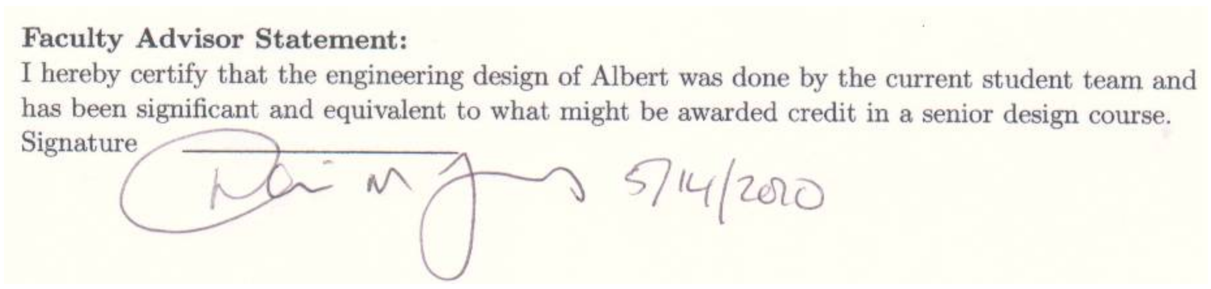
ALbert:
IGVC 2010 Fordham Rambots

Stephen Fox, Jennifer Feliz, Morgan Doles, and Marina Loginova

Robotics and Computer Vision Laboratory
Fordham University
441 E. Fordham Road, Bronx, NY

Faculty Advisor: Professor Damian M. Lyons

Director of the Fordham Robotics and Computer Vision Laboratory
Department of Computer and Information Science
Fordham University



Faculty Advisor Statement:
I hereby certify that the engineering design of Albert was done by the current student team and has been significant and equivalent to what might be awarded credit in a senior design course.
Signature [illegible] 5/14/2010

Contents

List of Figures

1 Team Overview

The Fordham Robotics and Computer Vision (FRCV) Laboratory 2010 IGVC team, Rambots, consists of four undergraduate students from two of Fordham University's campuses. The team is led by Stephen Fox, a junior mathematics major. Experienced as an administrator in the FRCV laboratory, he was the most suited to lead the team to the competition. The remainder of the team consists of two 3-2 Combined Plan Engineering Program students: Jenniffer Feliz, who is a sophomore majoring in Computer Science, and Morgan Doles, who is majoring in Engineering Physics. Marina Loginova, is a senior majoring in Math/Economics and is doing an accelerated masters degree in Computer Science.

Member	Contribution(hrs)
Stephen Fox	140
Morgan Doles	45
Jennifer Feliz	45
Marina Loginova	25

2 Design Process

As a small team, we were faced with the obvious challenege of a shortage of manpower. However, after analyzing the IGVC rules carefully, we determined we would be able to compete with one of our existing Pioneer 3 ActivMedia robots, though a small set of hardware modifications would be necessary in order to qualify at the competition:

1. Extension of the platform to meet the 3 ft. x 2 ft. size requirement
2. Addition of a payload area
3. Modification of the kill switch:
    - Move to rear center
    - Raise to 2 feet off the ground
    - Replace the button with one at least 1 in. diameter
    - Addition of a wireless kill switch with at least 50 ft. range

Weekly meetings were held between the team lead and at least one other member in order to maintain the pace and meet our goals. Our initial planning phase led us to design a very simple modular software architecture consisting of four primary modules, one for each obstacle-detecting sensor we planned to utilize, and one to synthesize the information and determine the direction to go based on certain goals. Since our team is spread across two campuses, sometimes meeting strictly every week was quite difficult. When this was the case, we held meetings online. With a modular architecture, it was possible to write and test each module individually, as well as test each in any combination. This proved to be of great advantage when we began testing our system.

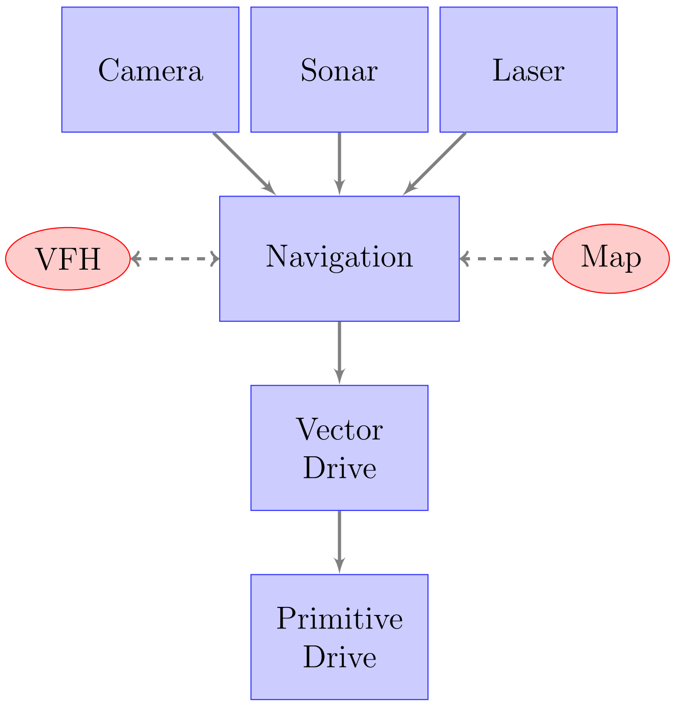


Figure 1: Modular Architecture of ALbert

3 System Integration Overview

3.1 On Board Computer

The computational power for ALbert is supplied by an on-board computer. The specifications include a 1.8 GHz Pentium M processor, 512 MB of RAM, a 2.5 inch 120 GB SATA HDD, and USB and serial ports. The on-board computer provides the platform for receiving sensor inputs and state information, as well as making navigational decisions. The robot is also equipped with a wireless b/g card, by which we connect to it through a secure shell by either an *ad-hoc* wireless connection broadcasted by the robot, as we must do in the field, or through a wireless router, as we do in the lab. We are also able to connect a monitor to the motherboard's VGA output in the event that ALbert fails to connect to the wireless network. Remote access to the robot is only used in the competition to initialize the navigational programs, though we can also use it for remote control.

3.2 Microcontroller

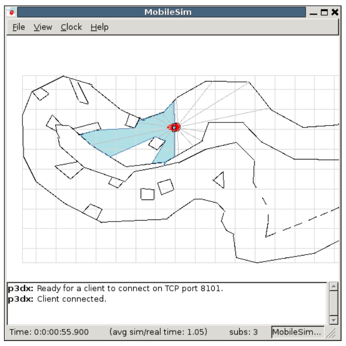
The software running on the on-board computer connects with the robot's microcontroller. This provides control for obstacle avoidance and navigation. This model of the Pioneer 3 was distributed with a 44.2368 MHz Renesas SH2 32-bit RISC microprocessor with 32K RAM and 128K FLASH memory. The microcontroller handles communication with many of the robot's components [1], including the motor encoders.

3.3 Operating System

We run the stable release of Debian Linux 5.0 with the 2.6.26 kernel because of its flexibility, functionality, and stability. It easily allows us to develop transportable and cross-platform code, as well as to utilize open-source libraries such as OpenCV 1.0.0. Furthermore, it provides a suitable interface for JAUS communication.

3.4 Advanced Robotics Interface for Applications SDK

ARIA is a C++ based open source development environment that is distributed by MobileRobots, Inc. with the Pioneer 3 robots. It provides a robust client-side interface for the robot which is released under the GNU public license. [1] We also created simulated course maps using Mapper3, and designed difficult trap scenarios to improve our navigational capabilities. In the spirit of the competition, we have made a concerted effort to minimize our dependence on the advanced features in the ARIA SDK. We have developed our own implementations of navigational algorithms, mapping capabilities, as well as algorithms for the manipulation of raw data collected from sensors.



```
p3dx: Ready for a client to connect on TCP port 8101.
p3dx: Client connected.
Time: 0:0:00:55.900     (avg sim/real time: 1.05)     subs: 3     MobileSim...
```

Figure 2: IGVC Simulation

4 Hardware Overview

4.1 Pioneer 3 Chassis

4.1.1 Motors

The robot is equipped with four motors with encoders that interface with the on-board microcontroller. The enconders allow Pioneer 3 robots to accurately estimate their state[1].

We tested the robot's ability to ascend a gradient of $4.5°$ with a 30 lb payload, a payload greater than required in the competition. We observed only minimal strain. We also tested the robot's ability to climb a $35°$ gradient with no payload, which it successfully completed. We predict that ALbert will be able to ascend the $15°$ inclines in the competition while bearing the 20 lb. payload.

4.1.2 Power

Pioneer 3 Robots carry three rechargeable 12 V DC batteries in their rear payload bay. These rechargeable batteries provide sufficient power to run the computer and peripherals, as well as the motors and the microcontroller. We have tested battery life to be between 20 and 25 minutes without a payload. We were initially concerned that a 20 lb. payload would reduce the battery life below a level suitable for the competition, but tests showed that the batteries would be more than sufficient for the five minute maximum of each run, so no major changes were necessary for power management.

4.1.3 Safety

ALbert, and all Pioneer 3 robots, are equipped with a hardware emergency stop button. Since this was too small for qualifications, and placed under 2 ft., we have modified it to meet the specifications. Furthermore, we have added a wireless E-stop for additional safety.

4.1.4 Compass

The TMC2 compass provides a useful and necessary heading for the ground vehicle. In the obstacle course, we rely on this, rather than the GPS during the obstacle course for heading detection. As it is a common component with Pioneer3 robot, we are also using the ARIA API to interface with it as well.

4.2 Range Sensors

4.2.1 Ultrasound Range Finders

ALbert is equipped with two sonar arrays of eight ultrasonic rangefinders. The sonar are powered by an on-board 5 V DC connection. We rely on these in conjunction with the laser for obstacle dection.

4.2.2 SICK Laser Range Finder

ALbert uses a high-precision LMS-200 laser range finder connected by serial to the on-board computer for obstacle detection. In order to detect potholes, we have inclined it at an angle of 9.1° to map the terrain ahead of it. This unique modification is how ALbert received its name, for we referred to him as Angled Laser, or AL, for some time. Since the laser is inclined at only a small angle, it can detect potholes at a distance of more than 2500 mm away while still performing local obstacle detection of varying height up to 367 mm, the height of the laser, within the 2.4 m range. Accordingly, it can also be used to improve the camera's detection of lanes by marking camera image points that do not appear to be in the ground plane.

4.2.3 Canon VCC50iR Camera

We mounted the Canon VCC50iR camera on top of the SICK laser range finder. It interfaces to the on-board computer through RS232. We use the ARIA API to control the pan, tilt, and zoom of the camera, as well as to grab images through a video server. In the IGVC, we use the camera solely for lane following.

5 Software

5.1 Navigation Module

5.1.1 Obstacle Avoidance: Vector Field Histogram

We use a modified version of Ulrich and Borenstein's Vector Field Histogram [2]. The heuristic part of the algorithm comes from their improved VFH+, but we did not find it necessary to upgrade to VFH*. For the sake of simplicity, we implemented the part of the algorithm to select an optimal direction while the vehicle is stationary. Simulated tests showed that the robot could navigate a mock course safely and effectively, and we improved this greatly after turning the parameters carefully. After such success in simulation, we tested our implementation with equal success in a simulated course environment. Although the vehicle must stop when it detects an obstacle in the course, the effectiveness of the navigational decisions seemed a worthwhile benefit. Since ALbert makes its navigational decisions while stationary, we have observed a very low proclivity for the robot to get turned around. In the rare event that it does, we are implementing a simple algorithm that prevents the robot from turning too much in a way that is no longer consistent with the minimum curvature specifications of the track.

5.1.2 System "Modes"

In order to improve our implementation of the VFH, we decided to give the robot a set of system states. Within the navigation module, we use the following categorizations:

1. **Safe, Simple Obstacle Avoidance** The system sets the radial threshold obstacle detection at 1200 mm in the VFH to navigate around areas where there are many perceived obstacles close together. It maintains a velocity under 500 mm/s from node to node, stopping to choose a new direction in about one second if an obstacle is detected in range.

2. **Look-Ahead Drive with Acceleration** If the system does not perceive any obstacles in a 1.2 m radius, the navigation module increases the threshold of the VFH. If this is clear, then the velocity is incremented proportionally to the clear distance up to a maximum of 700 mm/s. When the system detects an obstacle ahead, it again decreases the threshold and velocity until it returns back to "Safe" mode.

5.2 Sensor Modules for Environment Modeling

Each of the sensor modules is a separate C++ object with a common interface that stores a point cloud to standard template library list of a special three-dimensional point class. We have given each point a certainty value that is initially determined by the type of sensor from which the reading originates. Our implementation, which did not initially employ the certainty values set out by Ulrich and Borenstein, now uses these to improve its obstacle avoidance decisions.

5.2.1 Sonar Module

Since the sonar sensors emit a cone of ultrasound, our first approach was to model the worst case scenario of these sensors. Simulated tests quickly revealed the shortcomings of this simple model, which modeled an entire 30° region as occupied, with little consideration of the radius reading. We found we needed a better probabalistic model. In order to make this possible, we improved our point class to contain a certainty value within it. We now model the sonar readings in 30° arcs centered at each individual sonar. We assign the center 15° of the arc a higher certainty value than the tails. This, coupled with the VFH which uses the certainty values in creating the histogram, is much more effective.

5.2.2 Laser Module

Our LMS200 SICK laser range finder is mounted 200 mm ahead of the center of the robot on an aluminum frame that is manually adjustable to various inclines. We have inclined the plane of the laser at 9.1°; the laser intersects with base-plane of the robot 2.49m ahead of the robot's center. Each query to the range sensor returns 181 readings in the half-place of the direction of the laser. The laser can be used to map the terrain ahead of it by detecting aberrations, raised or depressed, in the ground plane that could be potentially dangerous. To transform the coordinates of the points returned by the laser (returned as radii in polar coordinates) readings in 3D space relative to the coordinate frame of the robot, we project the laser's plane on the plane parallel to the ground plane of the robot that passes through the laser's center, then translate it appropriately to the robot's origin. This can be seen in the following equation, where $(X_T, 0, Z_T, 1)^T$ is the translation vector from the center of the laser's origin to the robot's origin, and $(x, y, 0, 1)^T$ is the coordinate vector of each particular range-reading with respect to the laser's origin.

$$\begin{pmatrix} \cos\theta & 0 & 0 & X_T \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & 0 & Z_T \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} x\cos\theta + X_T \\ y \\ Z_T - x\sin\theta \\ 1 \end{pmatrix} \tag{1}$$

Although we do not keep a cumulative point cloud of the world view, we do maintain a local point cloud, which allows ALbert to avoid both potholes and raised obstacles. Tests employing our navigation algorithm with only the laser have shown the calculated navigational decisions to be highly reliable, and the location of the points to be highly accurate. The modular nature of our software coupled with the precision of this particular range device lends itself to future developments upon our generalized solution that will easily allow us to motorize the variation of the incline of the laser and create an accurate three-dimensional model of the environment.

5.2.3 Vision Module

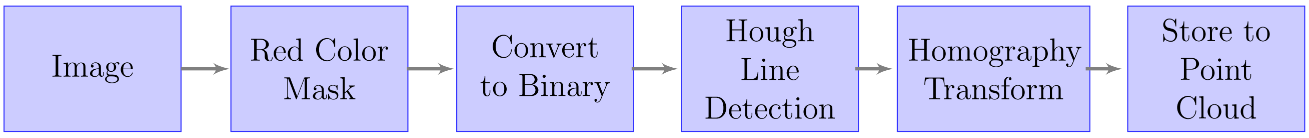
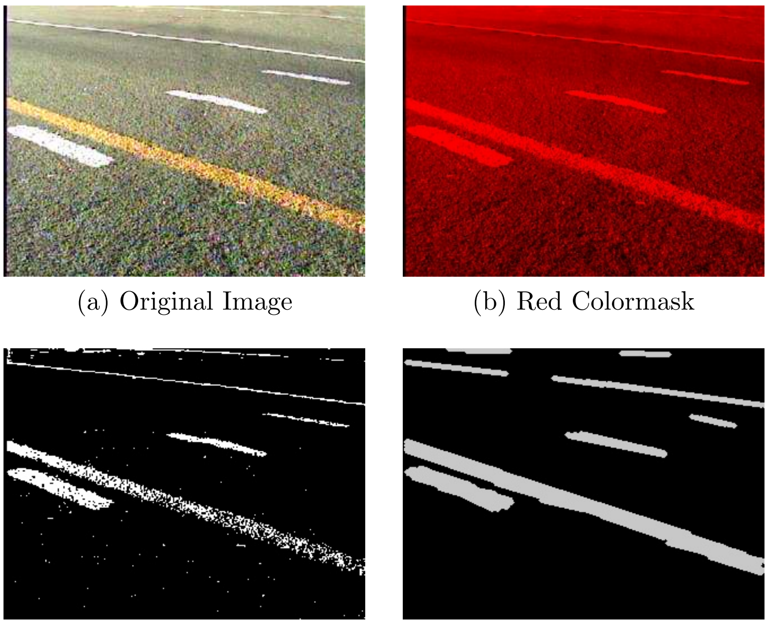


Figure 3: Vision Pipeline

Our vision module is used for lane detection. It interfaces with an on-board video server, SAV 0.9, to grab images from the Canon VCC50iR camera. Using native functions in

OpenCV 1.0.0, we filter the image through a red colormask, then convert it to binary at a threshold of 68, and search for lines using Hough Line Detection in OpenCV. The endpoints of the lines are then stored, and transformed using one of four Homography Matrix Transformations to calculate the coordinates of lines in relation to the robot. We found these settings very effective for detecting both white and yellow lines in green grass.



(a) Original Image (b) Red Colormask

(c) Binary Image (d) Line-Detected Image

Figure 4: Detection of White and Yellow Lines

To aid in the simplification of the vision module, we manually calibrated four useful homography transforms to be used by the software to transform the coordinates of the images to the coordinates of the ground plane, with origin at the robot's center. Each homography, which is stored in the software, is associated with a distinct set of pan, tilt and zoom settings, as well as a state setting for the vision module. They can be summarized as follows:

1. **Local, Center** Used for close detection of lanes when the robot slows or stops to calculate a new direction
2. **Local, Left** Used to scan the left view for nearby lane detection
3. **Local, Right** Used to scan the left view for nearby lane detection
4. **Center, Ahead** Used when the navigation module is in "Look-ahead mode" for more distant lane dection

6 JAUS Compliance

The highly modular design of the ALbert architecture affords a great deal of integration and compliance with the JAUS (Joint Architecture for Unmanned Systems) specification[3]. JAUS has help us to avoid incompatibilities between the architecture's various working components and provides for convenient future extension of our software platform. ALbert is compliant with the JAUS competition requirements.

7 Special Thanks

Special thanks to Dr. Lyons without whose continuing support, our entry in the competition would not have been possible. We would like to extend a special thanks to Eric Jones, a Computer Science graduate student who provided us with much encouragement and support, particularly in understanding the Joint Architecture for Unmanned Systems.

References

[1] Mobile Robots Inc., 19 Columbia Drive, Amherst, NH 03031, *Pioneer 3 Operations Manual*, version 5 ed., July 2007.

[2] I. Ulrich and J. Borenstein, "Vfh+: Reliable obstacle avoidance for fast mobile robots," 1998.

[3] J. W. Group, "The joint architecture for unmanned systems reference architecture specification," July 2007.