Multiple camera, laser rangefinder, and encoder data fusion for navigation of a differentially steered 3-wheeled autonomous vehicle

David C. Conner*, Philip R. Kedrowski, Charles F. Reinholtz
Department of Mechanical Engineering, Virginia Tech, Blacksburg, VA 24060

ABSTRACT

Virginia Tech is currently developing a new autonomous vehicle as a research platform. This vehicle is being used to investigate techniques in autonomous landmine/UXO detection. In addition, it was entered in the 2000 Intelligent Ground Vehicle Competition. This vehicle senses its surroundings using two (non-stereo) color CCD cameras, a SICK laser range finder, and wheel encoders. The cameras give a color representation of the area in front of the vehicle; while the laser range finder provides range data for obstacles in a 180-degree arc in front of the vehicle. Encoder feedback is used to determine position and velocity of the vehicle. This paper presents the techniques used to fuse this diverse and asynchronous data into a useful representation. The software architecture, which allows the various sensor fusion methods to be tested in a modular fashion, is also presented, along with the results from field-testing.

Keywords: Sensor fusion, vector field histogram, autonomous vehicles, mobile robots

1. INTRODUCTION

The Autonomous Vehicle Team (AVT) at Virginia Tech has a history of building vehicles for entry into the annual Intelligent Ground Vehicle Competition(IGVC), formerly known as the International Ground Robotics Competition. [1] The team has begun to branch out and explore other application areas, such as landmine/UXO detection, for its vehicles. Preliminary work in this field required the development of a vehicle capable of carrying currently available detectors across uneven and cluttered ground terrain. The design of this new vehicle was undertaken by the AVT during the 1999-2000 academic year. The main focus of this first year's development was upon the base vehicle, and supporting sensors and navigation algorithms. This new vehicle, shown in Figure 1, is called Navigator and was entered into the 8th annual IGVC, held July 8-10, 2000, in Orlando, Florida. [2]

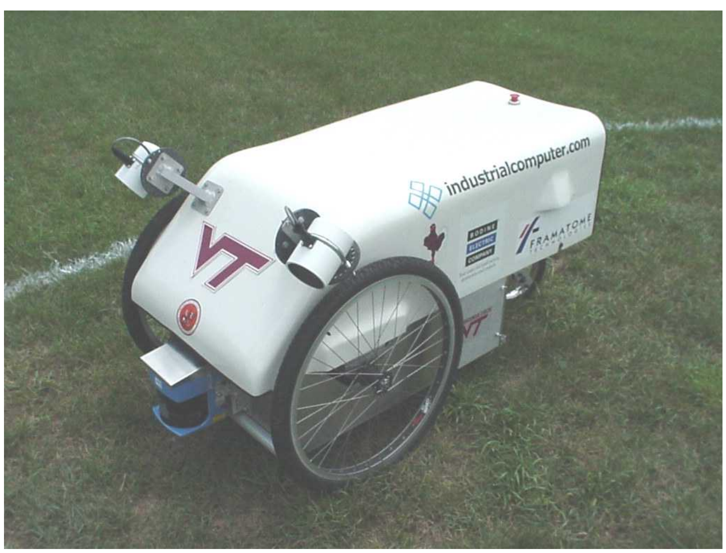

Figure 1. Navigator – a 3-wheeled differentially driven autonomous vehicle

*Correspondence: Email: dcconner@cmu.edu; WWW: http://www.cs.cmu.edu/~dcconner;
Address: David Conner, Robotics Institute, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213
Telephone: 412.268.8809 Fax: 412.268.5571

2. VEHICLE DESIGN

Navigator is a 3-wheeled differentially driven vehicle, with two large front wheels to provide both drive and steering. A third wheel in the rear of the vehicle acts as a caster and is free to rotate about its vertical axis. This configuration allows for the capability of zero radius turns and leads to a nimble vehicle that can steer without causing damage to the ground like tracked or multi-wheeled skid steering. The zero radius turn allows the vehicle to operate in cluttered environments where a similarly sized steered vehicle could not operate. The 26-inch drive wheels are driven by 2 24-Volt DC gearmotors rated at 75 rpm. This gives the vehicle a top speed of 5.8 miles per hour. Two 24-volt lead acid battery packs provide power to the vehicle. One battery powers the electronics, while the other powers the motors.

Navigator uses two separate computer systems for vehicle navigation and control. An industrial programmable logic controller (PLC) executes the main motor control, while a dual Pentium III computer (PC) handles the higher-level sensor fusion and navigation tasks, as well as providing the user interface. Both the PLC and the PC use 24-Volt DC power supplies, supplied by the batteries. The PLC provides the output signal to the motor drive amplifiers and reads the encoders to provide velocity and position feedback. Additionally, the PLC provides both analog and digital inputs and outputs. The inputs are used to read switch positions to determine the vehicle mode and motor current. The outputs are used to drive several status LED's and power relays and provide motor amplifier control commands. The PC includes the video frame grabbers and Ethernet card. By using a powerful computer system, several vision processing strategies could be tested in software. In a production vehicle, the best operations could be handled in hardware, thereby limiting the need for such a computer.

The vehicle operates using a hierarchical control scheme: the PLC handles low level motor controls, while the PC handles high-level navigation decisions. By using this structured control strategy, each piece of hardware is able to handle those tasks for which it is best suited. The PC and PLC communicate across the Ethernet to exchange control and status information. The PLC includes interlocks that stop the vehicle if communications are lost.

Navigation Manager, the software developed at Virginia Tech, fuses the diverse input data and makes the higher-level navigation decisions. Later sections of this paper discuss this software in detail. For now, assume that once the navigation decision is made, the steering commands are passed to the PLC via the Ethernet connection. The steering commands consist of a speed command and a turning rate command.[2,3] Together these two commands specify a desired arc for the vehicle to travel.[2] Additionally, the Navigation Manager software handles such user interface tasks as starting and stopping, along with data logging and real time data plotting.

The lower level control functions are handled by the PLC. The PLC includes hardware modules for interfacing with the industry standard drive amplifiers. The actual motor control is implemented in 'C' code in the PLC's central processing unit (CPU). The PLC receives speed and turning rate commands from the PC, and the control code converts these commands into the desired wheel speeds required for steer the desired path.[2, 3] The relationship between the speed, turning rate, and the vehicle wheel rotations is specified by the vehicle kinematics.[2, 3] Once the desired wheel speeds are calculated, a discrete implementation of the PID algorithm handles the actual motor control based on feedback from the encoders. The implementation uses a non-standard form of the discrete PID to avoid some problems discovered with tuning the algorithm under saturation conditions.[2]

3. SENSORS AND SENSOR FUSION

In order for the vehicle to act in manner that appears intelligent, the vehicle must be capable of sensing changes to its environment.[2] The ability to sense these changes is given by the vehicle's sensors, while its ability to react well is limited by its ability to use the data from its various sensors in an coherent manner. This section discusses the sensors used on Navigator and the method used to fuse the sensor data.

3.1 Charge Coupled Device cameras and Frame Grabbers

Two color Charge Coupled Device (CCD) cameras are used for image acquisition. The cameras are configured to give an overlapping view of the ground in front of the vehicle, as shown in Figure 2. The images are captured using two frame grabbers and processed using the Intel Image Processing Library (IPL) functions. The IPL functions, which are optimized for the Intel processors used on the vehicle, offer increased performance for basic image processing operations. The use of the dual processor computer allows the images to be processed in parallel, which increases the overall frame rate of the vision system.[2]

Once the images are acquired, they are processed in order to distinguish relevant features—such as course boundary lines, debris, and obstacles—from the background image. Once the individual images are processed, they are transformed to the ground reference plane, and combined into a composite image. This composite image is further processed to remove spurious bright pixels due to noise or glare.[2] This processing step requires some tuning based on the lighting conditions and the size of the relevant objects in the image.

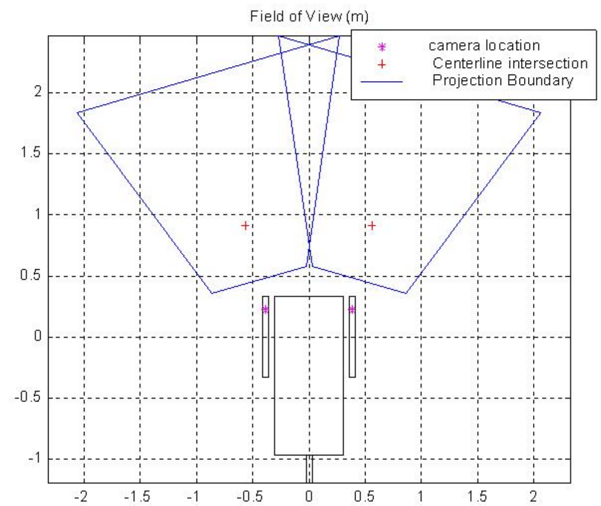

Figure 2. Camera projections relative to the vehicle

3.2 Laser Rangefinder

To supplement the computer vision system, Navigator used an LMS-200 laser rangefinder, manufactured by SICK. The laser rangefinder returns 361 data points in a 180-degree arc, giving a 0.5^{o} resolution. The laser rangefinder has a range of up to 30 meters; however, the look-ahead distance for Navigator was set at 5-meters. Figure 3 shows a graphical representation of the data.

The system detects obstacles based on the first derivative approximation of the range data. This enables specific obstacles to be identified. For tasks such as follow-the-leader, the obstacles are tracked according to the predicted position of the selected target obstacle.

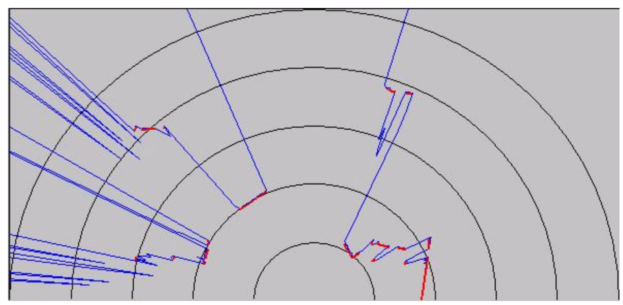

Figure 3. Laser rangefinder data (range in meters)

3.3 Encoders

Navigator includes wheel encoders on both of the drive wheels to provide velocity and position feedback. The PLC interfaces with the encoders and converts the encoder pulses into engineering units. The encoder feedback is used in the PLC control algorithm to regulate wheel speeds, and therefore the overall speed and turning rate of the vehicle.[2,3] Because the navigation system, to be described later, is reactive and is not concerned with position , the navigation system does not use the encoder feedback in its current configuration. The encoder feedback is used in position estimation for controller testing and tuning. The position estimate is based on dead reckoning. For applications that require position estimates more accurate than can be obtained by dead reckoning alone, it has been proposed to instrument the rear caster wheel as well.[3]

3.4 Evidence Grids

The concept of an evidence grid, which has also gone by the terms certainty grid, occupancy grid, or histogram grid, is a proven technique for fusing sensor data.[5, 6] The concept uses a tessellated view of the world, which is mapped to a grid. Each cell in the grid contains a value measuring the probability that the area represented by the cell exhibits some property. Typically this is used to determine if an area is occupied or free, based on sensor readings. The more modern implementations use probabilistic measurements to determine the cells values.[5] Furthermore the cells are manipulated according to probabilistic techniques. A common use of the evidence grid is for mapping an unknown space using the sensor measurements during vehicle exploration.

Navigator uses a technique similar to an evidence grid. Although not based in probability theory, the grid does represent a tessellated view of the ground in front of the vehicle. Once the camera images have been processed, transformed, and combined into a single composite image, the composite image is decimated to give a much coarser view of the world. This process is shown below in Figure 4. Because the image is a pixellated view of the world, the decimated result gives a tessellated view of the world. Although the resulting grid is still stored as an image, it functions as an evidence grid. Each pixel intensity in the decimated image relates to the expectation of a line or obstacle in the area represented by the pixel. As configured, Navigator uses a decimated image where each pixel represents an area upon the ground approximately 6 cm square.

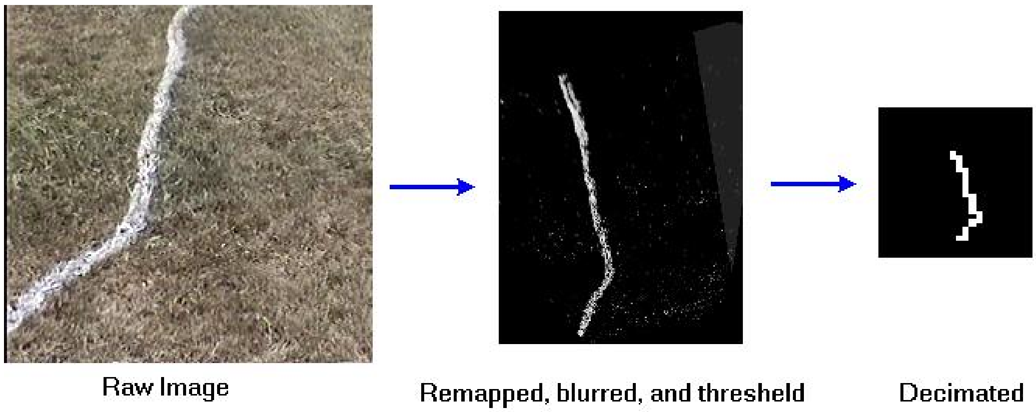

Figure 4. Image of line to evidence grid representation

The current version of Navigator does not use any sense of history in the decimated image. Future development will transform the resultant image based on vehicle motion, and then combine it the latest information obtained from the cameras. By weighting the old and new data differently, the algorithm can adjust the relative importance of the data based upon the confidence in that data.[2] For example, if the motion tracking is accurate and the vehicle is moving slow relative to the frame rate of the vision system it may be wise to use the historical data to build up confidence in the image processing. On the other hand, if the vehicle is moving fast or the motion tracking is inaccurate, it may be advisable to weight the new data heavier, and only use the old data if the cameras fail to obtain new data. Once this historical tracking is included in the algorithm, the decimated image will behave more like a tradition evidence grid, in the sense that the evidence builds up with repeated measurements. At any rate, the use of the decimated image in the algorithm does not differ from a traditional evidence grid.

3.5 Vector Field Histograms

In order to fuse the data from the computer vision system and the laser rangefinder into a useable representation, Navigator uses a method based on the Vector Field Histogram (VFH) developed by Johann Borenstein.[6] The VFH was developed to overcome limitations found in traditional potential field methods, while maintaining a fast, reactive, navigation scheme.[6,7] Although the VFH uses intermediate representations, such as an evidence grid, the overall operation is such that it reacts to changes in its sensor information, rather than planning its path based on its sensor model.[6] This leads to a relatively fast and computationally efficient algorithm for determining paths through densely cluttered environments.

The VFH is based upon a one-dimensional polar histogram, where the histogram value represents the obstacle density in a given polar sector about the vehicle.[6] Because the sensors are looking at different types of obstacles in different ways, the use of the VFH provides an convenient method of combining the data into a common and useful format that represents both 2-D obstacles (lines, potholes, etc.) and 3-D obstacles (barrels, trees, etc). This polar obstacle density (POD) is generally a function of the distance from the vehicle to an obstacle.[2, 6] Several different functions have been proposed, with each function giving a slightly different behavior.[2, 8] The current implementation in Navigator uses a linear version of the form

$$POD = k\,e\,(a - b\,d) \tag{1}$$

where a and b are positive constants, d is the distance from the vehicle to the obstacle, e is the evidence value, and k is an arbitrary scaling coefficient. The linear form was chosen over other forms because it offered a good compromise between reacting to obstacles as they get closer to the vehicle and over reacting when they first come into view.[2]

The VFH for Navigator used 91 values representing sectors, in 2-degree increments, from -90° to +90° relative to the forward direction of travel. Two separate vector field histograms were calculated: one for the vision data and one for the laser rangefinder data. The polar form of the VFH is nicely suited to use with the laser range finder data, which is inherently polar. The POD for each sector in the laser rangefinder VFH was calculated using the closest reading within that sector.

The VFH for the vision data required conversion from the 2-D planar distribution of evidence values to the 1-D polar histogram form. Figure 5 shows the relationship between the 2-D evidence grid and the VFH sectors. The basic technique uses a pre-calculated map of POD values at each cell in the evidence grid, which is then multiplied by the actual evidence values to obtain the POD for each cell at a given time. The VFH takes the maximum POD for the cells within a given sector for the histogram value. Because the evidence grid is stored as an image, it is possible to precalculate several masks used in the VFH calculations during the program initialization. During run time, a series of image processing operations are used to apply the masks to the evidence grid to calculate the POD values needed for the VFH. Use of these functions, optimized for the matrix operations on the computer processor, significantly increases the processing speed. The calculation of the values of the POD mask is based on the transformation of the camera coordinates to the ground plane coordinates. A series of masks are created which select the pixels within a given sector. The sectors are allowed to overlap, which compensates for missing pixels within a densely crowded area.

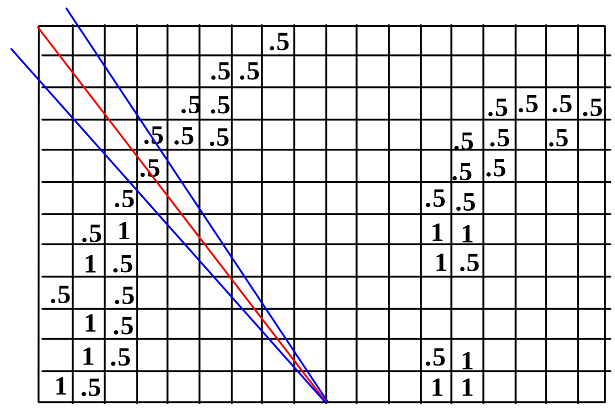

Figure 5. Evidence grid with polar sector shown.

Once the two histograms are calculated, fusion of the two data sets is straightforward. For each sector, the maximum POD is selected. Because the maximum POD value represents the closest obstacle, this technique allows the obstacles of greatest concern to dominate the navigation decisions. Figure 6 shows a representation of the two sets of VFH data superimposed on one graph. Areas of low POD represent obstacles in the distance, while zero values for POD represent free space at a given orientation.

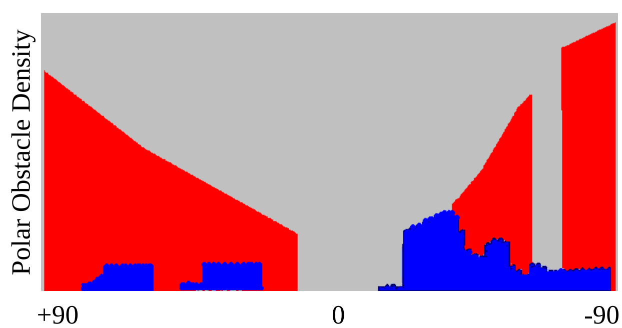

Figure 6. Composite VFH using range and visual data

4. NAVIGATION

Once the sensor data is fused into meaningful representation, the vehicle must select the most appropriate direction for travel. The selection of this direction is what gives the vehicle its apparent intelligence.[2] Navigator used a multistep process to polish the data, and select the most appropriate direction of travel.

The VFH representation is such that a high value represents an impassable area. In this arrangement, the vehicle must search for the minimum polar obstacle density.[6] Typically the VFH has areas of high POD and areas of low POD, Borenstein refers to these as "peaks" and "valleys".[6] In the original VFH algorithm, the vehicle selected the valley, closest to the target. Later refinements included aspects of the vehicle dynamics in the selection of the candidate valleys.[8] One complication with the software developed for Navigator was the lack of a predefined target. This lack of a predefined target, along with a desire to simplify processing, lead to the development of the passability representation.

4.1 Passability representation

The passability representation is the inverse of the obstacle density. This relationship is given by the following formula:

$$pass = \frac{POD_{threshold} - POD}{POD_{threshold}} \tag{2}$$

Sectors above a threshold obstacle density are considered impassible; therefore, the passability is set to zero. Furthermore, the passability is normalized against the threshold obstacle density, such that the maximum passability is one. By controlling the threshold obstacle density, the programmer can control how close a vehicle can approach a given obstacle. In Navigator, the obstacle densities were calculated relative to the position of laser rangefinder, therefore the threshold was calculated to correspond to an obstacle outside of the wheel boundaries so that the vehicle will never collide with obstacles.

4.2 Alleyway selection

In the VFH representation, the vehicle selects the direction based on the lowest POD values.[6] The VFH algorithm used low pass filters and obstacle enlargement to account for the vehicles size.[6, 8] The algorithm used on Navigator modifies these concepts slightly. Using the passability representation, the algorithm scans the passability histogram searching for discontinuities, which mark edges of obstacles. By seeking the higher passability regions between obstacles the algorithm identifies candidate alleyways(valleys in VFH). The algorithm then checks the angular opening of the alleyway to determine if the alleyway is wide enough for the vehicle to pass. Given equations 1 and 2, the width of the vehicle, and the passability at the obstacle, it is possible to calculate the required angular opening.[2] If the candidate alleyway opening is less than the required opening, the alley is marked as closed at the obstacles passability. Because the relationships are known a priori, the required openings for each passability can be calculated during program initialization and stored in a look up table. A final step in the passability processing uses a filtering process to round the corners of the alleyways. This has the effect of shading the passability values away from the obstacle boundaries. The filter is coded such that the filter is only applied using smaller adjacent values.

4.3 Steering selection

Once the candiate alleyways are identified and processed, the vehicle must select an appropriate navigation direction. For the Intelligent Ground Vehicle Competition, there is no target known ahead of time. The vehicle must travel an unknown course, staying within the boundaries and avoiding obstacles. For this reason, it was decided to bias Navigator to continue straight ahead until forced to turn by a course boundary or obstacle. This was accomplished by scaling the passabilities according to their orientation relative to the vehicle. In this way there is a cost associated with large steering angles. Figure 7 shows the resultant passabilitiy map in polar form after processing for alleyway width and the orientation scaling.

Once the passability histogram has been processed, a steering direction must be chosen. Early attempts based on the alleyway with the highest average passability did not work for large alleyways such as the right side of figure 7. Also, once an alley is chosen, the steering direction within that alley must be chosen. Again, early attempts based on steering toward the alley centroid caused the vehicle to oscillate on the course. The current version of Navigator searches for alleyways starting in the center and searching left and right. Boundaries of alleyways are marked according to the passability derivative. Furthermore, if the derivative is smooth but the absolute value changes by more than a given amount from the start a boundary is also marked. In this way, large alleys, such as shown in Figure 7, are subdivided into smaller alleys. Once the large alleys are subdivided, the vehicle picks from the small alleys chosing the one with the largest average passability. The vehicle steers toward the selected alley's centroid. Because of the scaling factor, this centroid is naturally weighted toward the forward direction. The steering is accomplished by calculating the turning rate as a function of the alley centroid. The speed setpoint of the vehicle is calculated as a function of the average passability of the selected alley. The Navigation

Manager program then sends the calculated speed and turning rate values to the PLC. Once the PLC receives the speed and turning rate commands, it handles the motor control as previously described.

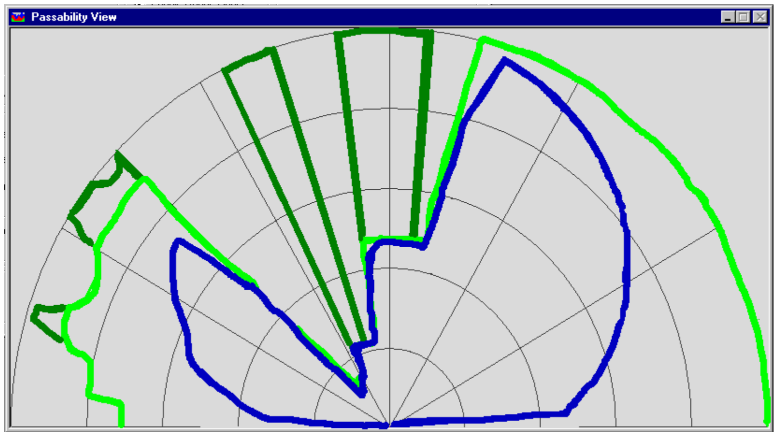

Figure 7. Polar passability plot showing raw, filtered, and scaled values

This reactive scheme never calculates an explicit path for the vehicle. The steering commands are updated based on the latest readings from the sensors. It the vehicle is not turning fast enough the changes in the VFH will force a larger turning rate. This happens because the POD values increase as the obstacles approach, eventually exceeding the threshold and forcing a new direction of travel.[6]

5. CONCLUSION

This paper has described the development and operation of a navigation method based upon the vector field histogram method. The method has proven to be reliable for fusing diverse data from the cameras and laser rangefinder. This method has been validated upon an actual hardware platform, the Navigator.

5.1Vehicle Performance

The overall performance of the vehicle was exceptional for a first generation vehicle. The vehicle was awarded 1st place in the design competition at the 8th annual Intelligent Ground Vehicle Competition. The vehicle obtained a 2nd place tie in the follow the leader competition, 3rd place in the road debris avoidance competition, and a 5th place finish in the autonomous challenge course. Navigator missed 4th place on the autonomous challenge course by just over 10 feet, while distancing the next closest competitor by over 60 feet. Lessens learned at the competition lead to refinements in the navigation code that improved the navigation decisions during subsequent testing.

5.2 Future Work

The next phase in the design of Navigator, being undertaken this year by a new team at Virginia Tech, is to analyze the dynamic forces on the vehicle, and optimize the frame design to lessen the vehicle weight. Most of the remaining issues with the performance of Navigator are dynamic issues related to the vehicle weight. By gaining a more complete understanding of the dynamic forces on the vehicle frame, the conservative design can be optimized in a fashion that both lessens the weight and provides a better distribution of weight.

Like all local path planners, this VFH based method requires trap detection and escape algorithms.[6] If a complete dead end is encountered, the vehicle must be programmed to execute a zero-radius turn and begin wall following or some other trap avoidance behavior. Additionally, cyclic behaviors, such as cycling between different traps, must be identified and avoided.[6]

Another area of development is the building of a global map based on the sensor data obtained during vehicle operation. The idea is to record data about the environment in order to build confidence in the location of objects. In this manner, obstacles can be used as references for localization in the global map. As Navigator "learns" the environment it can plan its desired path based on obstacles predicted to appear but which are out of the current view of the sensors. This will allow the vehicle to maintain a good constant speed and fluid motion throughout the environment.

Finally, the navigation system is built as an obstacle avoidance system. Use in landmine/UXO detection system will require more precise navigation specifically designed to provide coverage of a given area.[9, 10]

ACKNOWLEDGEMENTS

The authors would like to thank the following major sponsors of the Virginia Tech Autonomous Vehicle Team for their generous support and donations:

industrialcomputer.com	- industrial computer
Bodine-Electric Corp.	- electric motors
GE Fanuc	- programmable logic controller
SICK	- laser rangefinder
Ingersoll-Rand	- tools and financial support
Black and Decker	- tools
Framatome	- financial support

REFERENCES

1. P. J. Johnson, K. L. Chapman, and J. S. Bay, “Navigation of an Autonomous Ground Vehicle Using the Subsumption Architecture,” *Mobile Robots XI*, SPIE Vol. 2903, 1996 pp. 54-62.
2. D. C. Conner, *Sensor Fusion, Navigation, and Control of Autonomous Vehicles*, Masters Thesis, Virginia Polytechnic Institute and State University, July 2000.
3. D. C. Conner, P. R. Kedrowski, C. F. Reinholtz, and J. S. Bay, “Improved dead-reckoning using caster wheel sensing on a differentially steered 3-wheeled autonomous vehicle”, to be published in *Mobile Robots XV*, SPIE, 2000.
4. J. Borenstein, H.R. Everett, and L. Feng, *Navigating Mobile Robots: Systems and Techniques*, A K Peters, Ltd., Wellesley, MA, 1996.
5. M.C. Martin and H. Moravec, *Robot Evidence Grids*, tech. report CMU-RI-TR-96-06, Robotics Institute, Carnegie Mellon University, March, 1996.
6. J. Borenstein and Y. Koren, “The Vector Field Histogram -- Fast Obstacle-Avoidance for Mobile Robots.” IEEE Journal of Robotics and Automation, Vol. 7, No. 3., June 1991, pp. 278-288.
7. J. Borenstein and Y. Koren, “Real-time Obstacle Avoidance for Fast Mobile Robots.” IEEE Transactions on Systems, Man, and Cybernetics, Vol. 19, No. 5, Sept./Oct., pp. 1179-1187.
8. I. Ulrich and J. Borenstein, “VFH+: Reliable Obstacle Avoidance for Fast Mobile Robots.” Proceedings of the 1998 IEEE International Conference on Robotics and Automation. Leuven, Belgium, May 16-21, 1998, pp. 1572-1577.
9. H.M. Choset, M.J. Schervish, E. Acar, Y. Shen, “Probabilistic methods for robotic landmine search”, to be published in *Mobile Robots XV*, SPIE, 2000.
10. H.M. Choset, E. Acar, A. Rizzi, and J. Luntz, “Exact Cellular Decompositions in Terms of Critical Points of Morse Functions”, Proceedings of ICRA '00, IEEE, 2000.