Software Design for an Autonomous Ground Vehicle for the 13th Annual Intelligent Ground Vehicle Competition

Tim Roberts[a], Daniel Barber[b], Brian C. Becker[c], Fernando Gonzalez[d]
Robotics Laboratory at the University of Central Florida
3100 Technology Pkwy, Orlando, FL, USA, 32826

ABSTRACT

This paper presents the vision and path planning software design of a totally autonomous vehicle built to compete in the 13th Intelligent Ground Vehicle Competition, IGVC. The vehicle, Calculon, is based on a powered wheelchair and uses a variety of sensors for its navigation and obstacle avoidance including a 3CCD Sony color camera, an outdoor laser range finder, a DGPS, 2 wheel encoders and a solid state compass. The modules forming the core vision system include: filters, color classifier, image segments, and line finder. Our color classifier is based on a modified implementation of an adaptive Gaussian color model similar to those used in some skin detection algorithms. A combination of various image enhancing filters and the color classifier allow for the isolation of possible obstacles within the image. After filtering the image for areas of high brightness and contrast, the line finder performs a Hough Transform to find lines in the image. Our path planning is accomplished using a variety of known and custom algorithms in combination including a modified road map method, a Rapidly Exploring Trees method and a Gaussian Potential Field's method. This paper will present the software design and methods of our autonomous vehicle focusing mainly on the 2 most difficult components, the vision and path planning.

Keywords: Intelligent Ground Vehicle Competition, Computer Vision, Path Planning, Autonomous Vehicles

1. INTRODUCTION

Figure 1: Calculon

The Robotics Laboratory at the University of Central Florida proudly presents a software design for an intelligent ground vehicle. The vehicle named Calculon was designed and built to compete in the 13th annual Intelligent Ground Vehicle Competition, IGVC [1], held at the Traverse City Resort and Spa from June 10th – 13th 2005. The IGVC is an international college level competition where teams from Japan, Canada, and all over the United States compete in three challenges: the Design Challenge, the Navigation Challenge, and the Autonomous Challenge. The Design Challenge consists of teams submitting a fifteen page technical document summarizing all of the systems created and the design process used by the teams, a power point presentation highlighting the key systems and components used to build the vehicle and finally an overall inspection of the vehicle. The Navigation Challenge consists of an open course where waypoints are marked and the vehicles have to navigate to nine waypoints within six minutes. The waypoints are given to the teams in latitude and longitude positioning and in meters. The teams are not given the location of any of the obstacles, though, that are strategically placed throughout the field to trap vehicles. The Autonomous Challenge consists of a course that involves the vehicle staying between two painted lines. The vehicles have to follow the lane over a ramp, through a sand pit, while avoiding obstacles at all times. Calculon was designed and built with these challenges in mind. The vehicle won third place in the Design Challenge, eighth place in the Navigation Challenge, and twelfth place in the Autonomous Challenge and earned the team fourth place overall. In this paper, we present the software systems that were designed and created for the vehicle. The two most challenging aspects of the competition, machine vision and path planning, are discussed in detail in this paper.

1.1 Computer vision requirements

For a vehicle to successfully navigate the Autonomous Challenge course it must be able to detect painted road lines on grass, concrete, or other surfaces. This requires the use of machine vision because the vehicle is not allowed to cross over or drive on top of a single line to complete the course. The system must be able to detect both lines on the left and right side so that the center of the lane can be located. Finding other obstacles on the course such as barricaded and orange barrels can be done without the use of machine vision, but the system described in this paper is also capable of this task since these obstacles are common on both the Autonomous Challenge and Navigation Challenge courses. Image filtering and line finding techniques are used to make the system robust enough to operate in the outdoor and varied lighting conditions representative of the IGVC.

1.2 Path planning requirements

The two different courses described for the vehicle to traverse require different approaches for decision making at the IGVC. Therefore the path planning system has been separated into two parts, one for Autonomous Challenge the other for Navigation Challenge. The system designed for the autonomous challenge must be able to plot a course that stays in between two lines and avoids obstacles while not having a destination point to travel to. For the Navigation Challenge, the course plotted has a waypoint destination to reach, so the software does not need to worry about road markings, only a safe course to the individual waypoints that is short and safe.

2. DESIGN PROCESS

Through experience and researching robotic systems design and implementation, we found that a strict and unyielding design processes such as The Design Life Cycle or the Spiral Model were not well suited to our group, lab structure or our approach to the Intelligent Ground Vehicle Competition. Instead, we have followed an incremental design and development process, which allows for more flexibility when needed as well as for parallel development from various sub-groups which, was essential to our team. To prevent integration problems interfaces to different components were well defined in advance, so that the different teams could develop there software independently of each other. The final incremental model we chose followed the ‘Validation V’ [2].

2.1 Validation V and W

Our incremental design flow followed the ‘Validation V’. This design flow follows the same path as many other design paradigms, namely requirements stage, system design, construction, testing and completion. With just this simple flow, it is hard to see how this paradigm is incremental. However, by connecting the ‘Validation Vs’ together we see that we get a ‘W’ formation which represents the synchronization of multiple teams working together. With the use standardized software interfaces and protocols, the teams were able to attain modularity to the entire project that allowed many of the components to be built or worked on simultaneously and in parallel. The ‘W’ paradigm works exceedingly well with this level of modularity. During project pre-planning, we found this design paradigm to be most realistic for our group in terms of the actual implementation. Many others seemed as though they may work well in theory, but for a group of our small size and project such as an IGVC vehicle, the ‘Validation V’ and ‘W’ concepts were appropriate.

3. COMPUTER VISION

In the design of the vision system for Calculon, there were two main goals: identification of road lines and common roadside barricades. For the vehicle to navigate both courses at the IGVC these tasks had to be achieved. In previous years machine learning and statistical training of recorded data was the primary method of creating our machine vision systems. This method alone did not achieve the results desired and required too much training time and a lot of work to make system changes. To combat this issue, our design plan for machine vision called for the creation of tools to better facilitate building a vision system that could be quickly changed to incorporate new methods of obstacle detection. The end result was the development of a rapid prototyping tool called the Discover Vision Engine. The Discover Vision Engine makes use of image filters and line finding algorithms to create a vision system capable of performing the tasks needed at the IGVC.

3.1 Discover Vision

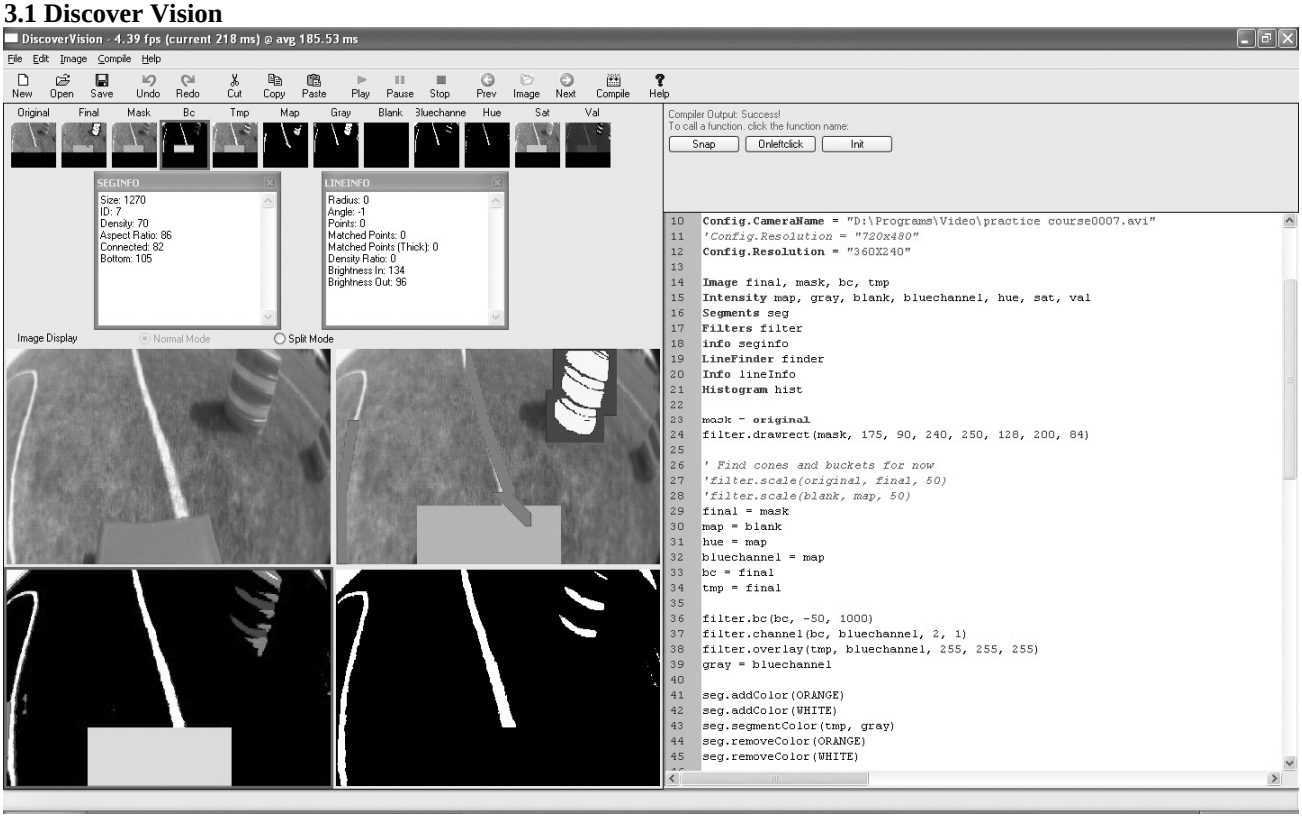

Figure 2: Discover Vision GUI

The Discover Vision Engine is modeled after existing tools such as MatLab, except aiming to present an interface capable of easily manipulating images instead of algorithm development. The Graphical User Interface, Figure 2, has two main components: a code editor and an image display. The code editor incorporates syntax highlighting and line numbers. The image displays uses two modes: a single pane for showing the full details of an image, or four panes, each containing a separate scaled down image, allowing multiple filtered images to be viewed simultaneously in real time. This is extremely useful to determine how affective a specific or combination of filters can be. The Discover Vision Engine can load and process images from a number of sources, including image files, video cameras, and saved videos.

A simple scripting language developed for the tool is used to interface between the GUI and the C++ backend. The scripting language supports multiple classes and member functions and is easily extendible. The scripting language supports overloaded functions as C++ does, and also offers the ability to create user defined functions that can be called with a click of a button. To increase speed, the script is compiled into byte-code for the engine interpreter to run. Each 'recompile' happens without any interruption to the programs execution, meaning you can immediately see the difference on the images, even in the middle of processing a video stream.

A user can open, save, and create new scripts allowing different aspects of the vision system to be tested individually. Furthermore, the compiled scripts can be exported as well. Because of the modular design, the engine interpreter can be separated from the GUI and used in another framework without difficulty. Therefore the interpreter is included in the final obstacle detection system, and never needs to be changed to incorporate a different sequence of image processing steps. The developer only needs to create a different script using the Discover Vision program, and the interpreter will use it in the final application. This makes changing a vision system for different tasks very simple to do because the final application does not need to be modified, saving valuable time.

3.2 Obstacle detection

To support the vision system, a 3 CCD Sony HandyCam DV camcorder positioned at the front of the vehicle, is angled toward the ground. It captures high-resolution DV video (720x480) and streams the frames to the computer at a rate of 30 frames per second. The vision system utilizes the Microsoft DirectX interfaces to grab frames from the video for processing.

The modules forming the core vision system include: filters, color classifier, image segments, and line finder. The color classification system used by Calculon is an implementation of an adaptive Gaussian color model similar to those used in some skin detection algorithms [3]. These skin detection algorithms have been modified to meet the needs of color classification in the fixed and predictable environment of the robotic vehicle. To classify the incoming pixels with respect to color, the system must first be trained on a set of sample images and masks. Each mask is custom made to train for a specific color from specific sample images which represent typical images the robot may encounter.

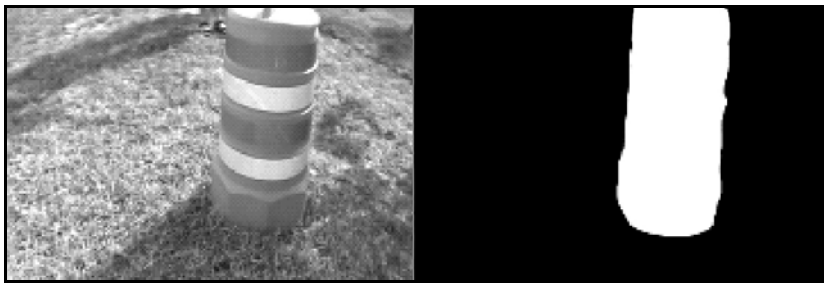

Figure 3: Example training images

In training, two images are opened, a sample image usually taken with the Sony DV camera, and a custom black and white mask. This mask was designed to capture whichever specific color we are trying to train. Some were created specifically to train the system to detect grass, others to train it to detect orange barricades or white buckets. The black and white masks use white to represent training pixels and black to represent non-training pixels. Different from most of the skin detection algorithms this is based on, the system exploits the fact that the images being trained on are not only representative of the colors we are trying to detect, but also representative of the typical images we can expect during actual vehicle operation. Because of this, the algorithm is training not only to detect the colors it wants, but to also specifically not detect the colors it does not want. This is a subtle but powerful difference.

The training algorithm starts by creating an RGB color cube data structure which can be implemented with either 24 bit color (16.7million colors) or 16 bit color (65 thousand colors). Each cell in this data structure is a floating point value which starts at 0.0. The next step in preparation of training is to create two Gaussian spheres. The first is based on a value, sigma-large, and the second is smaller, based on sigma-small. Because of the larger sigma the first also has a higher floating point value in the center. Once the training image and its mask are loaded for training, the system goes through each pixel in the training image. If the corresponding pixel in the mask is white, a value of the large Gaussian sphere are added to the color cube data structure, centering the sphere on the color cube cell corresponding to the RGB value of the pixel currently being trained on in the image. Similarly, if a pixel in the mask is black (a non training pixel); we subtract the values of the small Gaussian sphere from the color cube data structure, again centering on the cell corresponding the RGB value of the training pixel. The use of a larger and higher valued Gaussian sphere for positive training gives the algorithm an overall bias towards detecting our training color, as opposed to not detecting all other colors.

Once training images and masks have gone through the algorithm, live detection can be done incredibly fast. To detect the specified color in a new test image, the algorithm simply has to go through each pixel in the image, map that pixel's color into the color cube data structure and use a threshold value to decide whether or not a color is to be classified or not. Threshold values are typically low but positive values since the Gaussian cloud created around the specified color in the color cube data structure should have positive values while all other colors in the images should have negative values.

Overall the algorithm sacrifices a slow and long offline running time for training which is worse case O(nmp3) where n is number of image rows, m is number of image columns and p is the large sigma value chosen, in order to achieve very fast worse case O(nm) online color classification.

A combination of various image enhancing filters and the color classifier allow for the isolation of possible obstacles within the image. This data is passed to the image segments module to extract these possible obstacles into segments. Each segment is statistically analyzed for color, texture, size, shape, orientation, and other features to determine first whether this segment is an obstacle and secondly what type of obstacle the segment represents. This method is used primarily for the identification of larger objects such as construction cones, buckets, and ramps which are typical of the IGVC.

3.3 Line detection

Line detection is performed by applying a Hough Transform [4] across a segmented filtered image. The filters used for obstacle detection are done before line detection takes place. These filters serve to remove noise and other obstacles which do not represent road lines. Filters used include the color classification algorithm, brightness and contrast adjustment, and segmentation filter. The resulting image is decomposed into a matrix and the Hough transform is applied to each cell. Once line segments are identified, an additional step is taken to join adjacent lines together. This is done to fill in gaps between lines which are typical of the Autonomous Challenge course at the IGVC. By breaking down the image into a matrix, it is possible to approximate a curved road line using the Hough transform. This creates a more accurate model of the environment around the vehicle, and makes for more affective path planning. Figure 3 shows an example of this process.

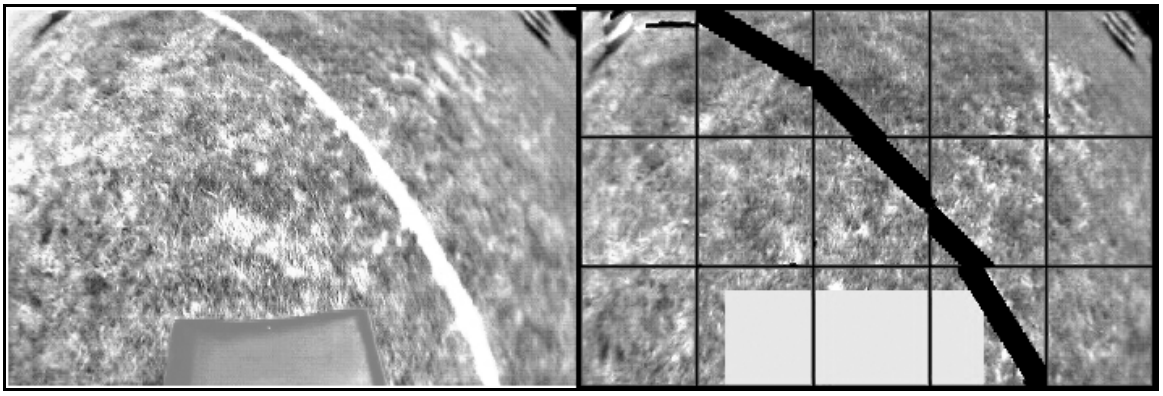

Figure 4: Example of Hough Transform in segmented image

4. PATH PLANNING

The Intelligent Ground Vehicle Competition provides two challenges for the vehicle to perform in, the Autonomous Challenge and Navigation Challenge. The Autonomous Challenge and the Navigation Challenge are two courses that are built to test the vehicle's ability to avoid obstacles while performing a given task. For the Autonomous Challenge the goal is to stay within two boundary lines on a 500 foot course which includes typical road barricades and barrels and varied terrain such as a ramp and sand pit in less than 5 minutes. In the Navigation Challenge the task is to drive to nine GPS waypoints in less than six minutes while avoiding obstacles that are designed to trap or block the vehicles path.

4.1 Autonomous Challenge

For the Autonomous Challenge, the path planning system relies entirely on a local map system. The robot is able to make decisions, plan the path and traverse the course entirely with only local information and a small amount of position history. The actual algorithms used in path planning include a Gaussian Potential Fields [5, 6] method and a depth first search. Within the Gaussian search space higher values indicate a lower probability of being on the path. Obstacles and line information from the Laser Range Finder and digital camera are combined into a local map which is then converted to Gaussian space. All obstacles including lines are treated the same in this newly formed search space.

4.1.1 Gaussian search space

Once the search space is created a depth first search, Figure 4, is performed to find a desired path. The parameters that can be changed for this search are: the number of branches, branch length, obstacle threshold, and angle

between branches. The total depth is variable so that the software will try different possible depths to find the path which will travel the furthest. By find the branch that covers the longest distance it is possible to find paths which do not lead into dead ends and traps. Branch trimming is done by performing radial checks at each branch endpoint and path width checks along the branch. This is done to ensure that that branch does not cross over an obstacle, and that at each point in the path the vehicle will be able to turn and change direction successfully. The total number of possible solutions is therefore only limited by the number of obstacles in the local map, local map size, and the algorithm parameters. Typically the search space produces three hundred possible solutions for the vehicle take.

4.1.2 Choosing optimal solution

To decide which of the possible safe solutions is best for the vehicle, information about the path and the vehicles travel history are taken into account. Since all the possible solutions are safe, the safest and longest path should be chosen, but this path must not differ too greatly from the direction the vehicle has already been traveling in and it should not travel to places already visited. Paths which maintain the current direction of the vehicle are optimal because it prevents the situation where the vehicle may turn around and start traveling the course backwards. Also, a path which travels the greatest distance will in most cases not go into a trap or dead end. To figure out what direction the vehicle has been traveling, DGPS waypoints of the vehicles path are recorded periodically. Using the most recent information it is possible to calculate the vehicles previous trajectory using linear regression. In addition to comparing to the vehicle's previous heading, each possible solution is compared to the recent history to ensure that it is not going back to points that have already been traveled to. Using this data, in the unlikely event the vehicle travels to a dead end, it is possible to turn around and escape this trap. Once the vehicle has escaped the trap, the next path chosen will be the one that maintains the previous headings, but also does not travel back to areas already visited resulting in a continuation of the course.

4.2 Navigation Challenge

While the Autonomous Challenge needed only information on its local surroundings in order to properly steer through the course successfully, the Navigation Challenge requires global information in order to most efficiently reach each GPS waypoint without getting caught in a trap by obstacles, navigating off course or mistakenly returning to the start point. To record this global information a world map is created dynamically. To build this world map, the local map is copied and added on a set interval. As the vehicle travels the course, it will slowly build a map of all obstacles and waypoints making path planning more reliable.

4.2.1 Rapidly expanding trees

Using the constructed world map, a modified version of the Rapidly Exploring Random Trees method, RERT [7] and Figure 5, is used. Instead of choosing random branches within the search space, the algorithm performs a 360 degree scan around the vehicle and chooses the branch endpoint that travels closest to the destination waypoint. This process is then repeated until a path that reaches the desired waypoint is found. To prevent choosing points which travel over or into obstacles, branch checks similar to those used in the Autonomous Challenge are performed. These checks make sure that the vehicle is able to move freely at any point along the chosen path to the waypoint. This method results in an extremely fast search technique which finds the shortest and safest path to the desired GPS waypoint.

4.2.2 Artificial boundaries

Competition rules require that the vehicle does not travel out of specified boundary lines for the course and that it does not return to the starting point until all other waypoints have been reached. For example, if the vehicle reaches waypoint number 1, and then plots a course to the second waypoint that travels through the start box, as soon as the vehicle reaches the start box, the run will be over. To prevent this situation artificial boundary lines are created within the world map. When the vehicle is started in the navigation mode, it records endpoints of an artificial box around the starting point. Once the vehicle leaves this starting area, artificial boundary lines are added to the world map. This prevents the path planning algorithm from plotting a course through the starting box. Once all other waypoints have been reached, this artificial box is removed, and the path planner can plot a course back to the start. The same technique is used for creating course boundary lines. The positions of the boundaries are given to the user at the competition, and can be placed into a settings file. This file is then opened at program start up, and the boundary lines are drawn into the world map preventing the vehicle from plotting a course off the competition field.

4.3.3 Limitations

Although the modified version of the RERT algorithm produces a safe solution quickly, it does not produce the most optimal path. Through simulation and testing it was found the algorithm fails to accommodate traversing to destinations that are surrounded by large walls or joined obstacles when information about the wall is incomplete. The result of this caused the vehicle to travel back and forth to try and get to a waypoint until the world map was able to model the complete surroundings. For future implementations of this method a depth first search and a limited scan area will account for this. By using a depth first search with multiple branches analyzed, the shortest path to the waypoint can be found. Also, by limiting the scan sweep to 180 instead of 360 degrees, the vehicle will not be able to continuously turn around and is forced to continue moving forward preventing oscillation. These additions to the method will be explored in future version of the path planning software.

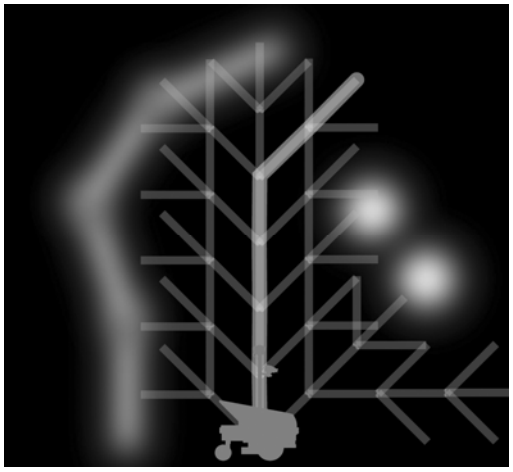

Figure 5: Potential fields with depth first search

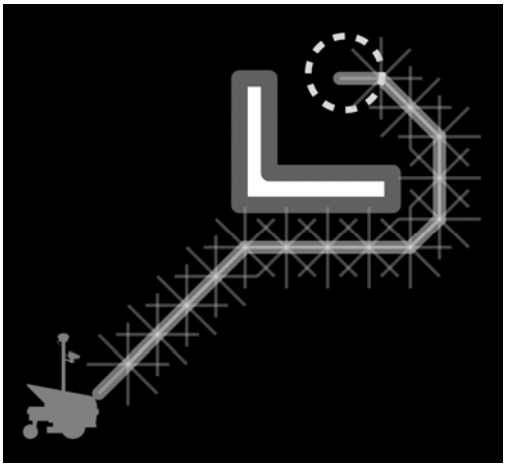

Figure 6: Modified RERT

5. CONCLUSION

In comparison to some of our competitor's, the approach to machine vision and path planning presented here leads to a much more robust system. A very common approach used by our competitors is to have totally reactive decision making algorithms for path planning. That is, it only considers the environment immediately in front of it. These reactive systems run faster allowing the vehicle to react quickly to immediate threats, however it does not plan ahead sufficiently to avoid traps. If a vehicle without proper planning arrived in a dead end, it would not be able to reverse and choose an alternative course. This approach does not result in intelligent decision making. Although this may have lead to good results in the previous competitions, it will not be able to account for real world situations or a competition made more difficult in future years. Future goals of this and other projects is to combine the planning and reactive systems together to create a reactive system that still takes into account the higher level goals handled through our methods.

In addition to a robust path planning system, the creation of a tool for developing machine vision systems provides an advantage over competitors. Through the Discover Vision Program, different algorithms and image processing steps can be quickly tested and proven to work in a changing environment quickly. Without this ability, making changes and testing a machine vision system a tedious and time consuming process. The option to expand upon this tool by incorporating new techniques very quickly also makes enhancing the overall system a simple task. Our lab will continue to build upon the successes we've had using these methods and improve upon them for future competitions.

REFERENCES

[1] http://www.igvc.org – Intelligent Ground Vehicle Competition, Rules and Regulations
[2] Alistair Cockburn, Using "V-W" Staging to Clarify Spiral Development, Human and Technology, 1995
[3] Yan, N. Ahuja, Gaussian Mixture Modeling of Human Skin Color and Its Applications in Image and Video Databases, SPIE/EI&T Storage and Retrieval for Image and Video Databases, 1999
[4] Richard O. Duda and Peter E. Hart, Use of the Hough Transform to Detect Lines and Curves in Pictures, Communications of ACM, 1972
[5] Koren, Yoram and Borenstein, Johann, Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation, Kluwer Academic Publishers, 1972
[6] Jean Claude Latombe, Robot Motion Planning, Kluwer Academic Publishers, 1991.
[7] S.M. LaValle, Rapidly-Exploring Random Trees: A new tool for path planning, Computer Science Dept., Iowa State University, 1998

[a] timr@mail.ucf.edu; phone 1 407 882 0293; http://robotics.ucf.edu
[b] dbarber@ist.ucf.edu; phone 1 407 882 0293; http://robotics.ucf.edu
[c] brian@briancbecker.com; phone 1 407 882 0293; http://robotics.ucf.edu
[d] fgonzale@pegasus.cc.ucf.edu; phone 1 407 823 3987; http://robotics.ucf.edu