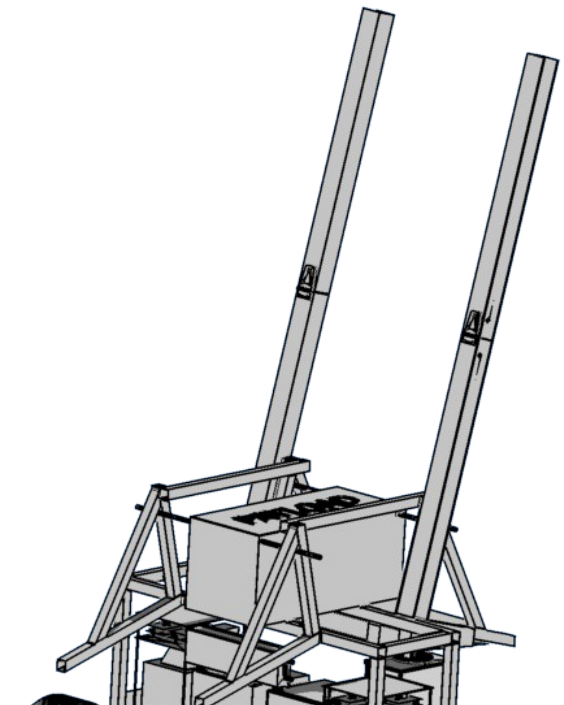
18th Annual Intelligent Ground Vehicle Competition

Georgia Institute of Technology – RoboJackets

Design Report: Jeani

June 4-7, 2010



Abstract

For the 2010 IGVC team competition, the Georgia Tech Robojackets IGVC team has designed a new mechanical base with new sensors, and made major changes to the software architecture. Moving to a four wheel drive vehicle should improve stability on the ramp, while still providing sufficient maneuverability in the rest of the course. The addition of a LIDAR system should improve out ability to detect obstacles and ramps. A new modular driver architecture decreases code complexity and improves code reuse, making it easier and faster to add software support for new electronic hardware.

Design Process

The 2010 competition marks the first time in three years the Georgia Tech team has designed a new mechanical base, so the team was restructured to better accommodate the workload. The team was largely divided into hardware and software / electronics teams. Following a collective conceptual design phase, the mechanical team completed the detailed design of the base while the software and electrical system team worked on integrating new sensors. Once the detailed CAD was done, most members helped with the actual mechanical build.

For most of the year, the team met once per week at our shop on campus. As the design became finalized, we started meeting more often in order to actually build the robot. Tasks were loosely broken down based on the individual member's experience, although with such a small team everyone did a bit of everything. Junior members were given well defined tasks, and mentored by a senior member.

This years team members:

Matt Carson – 3rd year Mechanical Engineering

Tasks: Detailed CAD design

Paul Foster – 4th year Mechanical Engineering

Tasks: Software, Electronics assembly

Rivers Ingersol – 2nd year Mechanical Engineering

Tasks: Electronics Stack design, Encoder Mount design, Misc. Mechanical

John Madden – 4th year Computer Science/Mechanical Engineering – Robojackets Public Relations

Tasks: Misc Software, Electronics, Mechanical Assembly

Stefan Posey – 5th year Aerospace Engineering – Robojackets Club President

Tasks: Misc Mechanical

Jacob Schloss – 3rd year Aerospace Engineering – IGVC Team Lead

Tasks: MCU firmware, Misc software, Misc electronics

Overall Design

The new robot for 2010 is a refinement of our previous design, following several years of competition with the same mechanical base. This year's robot is a four wheel drive base, with a rear mounted camera mast and forward and rear facing LIDAR.

In previous years, we entered a two wheel drive robot with a ball caster pivot. While this design is highly maneuverable, and has a small turning radius for its size, it is very difficult to control when going down ramps. Moving to a four wheel drive robot should provide adequate stability when going up and down ramps. By decreasing the robot's width slightly, we hope to keep an adequate amount of maneuverability.

Both vision and forward LIDAR systems are the primary sensors for this robot. This is an improvement over our previous design that only used vision to detect obstacles, and should be able to out preform a solely vision based system.

Mechanical Design

This years competition marks the first time in three years that the Georgia Tech team has submitted a new mechanical base. With a mature software stack, we have decided it is time to fix some of the limitations of our previous hardware, such as poor ramp performance and wasted internal bay space.

The most noticeable change is the removal of the ball caster. While the ball caster was very good for maneuverability, it caused the robot to form an inverse pendulum when descending ramps. This proved difficult to control. Our solution is a four wheel drive skid steer vehicle. This should allow for increased controllability on the ramp.

The robot is controlled by skid steering, which is much simpler to design than Ackerman steering. The robot utilizes four NPC robotics 1.1 HP DC motors. Two 12V sealed lead acid gel cell batteries are connected in series for power for the motors and LIDAR. The robot has per-wheel suspension consisting of pneumatic cylinders with springs, providing shock absorption and damping to help keep bumps from swinging the camera too much.

As in our previous design, the camera is mounted on a mast at the rear of the robot. This camera placement maximizes the field of view for the robot. The LIDARs are mounted between the motors in the front and rear of the robot, to provide front and rear quadrant object detection. It was important that the mechanical base be designed such that nothing block the lasers emanating from either LIDAR. This meant that all vertical bars had to be between the LIDARs near the center of the robot which created some high stress areas in the frame, but this was countered by adding material to strengthen up these areas.

As this robot has more secondary electronics than in previous years, designated zones for power and control electronics have been created, each with their own electronics tower. This should allow for easy maintenance and reprogramming, as we can individually remove an electronics stack from the robot, and take it to an electronics work bench without moving the mechanical base. Each of the control stacks have their own USB hub, which all boards in the stack feed into, allowing for a single connection from stack to laptop, rather than three or six individual connections. The interior bay has been sized to accommodate two electronics stacks and two batteries with little wasted space.

Electrical System

The electrical system in the 2010 robot is an incremental upgrade to the electrical system featured in the previous robot. The electrical system can be broken into three parts: drive electronics, data acquisition / command electronics, and sensors. We use six ATmega168 microcontroller based Arduino boards for command and data acquisition, as well as 4 OSMC motor controllers and a laptop. This year also features the introduction of our first in-house Arduino shield, an opto-isolater board designed for controlling an OSMC. We also have a custom FPGA board for data acquisition from the LIDAR.

Four Open Source Motor Controller H-bridges are used to control the motors. These boards are rated for high current operation in excess of the installed motors stall current, which should make them highly reliable. The OSMC boards takes an input PWM signal and direction signal from our custom opto-isolated motor controller arduino shield, and outputs a high current PWM signal to the motors. The motor interface arduino limits the speed to the legal 5 mph both by limiting the max PWM width and by a kill pin pulled forward from the wheel encoders. Laptop side programs take input either from a wireless gamepad or the vision code to command the motor arduino, and ultimately the motors.

The Arduinos and FPGA serve as the intermediate controller between most of the sensors and the laptop. Four Arduinos sit between the laptop and the wheel encoders, counting the quadrature ticks from the encoders and provide odometry and velocity information to the laptop over USB. The FPGA board serves to convert the non-standard serial output of the LIDAR to a format sendable over USB that the laptop can more easily process. Both the Arduinos and the LIDAR interface board are powered over the USB bus.

The use of open source, off-the-shelf electronics reduces complexity, while still giving us detailed information about the electronics in case we have any question about operation, or want to redesign or extend the board. Both the OSMC and the Arduino are open source, with full schematic and reference material available.

The robot has both wired and wireless e-stop to kill power in case of emergency. The e-stop box contains an industrial relay that disconnects the motors from the battery when tripped. The relay is self latching. In order to start the robot again, the e-stop must be reset using the energize button. This simple design has proved to be very reliable.

Sensors

The robot features video camera, LIDAR, current sensors, and wheel encoder sensors. The vision and LIDAR are used for obstacle and ramp detection, while the encoders are used to provide feedback information to a speed controller. It is expected that the LIDAR will augment the current vision system very well, and help to detect large smooth features such as ramps and barrels well. The addition of velocity feedback will also allow us to regulate speed on irregular terrain, such as when cresting the ramp, and when going over speed bumps. By combining the information from the wheel encoders and the current sensors, a rudimentary slip detector can be created by detecting when the wheels spins without drawing much current.

Software Architecture

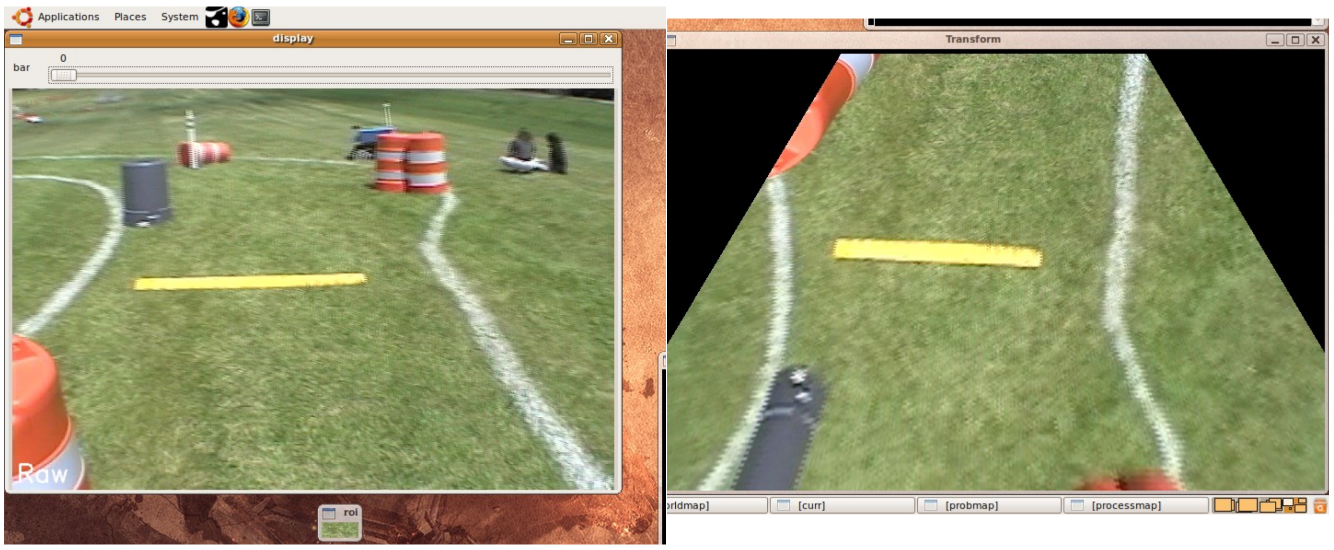
Software on our robot can be split between microcontroller firmware for data acquisition and control, and laptop side processing. Extensive use of open source libraries such as OpenCV and arduino/wiring allow for reduced code complexity and an easier entry point for new members.

A standardized serial interface and serialization layer has been written, which increases the amount of shared code between the various arduino drivers. This greatly reduces the amount of time needed to create even a complex interface, while providing a base of “known good code” that new members can extend when they start writing software to communicate with a new sensor or device.

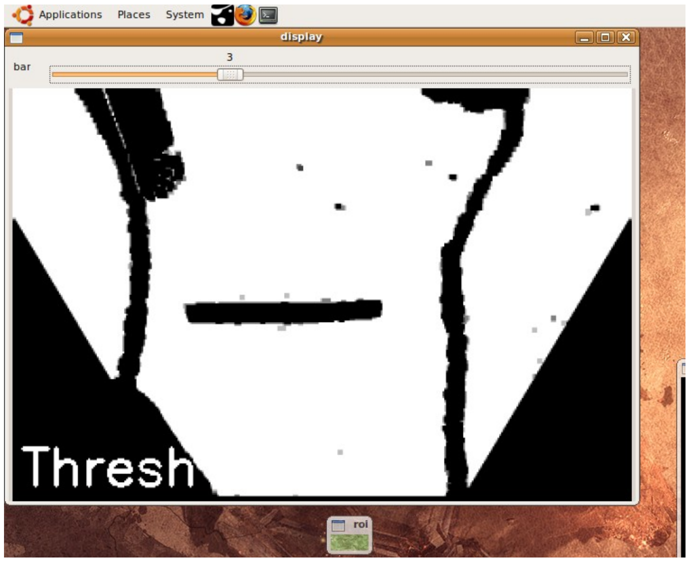
Software on the laptop generally processes and makes decisions based on the data. Several open source libraries are used, namely OpenGL and OpenCV for image processing, and Boost for serial communication and threading. The software currently only runs on Linux, however work was completed this year to port the last POSIX specific code to Boost, allowing the code to be moved to another operating system in the future.

Algorithms

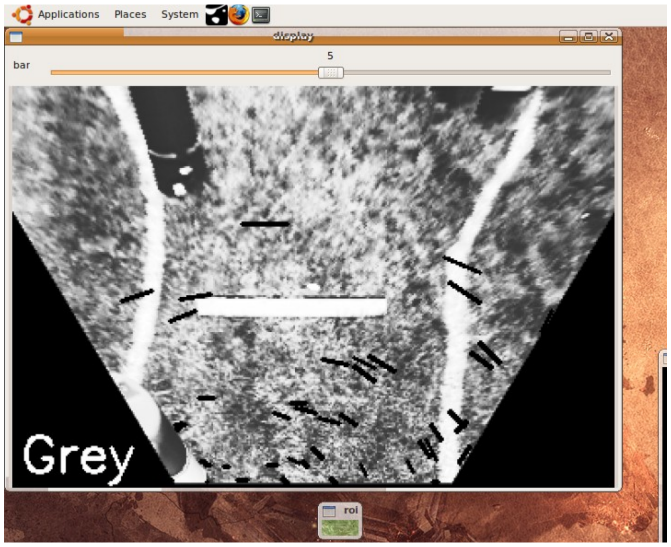
The robot uses vision as the primary method of detecting obstacles and lines. The vision algorithm has been developed and modified over several years of competition and is considered reasonably robust. After passing the input video through several different algorithms, a short-term map of the world is created, which the robot is driven off of.



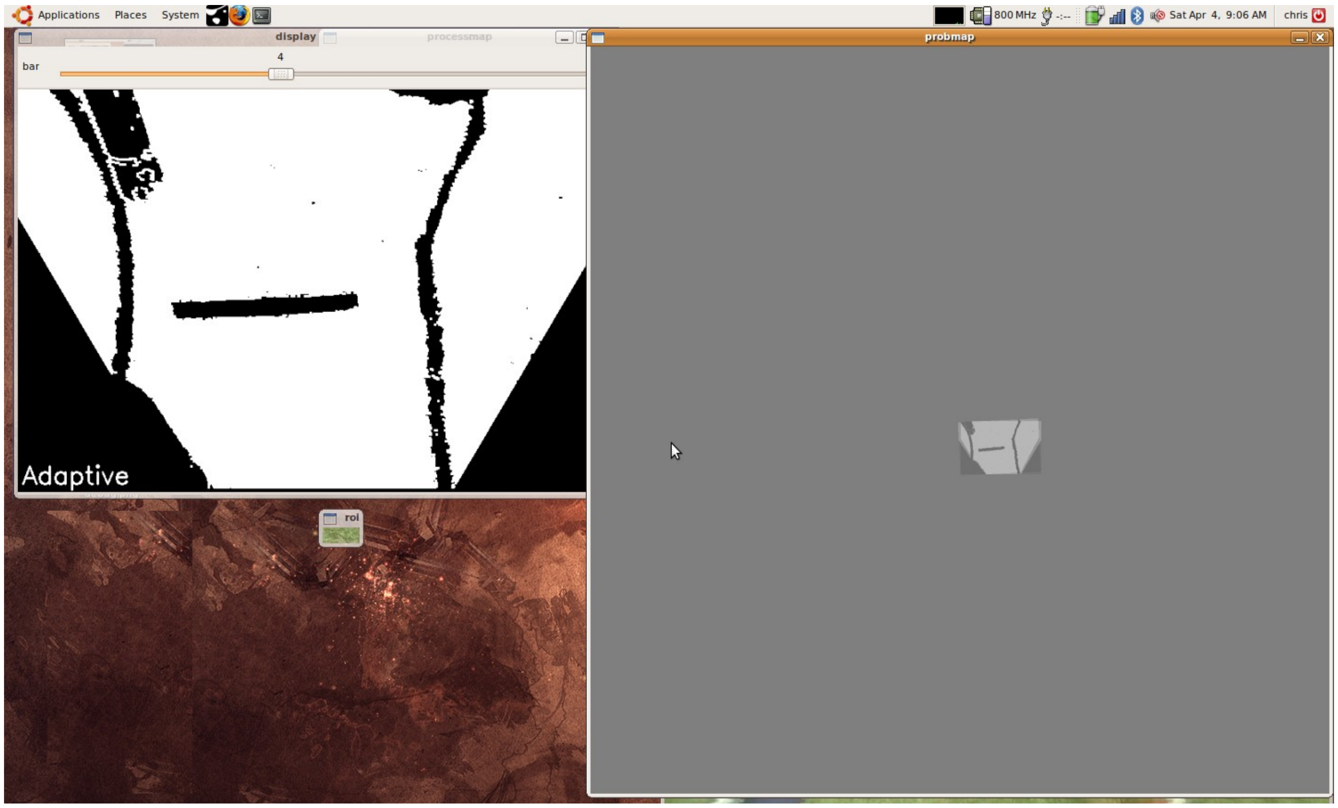
The input video, above-left, is first passed through an inverse perspective transform, as seen above-right. This transform makes both near and far off objects a normalized size, and makes the image appear to be taken from directly overhead. This flattened image assumes the course is a plane, which does cause distortion of the barrels, but this is accounted for in the mapping algorithm. The transformed image is much easier to process into a map than a normal, perspective image would be.



The images is then color segmented and thresholded based on the color that is centered directly in front of the robot, as seen above. Safe colors are marked white, the rest are black. The color is averaged in time between frames to allow for some variation in color, for example, if there is dead pach in the grass. This allows the robot to operate on many different surfaces with the same software. For testing we have operated on asphalt parking lots, navigating between the lines marking parking spaces.



After converting the transformed image to grayscale, feature tracking is preformed between subsequent frames. The tracked features are denoted by the black lines in the above grayscale image. The algorithm looks for features that have been translated and rotated between frames. This allows us to build a set of likely homographic transform between the images, which can be backed out into likely robot motion between frames. The possible homographic transforms often include several incorrectly matched points, so RANSAC, a nonlinear filter good at outlier handling, is used to reject the outliers and select the best transform.



Using motion data, camera frames are drawn into the world map, shown to the right above. The map is a grayscale image, representing a probabilty function of traversablility, where black (0) represents non-traversable, gray (127) represents unknown areas and white(0) represents traversable ares. The map is built up as the robot moves, and slowly decays back to gray to prevent loop closure errors from building up. This map allows the robot to "remember" that it just passed an obstacle and needs to not turn sharply in order to avoid a collision. The robot is driven from the map, by a path planning algorithm.

Path Planning

The path planner uses a simple potential fields algorithm. The robot is attracted to the furthest navigable point in view, and repelled from obstacles.

Predicted Performance

With the limitations of our previous mechanical base resolved, and the addition of two new sensors, out team has high expectations for this year's performance. The slightly reduced base size should allow for easier navigation through the switchbacks, and the removal of the ball caster should solve the problem experienced with descending ramps.

The new LIDAR system will allow us to detect large smooth objects such as the ramp, and our existing vision algorithms should allow us to navigate the entire course without hitting an obstacle. The robot should also be able to detect and correct for slippage based on feedback from the wheel encoders and current sensors, which should further increase controllability.

Overall, the new mechanical base and electronics loadout should perform much better than any of our previous submissions to the IGVC competition.

BOM overview

Steel	$200.00
Aluminum	$200.00
Misc Mechanical	$300.00
4x Motors / Wheels	$1,400.00
4x OSMC Motor Controler	$680.00
Laptop	$800.00
2x LIDAR	$100.00
Camera	$580.00
Camera lens	$140.00
Motor Interface board	$30.00
Misc Cables	$50.00
4x Wheel Encoder	$355.00
6x Arduino	$185.00
2x Motor Interface Shield	$40.00
Estop	$80.00
Current Sensors	$20.00
Total	$5,160.00