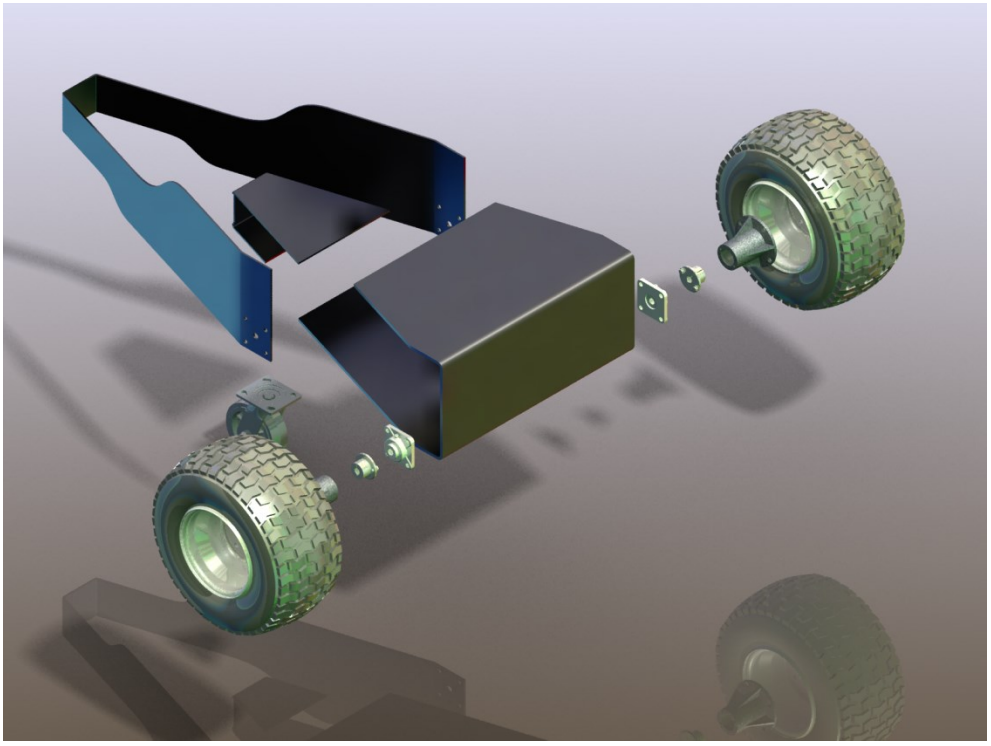
R. Daneel Olivaw II

IGVC | 13



Kettering B - IGVC Design Report

Horcasitas Jorge, Lombardi Lex, Seeman Garrett

This document contains the mechanical, electrical, and software design and system overview of the Kettering University B Section robot entry, R. Daneel Olivaw II, for the 2013 Intelligent Ground Vehicle Competition.

Required Faculty Advisor Statement

I certify that the engineering design of the robot, R. Daneel Olivaw II, described in this report, has been significant and equivalent to what might be awarded credit in a senior design course.

James S McDonald
Dept Head, ECE, Kettering University

Introduction

The B Section Kettering Robotics Team has learned much from last year's submission, including the importance of considering system integration in a high level design and leaving ample time for robot testing. Since last year, we have completely revised our software high level design, updated our electronics configuration, and recruited new team members, among other things. It is our pleasure to introduce this year's submission, R. Daneel Olivaw II.

Similar to last year's submission, our design philosophy focused on finding low cost and low man-hour solutions. For example, we are using the 24V electric motors and the acrylic spacer from the 2005-2006 Bulldog I/II IGVC entry. Expenses saved through the reuse and repurposing of older materials allowed us to focus our budget on more critical components like sensors, controllers, circuits, and batteries. Generous donations by Jervis B Web included high quality batteries, a primary power contactor, an on/off switch, an e-stop button, and an indicator stack. Porcupine Electronics, LLC also graciously offered two of their Fluke laser rangefinder interface boards we are now using for our laser rangefinder component.

We relied on open source hardware and the maker movement knowledgebase to provide proven solutions to common problems. The result of our design is a remarkably agile, durable, and easily implementable ground vehicle.

2012 – 2013 Team

Jorge E. Horcasitas	Team Captain, Software Project Manager
Lex Lombardi	Electrical and Mechanical Lead
Garrett Seeman	Software Lead, Software Product Owner
Michael Graham	Business Development Director
Jonathan Neuendorf	Electrical
Justin Katnik	Electrical
Joe Angelo	Mechanical
Loi Huynh	Software
Jacob Keeler	Software
Douglas Blaisdell	Software

Strategy

Software

Our school's schedule poses a major challenge: The team is only on campus for twenty-two weeks per year with a three month break after the first eleven weeks. This prompted the software team to adapt, and later modify, the Agile Scrum software development process in an attempt to increase development speed and easily break up the various components. The software team itself had three developers, a scrum master, and product owner (doubling as the lead developer).

Although the team initially wanted to create various sprints for the effort, we quickly realized that the already rapid environment at our school made it difficult to maintain the weekly standups, long planning meetings, and retrospectives used in Scrum. Stand-up meetings were instead held weekly and one running backlog was used to keep track of effort remaining.

Mechanical

R. Daneel is shaped like a triangular tub. It has 3 wheels; two are independent drive and the third is a trailing caster wheel. This drive method allowed us to integrate our forward, backward, and turning motion controls into only two control values: left and right motors. The low speed agility and ability to turn about the center of the drive axle fit with our design criteria well.

The polygonal body shape was designed in Solidworks and cut by our team on a CNC plasma cutter. Steel was selected primarily for its ease of fabrication.

The drawback of this low design is not having enough vertical height to meet the emergency stop requirement and sensor needs. Late in the design process a tail stack and sensor mast were fabricated to address these issues.

Electrical

Our low cost/low man-hours maxim is echoed throughout the design of the electrical system. Prebuilt components with standardized connectors were our first choice throughout the entire build. Except for the motors, every individual electrical component can be removed and reinstalled without the need to first remove another nearby component or access panel.

Our electrical system consists entirely of commodity hardware. For example, our mainboard is the Foxconn H67S. It is common, inexpensive, powerful, and small. Early in the design process we agreed on a USB interface for all of our sensors and actuators.

Mechanical Design

Fabricated out of 10ga steel sheet, the shape of the tub allowed us to drill and mount components anywhere, providing rapid development.

Independent drive was the logical choice for our design criteria, especially because we already own a pair of wheelchair motors used in Kettering's 2005 and 2006 IGVC entrant, Bulldog I/II. These off-the-shelf geared motors are intended for use in wheelchairs with similar loads and speed requirements – high torque and low output speed. This provides an excellent balance between simplicity of control, design time, and agility.

Our first challenge was to determine the ideal size and type of wheels and tires that would match the grassy competition surface and the maximum output shaft velocity of our motors. Through persistent searching, our team was able to locate two ideally sized wheels and tires. A golf cart repair business had used golf cart wheels and tires with an outer diameter of 17.5". Our motor's maximum output shaft speed is 220rpm. This provided an ideal solution for a vehicle traveling over a grassy field at a limited speed of 10mph. Taking the maximum motor speed and the diameter of the outside of the tire, we can find the theoretical top speed of the vehicle.

$$C = \pi D = 17.5\pi = 55\text{in}$$

$$55in \times 220rpm = \ 12100in/min$$

$$12100in/min \div 12in/ft \div 5280ft/mi \times 60min/hr = 11.5mph$$

In this type of vehicle, a wide track further facilitates maneuverability by providing a long lever arm to rotate about the vertical axis. The track is 3ft wide, and a significant amount of weight is near or below the axles. The force required to rotate about the center point of the axles is minimal compared to a similarly designed and operated 4-wheel skid steer. Changes in heading are accurate, efficient, and even graceful.

The ideal location for the center of gravity is low and centered between the axles. The shape of the tub provides a low center of gravity designed for low-speed stability. The batteries and the drive motors are the heaviest components. Naturally, we decided to mount them symmetrically and as close to the drive wheels as was practical.
The trailing end of our vehicle is supported by a single caster wheel. A rectangular 4-wheel chassis with two caster wheels was also considered. Our primary concern was that two or more caster wheels may turn in opposite directions and bind at a potentially critical moment. Our team settled on a single caster.

The open shape of the tub provided us with a burly, drillable surface area. This allowed for rapid, flexible design and ease of assembly to provide adjustable, stable mounting locations. This mast incorporates the GPS, IMU, vision system and ultrasonic and laser rangefinders into one easy to install package and provides for each sensors particular needs.

A tail stack is the final feature of our vehicle. It adds 1.5ft of height to the tail of the vehicle. It serves as an ideal location for the emergency stop button, indicator lights, and camera. The indicator stack is mounted at the top of the tail stack, displaying the vehicle's operating condition. Red indicates E-stop, yellow continuous indicates manual control, yellow flashing indicates automatic control, and green indicates power.

A mount was added on the front of the chassis to accommodate our two laser rangefinders. A servo is used to rotate the laser rangefinders about the vertical axis. However, it was determined that a second servo for rotation about the horizontal axis would provide minimal benefit during operation while introducing more that would need to be controlled. Thus, we use a bolt and wing nut to manually adjust the laser rangefinders about the horizontal axis. Two "L" brackets were used to attach this mount to the front of the chassis allowing enough space for the laser rangefinders to rotate.

Electrical Design

Emergency Stop System

Both wireless and manual emergency stop functionality is included. The manual deactivation control is activated by a large red button on the tail stack. Both the manual input and the wireless input must be energized for the emergency stop relay to be energized and allow the vehicle to drive. When de-energized, the emergency stop relay

directly shunts both of the motors leads' to ground, forcing the vehicle to stop. The red light in the indicator stack illuminates when the fail-safe is activated, clearly broadcasting its emergency stop condition.

Sensors

Our inertial measurement unit (IMU) is a YEI 3-Space. It contains an integrated accelerometer, gyroscope, and magnetoscope as well as a microcontroller that processes the raw input. This sensor provides easy integration with LabVIEW, excellent resolution and a USB interface. Our GPS sensor, a GlobalSat BU-353, is an all-in-one USB device as well. Our vision system uses a Microsoft LifeCam Studio capable of delivering 1080p at 15fps.

Microcontroller

An Arduino works as a single embedded solution to tie all of our non-USB sensors and actuators to a single USB port. It hosts our ultrasonic rangefinders, laser rangefinder servo control, and controls the autonomous indicator light. The Arduino is also responsible for detecting emergency stop conditions and informing the control software an emergency stop has occurred. The Arduino uses a predefined

packet structure over a USB COM port. The largest reason for choosing an Arduino is that they are cheap and easy to use, with a wide variety of freely available code online.

Motor Controller

Our motor controller, a Roboteq 50V SDC2150 was selected for its flexibility. Its native USB and RC pulse input allow us to program auto fallback of control in case of a loss of autonomous control. At any point the operator can override autonomous control remotely as well, which is useful during testing. When specifying the motor controller, we measured our motor's stall current and considered this the expected continuous load. This method tends to oversize controllers by a significant margin. However during testing we found the motor controller to be underpowered. We realized our data on stall current was erroneous due to the weak power supply used during testing. We intend to correct this in our design for IGVC 2014.

Processing

An off/on/start selector controls power to the entire system. The start position is connected to the mainboard and boots the PC directly through its PWR_ON header. We use a quad core i5 at 2.6GHz with 8GB of ram. For power we use a Mini-Box M4 DC-DC ATX power supply. These power supplies provide for an array of programmable options from a USB interface and offer an independent, hardware switch activated standard ATX power supply mode. Though it is overkill for our needs, this system was less expensive than any other design we considered.

Software Design

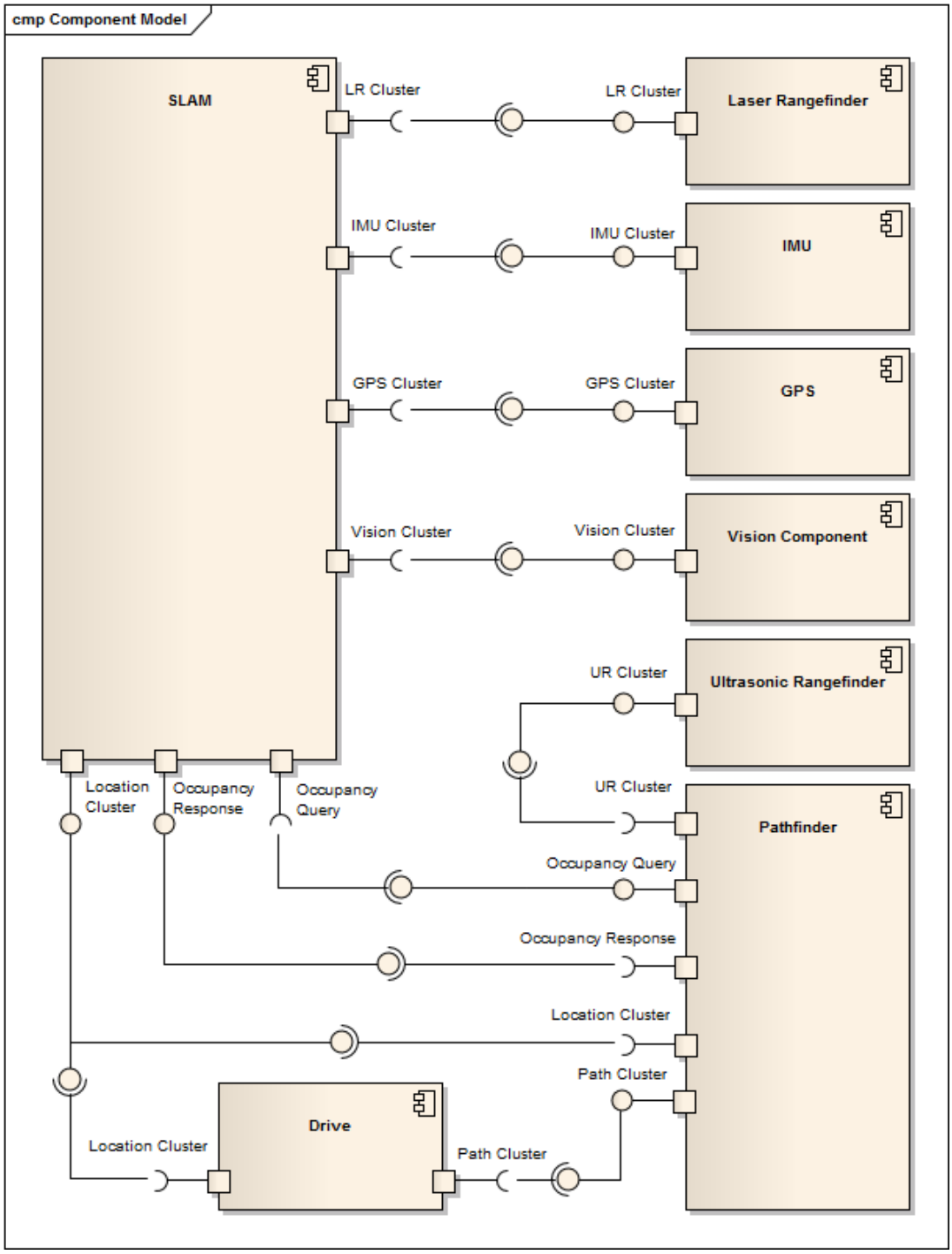
High Level Design

The system is divided into eight components with each having specific functions that ultimately enable the system to perform according to the IGVC software requirements.

Our approach is one that enforces separation of concerns among components to ensure a modular system is implemented. A decoupled, modular system enables testing of independent components and allows the team to swap out hardware or create new implementations of each component at any time without having a negative impact on the rest of the system.

The design of how each component's implemented is found in the component model on the following page. The UML interfaces represent the LabVIEW cluster datatype and

will enforce specific data publishing (output) and subscribing (inputs) per component regardless of its lower-level implementation.



Timers

All components will have timers with predefined expiration times. When a component's timer expires, the component shall do the following: read data from any

cluster it is subscribed to, perform its internal operations, and write data to the cluster it publishes. Effectively, the data clusters provide a snapshot of the data published per component from the last time its timer expired. The Component Design section specifies what each component's published data cluster contains. The expiration times of each component's timer is found below.

- IMU – Every 50 milliseconds (ms)
- GPS – Every 500 ms
- Vision – Every 200 ms
- Laser Rangefinder – Every 20 ms
- Ultrasonic Rangefinder – Every 20 ms
- SLAM – When any timer expires
- Pathfinder – Every 3000 ms
- Drive – Every 50 ms

Simultaneous Localizer and Mapper (SLAM) Component

Localization

The robot determines its location based on data from the GPS and IMU components. It is important that this location be as accurate as possible as all data collected is recorded based on where the robot believes it is. In order to facilitate this, we start out stationary until the GPS returns a location within a reasonable amount of error. This position gets saved as the offset and every position recorded in the future is transformed into a map location based on this offset.

Since the GPS only returns a reading every 500ms or so, the IMU is used to track the robot's position in between readings. If for some reason a GPS reading returns with a large amount of error, we ignore that reading and keep tracking based on the data the IMU is returning until the GPS samples an accurate reading.

Mapping

The robot stores all the information it collects about its environment from the LR4 and the Vision sensors in a large occupancy grid consisting of a couple million 100cm^2 cells represented as a two dimension array. Since this is running on a desktop motherboard, there is plenty of memory available, eliminating the main disadvantage of this approach over something such as a quadtree. Each cell consists of two 8-bit integers indicating: the number of times this cell has been updated, and the probability (from 0 – 255) of an obstacle existing. When an obstacle is detected in a cell, increment the update number then divide 255 by the number of updates and add that to the current location's

obstacle probability. If the cell is recorded as blank, subtract instead. We treat a number over 30 to be occupied and a number less than that to be free. This parameter can be manually adjusted to better suit the environment and the error rate of our sensors.

Pathfinder Component and Obstacle Avoidance

In order to find its way around the course, the robot uses a modified version of the A-Star pathfinding algorithm. The pathfinding component was created from scratch in LabVIEW in order to integrate easily with the rest of the projects code base. A-Star was chosen over other approaches (such as D-Star or LPA*) because it is more simple to code and we have more control over when it runs. Traditional A-Star is not fast enough to sift through a map of millions of nodes in a matter of seconds, so an optimization routine called Jump Point Search (JPS) is applied which drastically reduces path calculation time by a couple orders of magnitude, depending on the situation. JPS reduces the time spend iterating over large areas of open space.

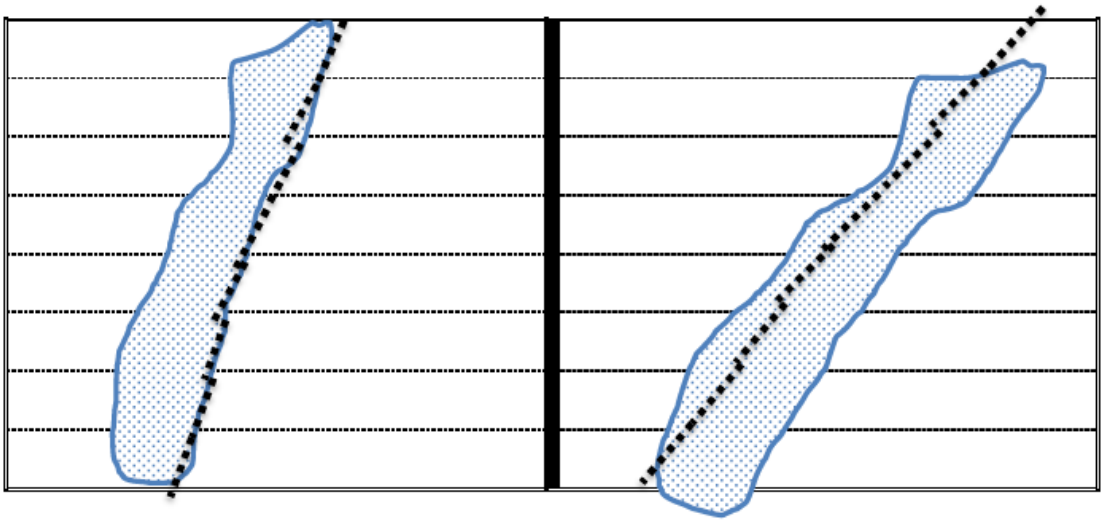
These iterations require a significant amount of time searching the open list for updated nodes which takes an increasing amount of time as the open list increases in size. The JPS optimization essentially replaces this operation by recursively opening neighboring nodes if they have no obstacles as neighbors, and picking only one of many symmetric paths. By choosing the path that takes any required diagonal moves as soon as possible it eliminates a large amount of the search and drastically reduces the number of nodes in the open list.

Once a path is found, it is then passed to the drive program. For sequential iterations for paths that are quite long an additional optimization will only recalculate the beginning half of the path. This quickly updates the path with any changes around the robots immediate vicinity, preventing collisions with previously unseen obstacles or obstacles that are difficult to detect at a distance such as the boundary lines.

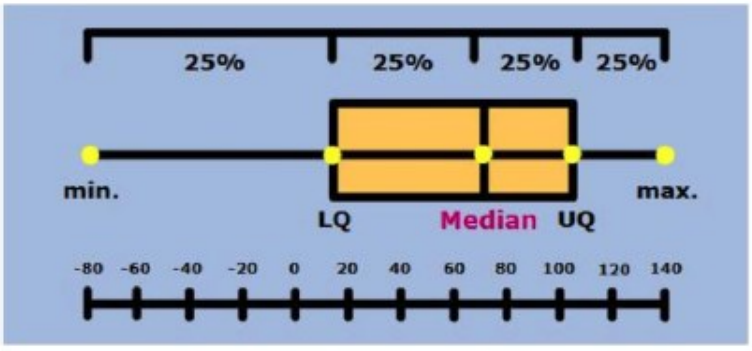
Vision Component

The lane detection algorithm takes the images from the camera and applies an HSL threshold to translate them into a binary red-and-black map, where the red represents the white pixel (essentially the white lanes on the ground), and the black represents everything else. The algorithm splits the image into two equal halves, and each half will

is split into eight equal rows. Every section of those 16 areas will be put into a “best-fit-line” VI to calculate the best fit regression line.



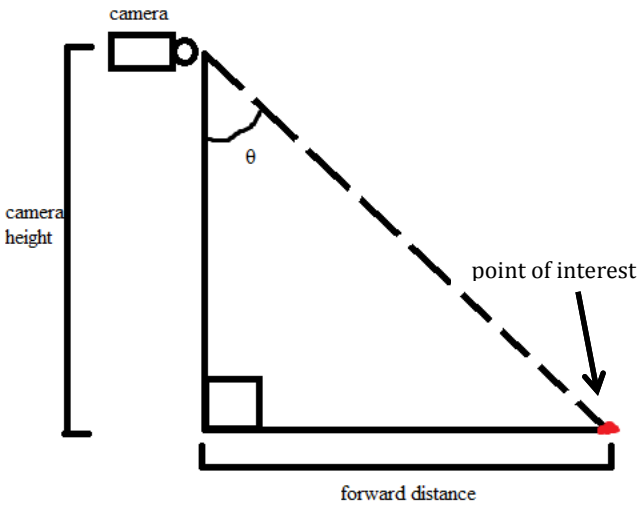
There is a great deal of noise in this best-fit-line, and sometimes the algorithm does not yield the best line to represent the current situation if there is no noise anywhere.



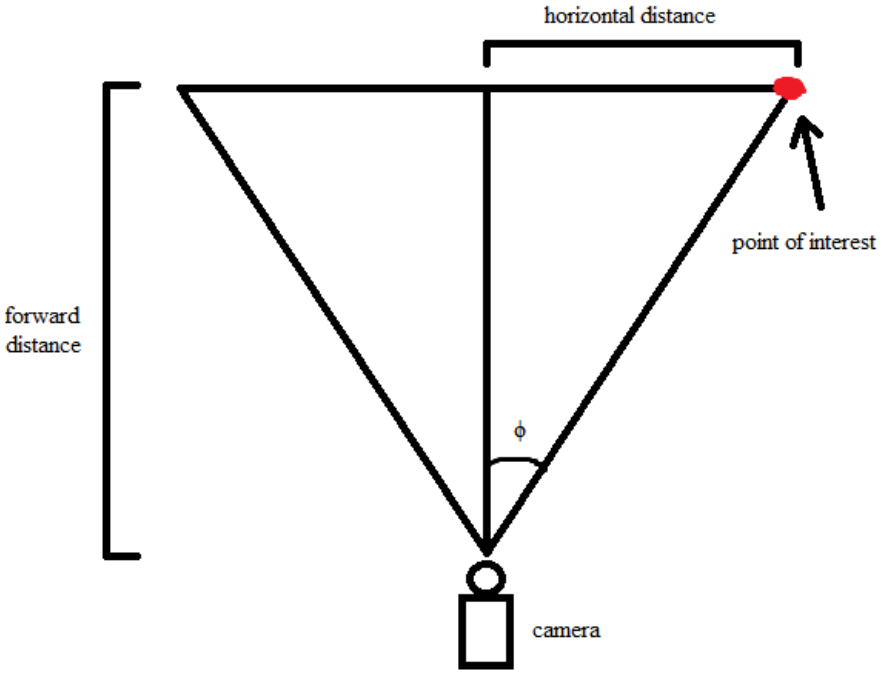
For the left half of the image, we only use the points which belong to the range from LQ to max. For the right half of the image, we only use the points which belong to the range from min to UQ. We repeat the above method twice, so that we can filter more noise. The method that we use to reduce the noise is to filter the points by the idea of calculating quartile.

Once lines have been superimposed on the image, a pixel coordinate to decimeter convert is used. This part of the code is used to change the lines found by the vision program into obstacles with set positions. Once the vision program finds points on the lines it draws across the camera images, the Coordinate to Decimeter Converter changes these coordinates to discrete distance values.

In order to find how far forward the point is, the converter relies on the height and vertical angle of the camera. The forward distance can be solved for by finding the product of the height and the tangent of the angle of the particular pixel of the camera.



The distance horizontally from the robot must also be found in order to provide a location for the obstacle. This can be found using the forward distance and the horizontal camera angle. The horizontal distance can be solved for by finding the product of the forward distance and the tangent of the horizontal angle of the particular pixel of the camera. The distances of these points are then sent to the SLAM component to be mapped as obstacles.



Laser Rangefinder Component

The Laser Rangefinder Component is responsible for extracting distance measurements from the two LR4 Laser Rangefinders asynchronously. Since the Fluke 414Ds we are using have a sample rate of ~400ms, the rangefinders shall be sampled at 200ms offsets from each other to increase the total sample rate to ~200ms. The data is used by the Mapper in the SLAM Component.

Ultrasonic Component

The Ultrasonic Component is responsible for extracting data from a microcontroller reporting instance measurements from 2 ultrasonic sensors. The data is used to create a proximity sensor used to stop the robot if objects come within two to three decimeters from the front of it. During this time, the robot will recalculate its path. This value is tuned during testing for optimal stop conditions.

IMU Component

The IMU component is responsible for extracting data from the IMU hardware; specifically the change in positions since last reading. This data is used by the Localizer in the SLAM Component. The data includes change in X and Y position as well as change in angle since the last sample (timer expiration).

GPS Component

The GPS component is used to read the Latitude and Longitude position of the robot every 700 milliseconds from the GPS hardware. The data is used by the Localizer in the SLAM Component. This component is used along with the IMU component to determine the robots current point on the map.