new target heading. But when Q encounters an obstacle field, data collection for the path history is temporarily suspended and the target heading calculated before entering the obstacle field is held constant. This helps Q maintain a sense of direction even while traversing a chicane. Fig. 9 illustrates how, this algorithm helps Q traverse a chicane.

f. Waypoint Navigation

Autonomous waypoint navigation is accomplished in three phases. In the first phase, Q traverses the lane and obstacles while heading towards the first GPS waypoint. The heading to the waypoint is determined using Q's current position, the GPS coordinate, and Q's compass. Q will try to head straight to the point if there are no obstacles. In the second phase, Q is in the "no-man's land" where obstacles are more sparse. Priority in the cost calculation is taken to the GPS heading to the second waypoint. Once Q finds the other side of the course and reenters the white lane, Q is in phase three and behaves as it did in phase one.

g. Image Processing

The Basler camera has been positioned for maximal viewing range with bottom of the image closest to the robot. The field of view is shown in fig. 10.

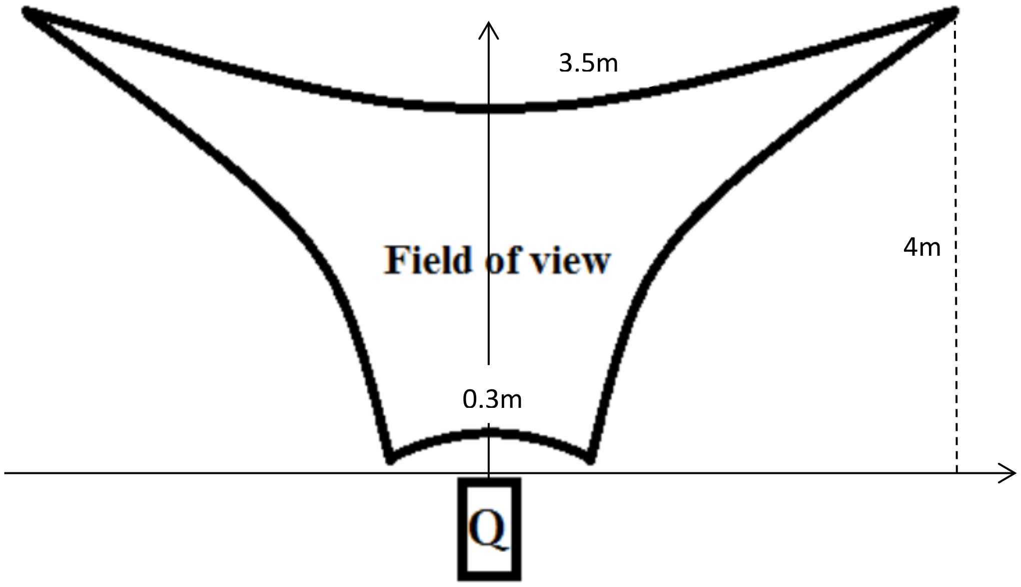

Figure 10: Q's Camera Field of View

A new vision algorithm was developed to achieve a more effective, consistent performance despite varying lighting conditions. Fig. 11 illustrates the processing algorithm for each acquired image. The original image (Fig. 11a) had a RGB color map, and because the background was mostly grass with high intensity in the green plane, a blue plane extraction eliminated most of the grass and distinguished the white lines (Fig. 11b). X1.5 lookup table and X0.5 were then applied to the blue plane consecutively to further enhance the contrast (Fig. 11c). The green plane of the original image (Fig. 11d) was then subtracted from the blue plane to eliminate noisy reflections from the grass (Fig. 11e). LabVIEW's auto threshold by metric was then applied to the image, generating the binary image (Fig. 11f). Hough Transform was then used to reconstruct the white line from the image, such that any noise left over was automatically removed (Fig. 11g).

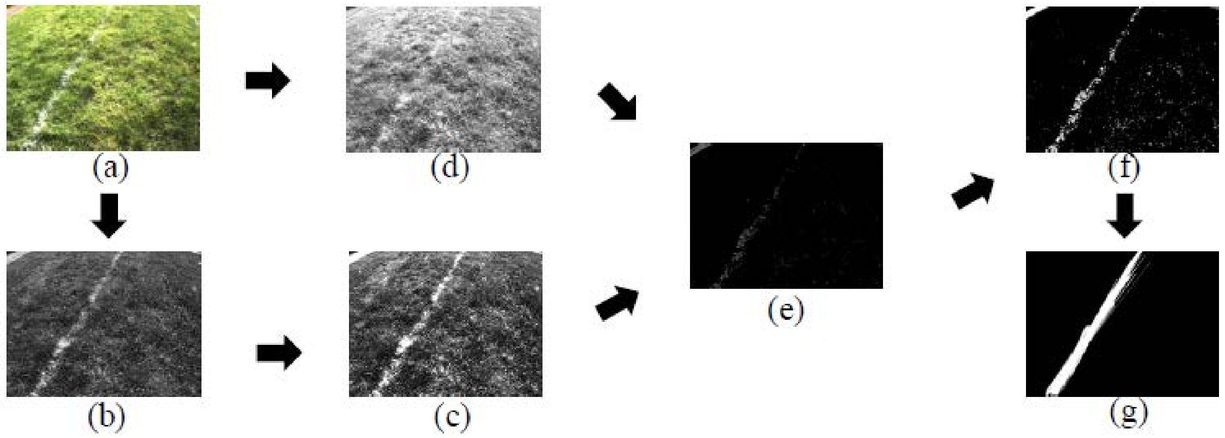

Figure 11: Illustration of the image processing algorithm employed on Q for the 2013 IGVC: (a) Original image, (b) Blue plane extraction, (c) Blue plane ^1.5, (d) Green plane extraction, (e) (c) – (d), (f) Threshold by metric, (g) Hough Transform.

The processed image then underwent a calibration procedure which analyzed the foreground object (the white line) and created a 2D occupancy plot (fig. 12d). The calibration mask (fig. 12a) contained real-world coordinates of where each white circle was relative to Q. This information is used to transform input image into a corrected image (fig. 12c). The resulting plot data (fig. 12d) is sent to cRIO to be combined with the SICK data for Vector Field Histogram.

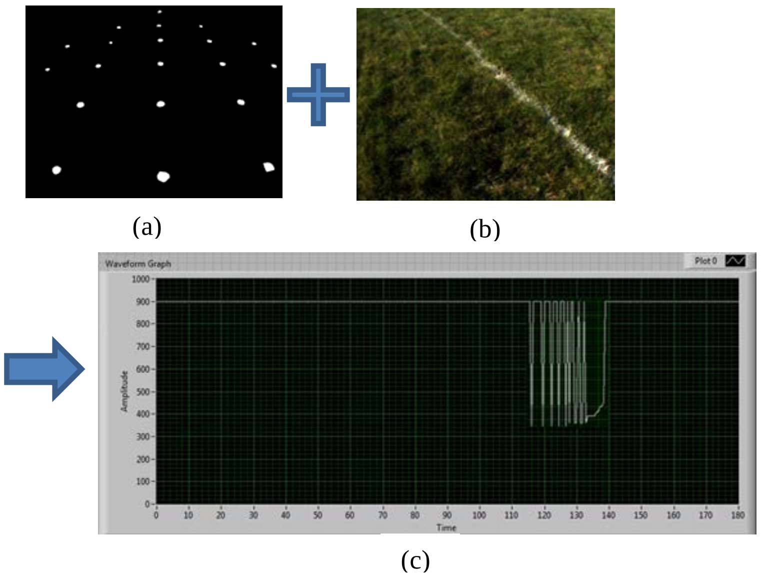

Figure 12: Image calibration process (a). calibration mask, (b). input image (hough transform image), (c). corrected image, (d). plot of obstacle distance versus angle.

The throughput of the vision system has limited the reaction time of the robot. The algorithm has undergone pipelining. The various subtasks in the algorithm now run parallel to each other thereby reducing the execution time and increasing throughput. Portion of the code has also been ported to the FPGA of the EVS, thus decreasing processing time.

In addition, new software was written to check the brightness of each input image by calculating the mean and standard deviation of the grayscale level on the blue plane. The result is then compared to a lookup table. If the image is determine to be too bright or too dark, the gain of the camera is decreased or increased respectively. Otherwise, no change is made to the gain.

Barrel Blocking Algorithm

In the advanced course, multiple colored barrels are used as obstacles. With the last year's algorithm, the problem arose when blue and white barrels were in the image; they would appear as white blobs in the color plane extraction. With these blobs, the Hough Transform performed poorly in reconstructing white lines. For IGVC 2015, we are extending the 2014 algorithm by incorporating barrel masking in the input image. This method was adapted from the line extraction algorithm used by the 2012 IGVC team from IIT Bombay [2].

In this barrel masking method, the laser range sensor is used to determine the angle and distance of the barrel, which is then translated into pixel locations on the image. These pixels are then set to 0. As a result, the barrel is eliminated from the final binary image. Figure 13 demonstrates the method.

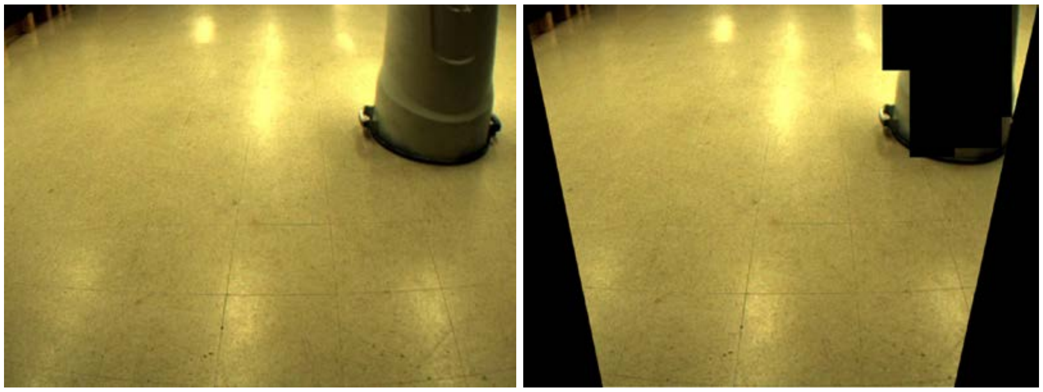

Figure 13: Barrel masking method

For the 2014 competition, our new algorithm performed well as the robot was able to navigate the lines even with the presence of blue and white barrels. However, the algorithm speed was not sufficient in previous years, and will be improved for the 2015 competition.

h. Motor Control

The lack of proper speed control in the previous year led Q to rapidly decelerate or accelerate when approaching or moving away from obstacles. In order to combat the issue of rapid speed change, we implemented a step-wise speed control algorithm. This algorithm converted raw SICK data into actual distance measurements enabled Q to move at 5mph when more than 3m away from the nearest obstacle. For distances between 1 to 3m, Q's speed decreased as a linear function, resulting in gradual deceleration as opposed to a sudden halt when approaching an obstacle.

6. Interoperability Profiles Challenge (IOP)

The IOP Challenge requires each team to implement a set of communication protocols defined in JAUS Transport Specification (AS5669A), JAUS Core Service Set (AS5710), and the JAUS Mobility Set (AS6009) [5,6,7]. The protocols together define various JAUS messages by which the unmanned vehicle communicates with the Common Operating Picture (COP). COP is a simple validation, reporting, and recording tool for the IGVC judges to use while verifying implementations of JAUS. Q will be required to perform the following tasks with JAUS: Transport Discovery, Capabilities Discovery, System Management, Events, Velocity State Report, Position and Orientation Report, Remote Control, and Waypoint Navigation.

The JAUS Transport Specification (AS5669A) specifies how JAUS messages should be formatted over User Datagram Protocol (UDP) packets. Q, as an IOP-compliant vehicle, could send and receive UDP packets to and from the COP. The JAUS Core Service Set (AS5710) provides specification for implementation of the discovery, access control, and management services whereas the JAUS Mobility Service Set (AS6009) provides specifications for the messaging to be used for position communications and waypoint based navigation so that the messages and associated protocols are standardized. (A service is a feature that is to be implemented via a group of related messages.) The JAUS services are defined using Unified Markup Language (UML) diagrams and the state transition table which provides details about conditions or triggers required for transition between states.

The JAUS module of Q consists of multiple parallel loops. They communicate via special data structures known as queues in which the consumer loop waits for data segments produced by the producer loop. LabVIEW supports a robust implementation of queues so that two loops can share one queue safely. Incoming messages are divided into two categories: simple and complex (See Figure 14).

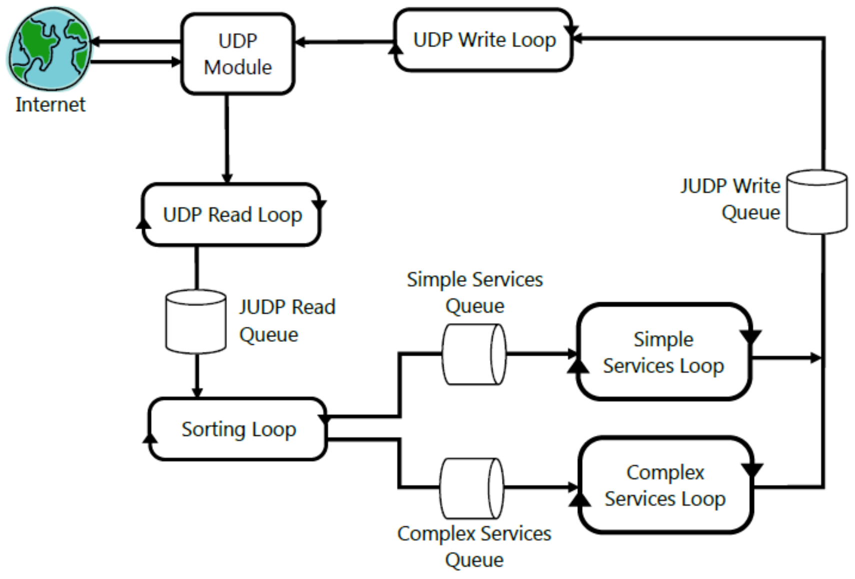

Figure 14: Data flow of incoming and outgoing JAUS messages. JAUS messages are wrapped in UDP packets, which are handled by the standard Internet infrastructure.

The messages in the complex category require that multiple overlapping states are kept over many loop iterations. In other words, they require a finite-state machine. Rather than implementing the finite-state machine ourselves, we make use of the State Chart Module provided by LabVIEW. The module lets us simulate states that contain sub-states. It can also initiate specific actions according to the current state of the machine; for the sake of modularity, we off-load the resulting actions to separate sub-VIs.

Last year only a subset of messages presented in the IOP interface standard had to be present. However, this year's competition requires additional messages. We upgraded our JAUS component to support the additional messages. In particular, we added support for remote control. The SetWrenchEffort message dictates translational and rotational movement with commanded effort percentage. Since Q already supported remote control by an Xbox controller, the work was a matter of exposing the capability to the IOP interface.

We fixed bugs in the global events manager, which was added at the last year's competition. Required by the IOP interface task, the Events service regularly notifies the remote client of such information as velocity, local position, status, and heartbeat. The client specifies those pieces of information it wants and the interval at which it wants to be notified. The IOP-compliant component is to send response messages at the specified time interval. Some of Events messages were not formatted correctly and some exhibited incorrect semantics. The bugs have been fixed this year.

Adding new messages and making revisions required extensive testing. We wrote an IOP tester in C++. The application has an intuitive graphical interface that lets us generate any string of messages we want and send them to either the real IOP component or a simulated one. It takes minimal work to support a new class of messages because the code for sending and receiving messages is separate from the code for generating and decoding them. All it takes to add a class of messages is two small modules to generate and decode these messages. The framework also helped us discover bugs and undefined behavior from the existing code base.

7. Failure Modes

Several ways in which the robot could fail during the competition were identified and corresponding resolution methods were implemented. One problem the robot could face during the autonomous navigation is turning back to the start instead of navigating forward. In order to resolve this problem, a path history feature was added. With this feature, the robot knows which direction it came from and assigns high cost to that direction. Similarly, the robot sometimes is unable to acquire GPS coordinates due to freezing of the antenna. This makes the robot go in a straight line instead of towards the target coordinate, which can result in failure. Our robot solves this issue by stopping whenever the GPS coordinates freeze and waiting until the new coordinates are available again. This way, the robot does not travel in the wrong direction while the GPS is frozen. Another possible problem our robot might face during the competition is the change in sun’s intensity. Our algorithm accounts for this problem by implementing a feature that decreases gain when the image is too bright and increasing gain when the image is too dim.

8. Safety

Safety has been given the utmost priority in the design of Q 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.

Wireless Emergency Stop

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 Q and on the transmitter.

9. Vehicle Cost

Component List	Price (USD)	Cost Incurred (USD)
Basler Camera + Wide Angle Lens	900	900
Crescent A101 DGPS	1500	1125
Caster Assembly	150	150
Chassis	650	0
Encoders	100	100
Honeywell Compass (HMR-3300)	750	0
Motors	1000	0
NI cRIO-9022 Real-Time Controller	3199	3199
NI cRIO-9133 Reconfigurable Chassis	1999	1999
NI EVS-1464RT Embedded Vision System	4499	4049
SICK LMS291 Laser Range Finder	4000	4000
Power Supply Board	150	0
Encoded Receiver/Transmission Set	77	77
Remote Control	180	0
Roboteq AX3500 Motor Controller	400	300
Wiring/Electrical	50	50
Total	19604	15949

10. Sponsors

We would like to express thanks to our sponsors for making our trips to IGVC possible every year, and for their generous donations to Q's components.

- Enterprise Rent-A-Car
- Hemisphere GPS
- Honeywell International Inc.
- National Instruments
- PerMobil Corporation
- Travelers Insurance
- Trinity College

11. References

[1]. “USA - Products - Support - Product Support - Permobil." Power Wheelchairs - Permobil.Web. 14 May 2010. <http://www.permobil.com/USA/Products/Support/Manuals/drivers/>.

[2]. Shah, Ashay; Murtuza Patanwala, and et. al. "Pushpak IIT Bombay." Intelligent Ground Vehicle Competition (IGVC). IGVC Design Reports. (2012)

[3]. Ulrich, Iwan, and Johann Borenstein. "VFH+: Reliable Obstacle Avoidance for Fast Mobile Robots." Proceedings: 1998 IEEE International Conference on Robotics and Automation, May 16-20, 1998, Katholieke Universiteit Leuven, Leuven, Belgium. IEEE International Conference On Robotics and Automation, Leuven, Belgium. Piscataway, NJ: Robotics and Automation Society, 1998. 1572-577. Print.

[4]. Giri, Binod, et al. "Improved Obstacle Avoidance and Navigation for an Autonomous Ground Vehicle." *IS&T/SPIE Electronic Imaging*, 2015.

[5]. Wright, Adam, Orko Momin, Young Ho Shin, Rahul Shakya, Kumud Nepal, and David Ahlgren. "Application of a Distributed Systems Architecture for Increased Speed in Image Processing on an Autonomous Ground Vehicle (Proceedings Paper)." Intelligent Robots and Computer Vision XXVII: Algorithms and Techniques. Proc. of IS&T/SPIE Electronic Imaging, San Jose. Vol. 7539. San Jose: SPIE, 2010. Print.

[6]. “JAUS / SDP Transport Specification” (AS5669A), Revised, SAE, 2009.

[7]. “JAUS Core Service Set” (AS5710), SAE, 2008.

[8]. “JAUS Mobility Service Set” (AS6009), SAE, 2009.