

A Method for Integrating Vision and Laser Range Measurements in Autonomous Ground Robotic Vehicles

Bluefield State College Center for Applied Research and Technology

Robert N. Riggins, Bruce V. Mutter

briggins@bluefieldstate.edu

bmutter@bluefieldstate.edu

1. Abstract

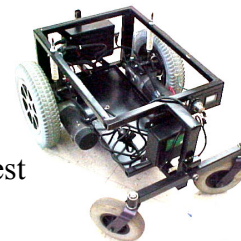
A typical Autonomous Ground Robotic Vehicle (AGRV) uses a combination of sensors to monitor movements and the surrounding environment. Placing multiple sensors on an AGRV may allow for complexity in sensor data, but far more important is integration of the information from these multiple sensors to perform a given task optimally. One popular choice of sensors includes a Laser Measurement System (LMS) and a vision system. Good examples of robots using LMS and vision are vehicles entering the annual Intelligent Ground Vehicle Competition (IGVC) and competing in the 2005 Grand Challenge sponsored by the Defense Advanced Research Projects Agency (DARPA). This paper focuses on one method of integrating non-stereoscopic vision (camcorder) information with laser distance measurements. First, background information on one such AGRV mechanical structure and a sensor suite is provided. This platform allows testing of algorithms using real hardware. The paper also explains the AGRV processes and image management. The core presentation concerns the method used for integrating LMS with vision. Once integrated, LMS and vision act as one set of data with one format, yet the method exploits all the information available from both. Finally, the paper illustrates one way to use this processed information for finding paths through a field of obstacles and road edges.

2. AGRV PLATFORM

The Center for Applied Research and Technology (CART) at Bluefield State College designed and built an AGRV the students called "V2". Being a little larger than an electric wheelchair and weighing slightly less than 300 pounds, the vehicle has a control system that gives the robot superb maneuverability. A full suite of sensors allow the robot to sense many aspects about its environment. The particular sensor suite for the AGRV allows algorithms to mimic human decision making. Therefore, our vehicle provides an excellent platform for studying various autonomous algorithms such as the ones presented in this paper. This section will present the hardware design of for the vehicle in three parts: the mechanical system, the electrical system, and other design concerns.

2.1 Mechanical System

The overall mechanical design focuses on simplicity, durability, compactness, maintainability, and most importantly, safety. The vehicle is designed to operate and navigate safely in both indoor and outdoor environments. This small and versatile design provides the opportunity to test and develop the human-like system on a fully functional platform. The mechanical design can be divided into three separate categories:



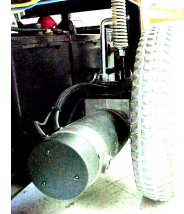
vehicle frame, drive system, and vehicle body.

Vehicle Frame

The vehicle frame is constructed of steel tubing. Steel tubing was chosen due to its light weight, durability, and ability to house wiring. The tubing acts as a conduit to conceal and organize connections as well as to shield vulnerable lines from RF noise. The rectangular design allows the frame to be strong while creating a protective carriage that houses the batteries, chargers, and other various components.

Drive System

Our ARGV uses two 24-volt DC motors to power two drive wheels independently. The motors attach to the drive wheels at 90-degree angles and pivot vertically through brackets welded to the frame. The brackets prevent any horizontal movement reducing stress on the motors. The motors attach to the suspension system and travel with the wheels independently. The angles that the motors are mounted also vary as the vehicle travels across uneven ground. This ensures that a motor will not hit the ground when its respective wheel enters a hole. The two rear wheels are free to rotate and change direction as the vehicle changes course. The rear wheels are mounted on a pivoting arm that allows the wheels to travel vertically, independent of the main drive wheels. The pivoting arm allows 30 degrees of rear wheel travel in both directions.



Vehicle Body

The vehicle's body framework is constructed of aluminum tubing. The exterior of the body consists of aluminum panels with lexan inserts around the entire surface. The panels are held in place with quarter-turn fasteners that can be removed by hand very quickly. Due to the number of panels and their positions, components can be added or removed easily. The body protects components from water and the internal heat that the vehicle generates. It is equipped with fans that cool and circulate the air inside the vehicle. Shelving inside the vehicle's body allows for component positioning and spacing, assisting in cooling the interior of the structure.

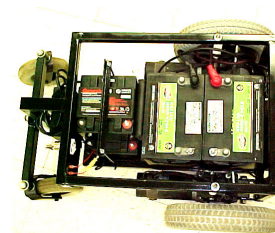


2.2 Electrical System

In the pursuit of a more human-like autonomous vehicle, a complex electrical system was designed to better imitate the human decision-making process. This imitation requires many sensors, multiple computers, a great deal of simultaneous processing, and various levels of redundancy. The system consists of four parts: the power system, sensors, computers, and vehicle control.

Power System

Two 12-volt deep cycle marine batteries connected in series provide the power to the controller, motors, vision computer, and LMS. Two smaller 12-volt batteries power the sensors, emergency stop contactor, and a DC-DC converter. The on-board laptop is equipped with two batteries for its own power. The DC-DC converter provides +12V, -12V, and 5V for the various requirements of the electronics. After performing a power consumption analysis, the team balanced the power consumption across all batteries. In normal operation, the vehicle operates for six hours on a fully charged set of batteries.





The vehicle is equipped with its own onboard charging system. The charging system consists of one 24-volt charger and two 12-volt chargers. The on-board laptop also has its own charger. Once switched to a charging mode, all batteries and electronics are isolated.


Sensors


The platform structure supports seven different types of sensing devices. This variety of sensors was chosen to provide various levels of data and redundancy similar to human senses. The following list the sensors onboard our ARGV, a brief summary and their respective data:


- **Stereoscopic Camera** – The stereo camera mimics human eyes. Not unlike humans, we can take process two slightly different images and create one image with depth information. Camera data contains the entire environment: lines, potholes, obstacles, etc. Another option is to use a single camcorder in place of the two stereoscopic cameras. For the algorithms presented in this paper, we used the single camcorder option.



- **LMS** – The LMS uses a laser to scan 180 degrees of the environment the vehicle is traveling towards. The LMS data contains the precise distance and angle of all obstructions in the plane of the laser.
- **DGPS** – The Differential Global Positioning System (DGPS) receiver uses global positioning satellites to obtain a position fix. It then uses a reference station and/or WAAS satellites to obtain corrections that improve accuracy. The DGPS data contains position (latitude, longitude), heading, and velocity.


- **Digital Compass** – The digital compass detects the earth’s magnetic fields. The digital compass data contains very accurate heading when moving slowly or stationary.


- **Encoders** – The encoders detect movement of the motor shaft with great precision. Data from the encoders contains position, velocity, and azimuth.


- **Diffuse Sensors** – By emitting light that reflects from a surface back to the sensor, the frequency can be analyzed and compared to a programmed frequency. The sensors can be programmed to detect a particular frequency (color) on the ground.


- **Proximity Sensors** – By emitting light that reflects from a surface back to the sensor(s), the proximity sensors can find obstructions.



Computers

The computing system is divided into two parallel systems. A central computer is responsible for planning paths, for controlling the vehicle, and for interfacing to all sensors except for the vision. The second computer system dedicates itself

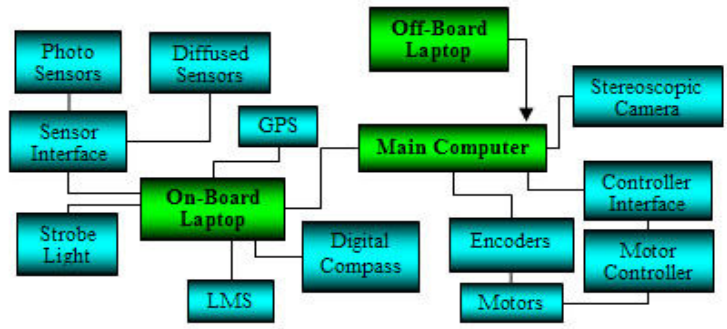


Figure 2.1 Computer Integration

to vision. A third, off-board, computer was also implemented to provide remote control, monitoring, and convenience. Figure 2.1 shows the two onboard computers as well as the off-board computer.

Vehicle Control

Our ARGV uses a closed-loop proportional control system. The team designed and built an interface that provides the controller with analog signals from the computer's digital signal. Encoders monitor the motors and provide feedback to the control algorithm. Figure 2.2 shows a block diagram of the control system.

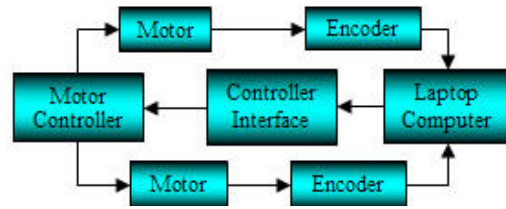


Figure 2.2 Control System

2.3 Other Design Considerations

Safety

Safety was an important concern in all aspects of the design, fabrication, and operation of the ARGV. This was accomplished through many different processes and was infiltrated throughout our design process. Especially important are the two manual pushbuttons located at the rear of the vehicle which, when pressed, disconnect power to the motors thereby effectively stopping the vehicle. In addition to these emergency stops, we included a remote e-stop system consisting of a transmitter and an onboard receiver. Fuses, circuit breakers, and disconnect switches protect all of the components from overloads, noise spikes, and short circuits.

Reliability

We have stressed the reliability of the ARGV by using redundancy. Different sensor groups have redundant functions. Both the stereo camera and the diffused sensors detect the presence of lanes. The stereo camera algorithm also doubles with the LMS and proximity sensors in detecting objects.

Durability

The solid mechanical design makes the vehicle very durable. Its framework houses and protects components. The exterior shell of the vehicle prevents water and debris from coming in contact with the electrical system. Components on the exterior of the vehicle are waterproofed and designed to withstand minimal damage. The vehicle can be operated under normal circumstances without fear of accidentally damaging vital components or affecting the vehicle's overall performance.

3. IMAGE PROCESSING ALGORITHM

Each pixel in each captured image contains information in the form of three 8-bit binary numbers for red, green, and blue (R_p, G_p, B_p) . One way to characterize each pixel graphically is to create a color vector in 3-dimensional space with pure red, pure green, and pure blue as orthogonal axes. Figure 3.1 illustrates such a color box where one can draw color vectors for pixels. The magnitude of this color vector represents the overall brightness of the pixel and the direction of the vector represents the relative color in the pixel. Written in vector form with c as the pixel column and r as the pixel row, the pixel color vector becomes:

$$\vec{P}_{RGB_{c,r}} = R_{P_{c,r}} \vec{i} + G_{P_{c,r}} \vec{j} + B_{P_{c,r}} \vec{k}$$

where $(\vec{i}, \vec{j}, \vec{k})$ are orthogonal unit vectors.

The algorithm filters spurious noise by using regions containing m^2 pixels in each region. There are

$$\frac{\text{Total number of pixels}}{m^2}$$

regions in each image. Averaging the m^2 pixel color vectors in each region creates a regional color vector:

$$\vec{R}_{RGB_{j,k}} = \frac{1}{m^2} \sum_{c=mj}^{c=m(j+1)-1} \sum_{r=mk}^{r=m(k+1)-1} \vec{P}_{RGB_{c,r}}$$

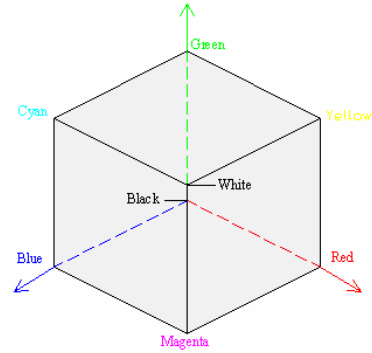


Figure 3.1 Color Cube

where j is the column number of the region and k is the row number of the region. One can compute the regional color vector equation above for each j and k to find the color vector for each region:

$$j = 0 \text{ to } \left(\frac{\text{ImageWidth}}{m} \right) - 1$$

$$k = 0 \text{ to } \left(\frac{\text{ImageHeight}}{m} \right) - 1$$

The result is a regional color vector:

$$\vec{R}_{RGB_{j,k}} = R_{R_{j,k}} \vec{i} + G_{R_{j,k}} \vec{j} + B_{R_{j,k}} \vec{k}$$

The regional color vector $\vec{R}_{RGB_{j,k}}$ contains averaged color distributions for the m^2 pixels in the region specified by j and k .

In general the AGRV will only have to recognize and navigate through and around certain known objects and surfaces such as sand, bridges, grass, tarps, construction barrels, and path markings. Suppose there are n such surfaces and objects. Each of these surfaces and objects has its own color vector as long as there is uniform distribution of color throughout the surface or object image. The algorithm assumes each region of each image must fall on or close to one of these n hypothesized surfaces or objects. Each has a possibility of being the correct one, so the hypothesized regional color vectors are given by:

$$\vec{H}_{RGB_L} = R_L \vec{i} + G_L \vec{j} + B_L \vec{k}$$

$$L = 1, 2, \dots, n-1, n$$

Any number of hypotheses is possible as long as no two vectors are collinear. The amount of separation of the hypothesized color vectors relate directly to the accuracy possible from this algorithm.

The algorithm described in this section exploits the property that the same pixel in shade or bright sun would have the same general direction in color space even if the magnitudes were very different. Therefore, the important quantities of interest are the angles between the hypothesized vectors and the actual regional color vector. The dot product between these vectors gives this angle:

$$\theta_L = \cos^{-1} \left[\frac{\vec{R}_{RGB_{j,k}} \bullet \vec{H}_{RGB_L}}{\left| \vec{R}_{RGB_{j,k}} \right| \left| \vec{H}_{RGB_L} \right|} \right]$$

The algorithm chooses the hypothesis with the smallest angle θ_L for each region. Also, probabilities of occurrence can also be accounted for—for example grass could have a greater probability than sand. Since the actual angle is not important, the decision can also be made using only the cosine of the angle:

$$\cos \theta = \frac{\vec{R}_{RGB_{j,k}} \bullet \vec{H}_{RGB_L}}{\left| \vec{R}_{RGB_{j,k}} \right| \left| \vec{H}_{RGB_L} \right|} \equiv r_L$$

For each region, the algorithm chooses the hypothesis with r_L closest to one.

4. INTEGRATION OF LMS AND VISION

Due to the particular hardware design of this robot platform as described in Section 2, laser and vision data formats are very different. The LMS mounts in front nine inches above the ground and sweeps horizontally 180 degrees from right to left. The video camera mounts on a post five feet high and points to the front and down at an angle of 45 degrees. As discussed before, this arrangement of camera and laser system is not so peculiar to just this robot since many autonomous robots use similar sensor placements.

This section explains the process of integrating these two sets of data and describes how to place the combined data on a map. The map, as illustrated in Figure 4.1, is a top-view two-dimensional rectangular coordinate frame containing a horizontal semicircle five meters in radius and centered at the front of the robot.

The LMS computes distances and angles to objects that lie in a plane above the ground in front of the robot. For example, in the figure, LMS has detected the presence of two construction barrels, indicated by the two small red curves. The LMS sends this data to the navigation computer in the format of a 1 by 181 vector of distances. Each vector element represents a distance to an object at an angle equal to the index (j) of the element

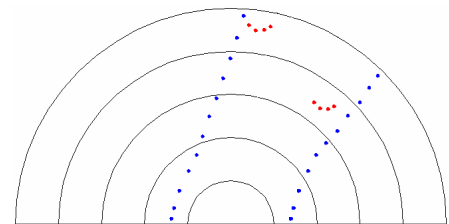


Figure 4.1 Dynamic Memory Map

LMS Distance (j) for j=0 to 180

where *LMS Distance(j)* is an element of the LMS vector of data. The range of distances for this particular LMS covers 0 to 8.192 meters. The LMS's ability to measure distances up to 8.192 meters is more than adequate for the map's maximum size of five meters. At a top speed of five miles per hour the robot needs only to look ahead a maximum of five meters in order to have more than two seconds travel time. Given our vehicle dynamics, a travel time of two seconds gives the robot enough time to react to an object by adjusting its heading and speed.

LMS data represented by the vector *LMS Distance* translates directly to the map described above. No distortions in the LMS data exist, so the data can be placed on a "bird's eye" view of a rectangular coordinate frame in front of the LMS. The figure shows sample LMS data plotted on concentric semicircles with radii in one meter increments. Since the algorithm can plot the LMS data directly onto the semicircle map, converting the data to another format is not necessary.

Vision, on the other hand, is not as straightforward as LMS. The rectangular picture box on the computer screen translates to a footprint in the shape of an "Aladdin's Lamp" on the ground due to radial distortion and due to the height and angle of the camera. For comparison, one could superimpose this footprint onto the map's semicircle in the figure. The challenge would then be clear: the algorithm must translate the pixel position on the computer screen to the pixel position on the footprint on the map. Each pixel on the computer screen has an (x, y) coordinate. The "x" value of a pixel must lie on some vertical curve on the map with one end of the curve at the bottom of the map and the other end at the top. At the far left side of the picture the curve is concave to the left and on the far right the curve is concave to the right. In the same manner, the "y" value of a pixel from the computer screen must lie on some horizontal curve on the map with one end of the curve at the left and the other end at the right. The bottom curve is concave down and the top curve is concave up.

The following method of integration converts the obstacle pixel values of each vision frame to a 1 by 181 vector of distances with each element representing the distance to an obstacle at an angle equal to the index (0 to 180 degrees) of the element in the vector. This converted format is exactly the same format as the LMS format described above. Converting vision format to LMS format allows the algorithm to place both the LMS and vision data directly onto the same semicircle map.

Curve-fitting analyses using MATLAB show that vision pixels translate to the map on parabolic curves. For example, a row of pixels on the computer screen translates to a vertical parabola on the map. The axis of the vertical parabola is parallel to the y axis. Likewise, a column of pixels translates to a horizontal parabola on the map with its axis parallel to the x axis. Therefore, each pixel translates to the map on a pair of parabolas: a vertical parabola and a horizontal parabola.

For this AGRV application, the horizontal parabolas are nearly straight lines, so with little loss of accuracy the algorithm assumes straight lines for the horizontal parabolas. Making this simplifying assumption saves computer time. Test results justify using this simplification in this particular case; however, such a simplification would not be necessary in general.

The following equations relate the pixels to the map:

$$x_{map} = \left(x_{pixel} - \frac{\text{image width in pixels}}{2} \right) \left(\frac{\text{line width in meters}}{\text{image width in pixels}} \right)$$

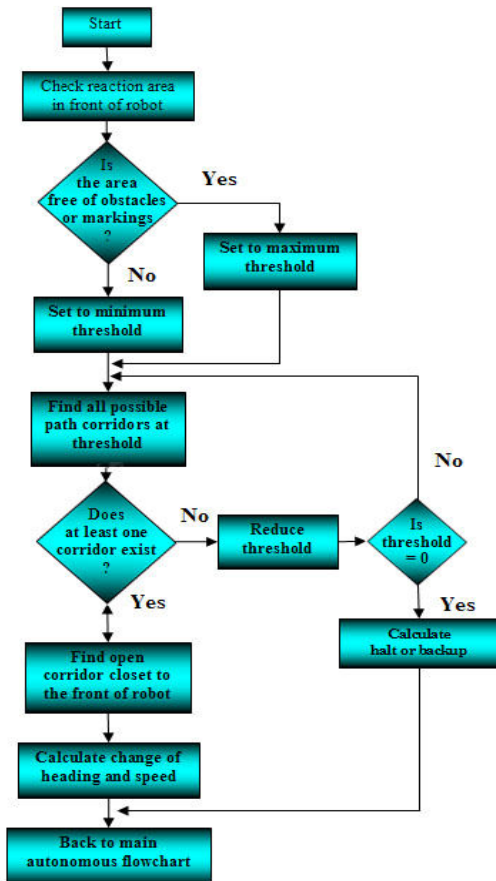
$$y_{map} = \left(\frac{b-a}{c^2} \right) x_{map}^2 + a$$

$$r_{map} = \sqrt{x_{map}^2 + y_{map}^2}$$

$$\theta_{map} = \tan^{-1} \left(\frac{y_{map}}{x_{map}} \right)$$

where (x_{pixel}, y_{pixel}) is the pixel location of the obstacle or marking on the computer screen, and (x_{map}, y_{map}) represents the location of the obstacle or marking on the map. The distances and angles (r_{map}, θ_{map}) form a vector similar to the LMS vector:

Vision Distance (j) for j=0 to 180



where r_{map} becomes the element *Vision Distance(j)* and θ_{map} the index j . The parameters a , b , and c determine the shape and orientation of the parabola. Curve-fitting analyses give the following equations for the parameters a , b , and c :

$$a = -.3911 + 4.453e^{-.008y_{pixel}}$$

$$b = -.5385 + 4.9745e^{-.00819y_{pixel}}$$

$$c = .9494 + 2.9771e^{-.01037y_{pixel}}$$

Given the coordinates of a pixel (x_{pixel}, y_{pixel}) on the computer screen, the algorithm described above computes the position (x_{map}, y_{map}) on the map. By converting vision format to LMS format, the algorithm can now place vision data on the same map as LMS. The path-finding algorithm of Section 5 uses this map containing both vision and LMS data.

Test results show that this integration method works very well. Objects sensed by both camera and LMS translate to the same location on the map. The algorithm will also place markings seen by the camera but not the LMS on the map such as the road edges shown in blue on Figure 4.1. Although the numbers are specific to the camera, the lens, and the position of

the camera, the general form of these equations works for many different cameras, lenses, and locations. The tests include two different types of cameras and two types of lenses. The only parameters to calculate in changing cameras or lenses are a , b , and c parameters.

5. PATH DETERMINATION

At this point we have a “birds-eye” view of all obstacle locations and markings the robot “sees.” The area the robot “sees” is a semicircle in front of the robot with a five-meter radius centered at the front of the robot. Information for obstacles and markings derives from processing and integrating data from LMS and vision as described in Section 4. The format of the information is a 1 by 181 vector of distances at each degree of angle starting at zero degrees on the right and ending at 180 degrees on the left. The next task is to navigate intelligently around obstacles and markings using this vector of processed sensor data.

The idea is to mimic the human decision making process when navigating through a field of obstacles and markings. Humans will set a long range plan in mind for moving from one point to another. For example, a driver will aim the car in the general direction toward the middle of the road at some distance in front of the car, with the distance depending on speed; however, if an obstacle or marking is encountered in-between, the driver will slow down and navigate the obstacle or marking. In the same way the robot will attempt to find an open corridor to a point five meters down the path. If a path wide enough for the robot is not found, the robot will lower its “aim” distance and again attempt to find a path. If the robot cannot find a path even at close distances, then, just like the human, it will back away and try again somewhere else. The flowchart depicts the decision process the robot uses in navigating.

Using this flowchart as a guide, the robot first checks a reaction area just in front of the robot for objects and markings that somehow were missed before at longer range. If something is encountered the robot will react to it by slowing down and maneuvering around it. This action is analogous to a human driver’s action when a child runs in front of his car, or when a pot hole that he had not seen before suddenly appears in front of the car. The robot uses a reaction area in the shape of a semi-ellipse centered at the front of the robot. Although the size of the reaction area is kept small with respect to the size of the path planning semi-circle, the size, position, and shape of this reaction area are variable, depending on speed and other factors. If the robot finds something in the reaction area, then the threshold (maximum look-ahead distance) starts at close range (two meters). Otherwise, if the reaction area is clear, the threshold starts at the long range of five meters.

The navigation algorithm next finds all possible path corridors that are wide enough for the robot up to the threshold. The algorithm marks each candidate path corridor with a left angle and a right angle. For example, Figure 5.1 shows one candidate path corridor using a threshold of five meters. The corridor in this example is wide enough for the robot all the way to the five-meter threshold and is marked by a left angle and a right angle (a_2 and a_1). The number of candidate corridors could range from none to many. If no candidate corridors exist at that threshold, the threshold is dropped by one meter and the search repeated. When the threshold drops to zero, then no paths are possible so the robot backs out of the situation and tries the algorithm again from an entirely different viewpoint.

If the algorithm finds at least one candidate corridor at any threshold, the algorithm chooses the best corridor and suggests appropriate robot commands such as a change in heading and speed. “Best” is defined as the corridor closest to straight ahead. The optimal heading of the robot is not

necessarily toward the center of the gap at the end of the corridor. For example, if a human driver wants to turn onto a side road he may initially aim for the center of the side road opening, but will eventually aim more toward a point in front of the side road opening as he gets closer. The robot algorithm calculates a change of heading in a similar way. As the robot approaches the opening, it will adjust the heading depending on distance to the opening and angle of the opening with respect to heading. The algorithm uses the following function to calculate change of heading:

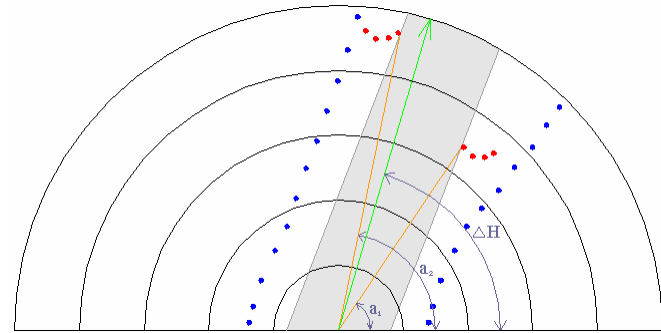


Figure 5.1 Candidate Path Corridors

$$\Delta H = \left(\frac{a_2 - a_1}{2} \right) * 2^{-(R-1)} + a_1$$

where a_1 is the left angle, a_2 is the right angle, and

$$R = \frac{\text{Distance at } a_1}{\text{Distance at } a_2}$$

Note when the opening is at a sharp angle relative to heading then ΔH is closer to the most distant point of the opening since $R \ll 1$ or $R \gg 1$. If the opening is perpendicular to the heading, then values $R = 1$ and ΔH will be halfway between the angles a_1 and a_2 (the robot would head toward the center of the gap). Speed of the robot is greatest at long range, but decreases as the robot approaches the gap. This simple but effective approach gives the robot “good dynamics” in maneuvers and allows smooth transitions during changes in heading.

6. CONCLUSION

This paper presented one method for integrating non-stereoscopic vision information with laser distance measurements for an AGRV. By properly integrating the data from these two sensors, the AGRV can make smart decisions based on all information. The exact nature of this data depends on the position and orientation of these sensors on the robot. The Bluefield State College CART robot team designed and built an AGRV robot that serves as an excellent platform for testing algorithms such as this one. The AGRV structure and sensor placement is similar to the most successful vehicles competing in the IGVC and the DARPA Grand Challenge. The lessons learned from the results of this integration scheme will be useful in the development of future vision and laser measurement-based decision making algorithms for the AGRV community.

ROBERT N. RIGGINS is an Associate Professor of Electrical Engineering Technology at Bluefield State College. He has been teaching in the electrical department for the past 6 years. In August of 1996, he retired with the rank of Lieutenant Colonel from the United States Air Force after 20 years of service. His experience in the field of Electrical Engineering spans the last 26 years, and includes a broad range of topics such as electronics, navigation, control, avionic sensors, robotics and sensors, computer programming, microcontrollers, and microprocessors.

BRUCE V. MUTTER, the founding director of the Center for Applied Research & Technology, Inc. (CART), teaches project management and engineering economics at Bluefield State College as an Associate Professor in the Division of Engineering Technology. He earned his A.S. and B.S. degrees in Architectural and Civil Engineering Technology from Bluefield State College, and his M.S. degree in Construction Management and C.A.G. in Environmental Design and Planning from Virginia Tech. He is a current Ph.D. candidate at Virginia Tech and completed additional graduate coursework with the University of New Mexico and New Mexico State University