Autonomous Vehicle Navigation Using Stereoscopic Imaging

Functional Description and
Complete System Block Diagram

By:

Adam Beach
Nick Wlaznik

Advisors:

Dr. Huggins
Dr. Stewart

December 14, 2006

I. Introduction

The objective of the Autonomous Vehicle Navigation Using Stereoscopic Imaging senior capstone project, NavBot, is to develop a robot that can independently navigate through a terrain that contains colored obstacles. The system will utilize stereoscopic imaging and color correlation to detect objects in its path. It will then calculate the distance from the robot to the obstacles. Distance calculations will be made using the pinhole model for the cameras. There will be two modes of operation for the robot. The first mode, calibration, will be used to setup and ensure that the subsystems are functioning within specifications. Navigation mode will be the main mode of operation. The goal is to navigate through the terrain as quickly as possible.

II. System Description

The system will consist of two cameras, a Gateway laptop computer, and an ActivMedia Pioneer 2 Mobile Robotic Platform as depicted in Figure 1. The robotic platform will be the same one that was used in the GuideBot and MapBot projects of 2005 and 2006 respectively. The cameras will be Logitech Buddy Cams. The two cameras and the laptop computer will be mounted on top of the robotic platform.

The NavBot will use stereoscopic imaging and color correlation to assess the terrain through which it is moving. Stereoscopic imaging is a technique used to create a three-dimensional map from 2 two-dimensional images. The distance from the robot to obstacles in its view can be calculated from the resulting three-dimensional map. The robot will be stationary when the images are to be taken, allowing the images to be captured one after the other rather than simultaneously. The three-dimensional map will be generated using edge detection and color space correlation. Appendix C contains a short description of color space correlation. The distance to the obstacles will be calculated using the pinhole model for the cameras. Information regarding the pinhole model can be found in Appendix B.

There will be two modes of operation: Calibration and navigation. The calibration mode will be used to set up the various subsystems and confirm that they are operating with in the specifications. The system will switch to navigation mode once calibration is complete.

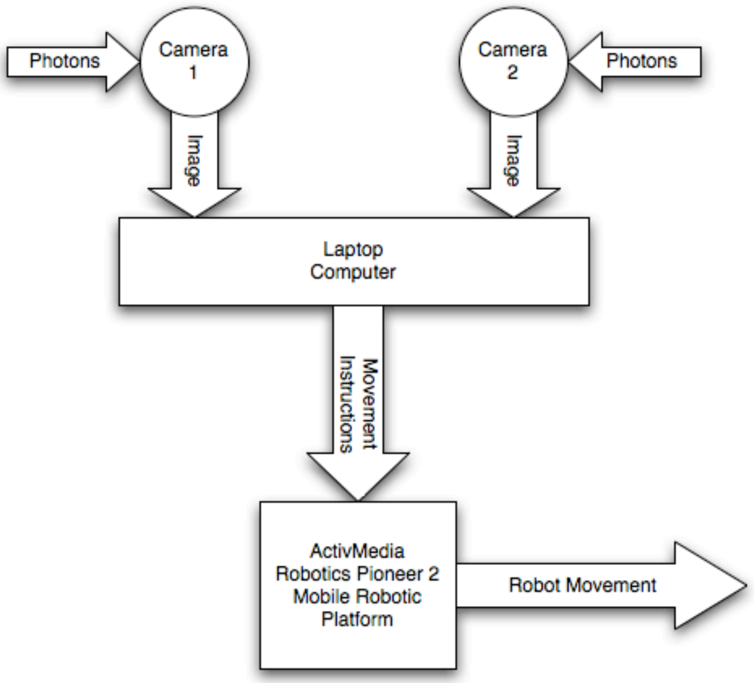

Figure 1: System Block Diagram

Subsystems

As depicted in Figure 1, the system has three subsystems. They are the camera subsystem, the laptop and software subsystem, and finally the robotic platform subsystem.

1. Camera Subsystem

The camera subsystem will consist of two Logitech Buddy Cams. These are color webcams that interface to the laptop via USB. As depicted in Figure 2, the cameras are mounted on a horizontal platform rigidly attached to the robot and are a fixed distance apart. The camera subsystem converts photons into digital data upon a trigger sent to them from the laptop computer. This digital data is compatible with Matlab. The image is stored in the Matlab workspace as a three-dimensional array containing the red, green, and blue values (0-255) for each pixel. The inputs and outputs to the camera subsystem are shown in Figure 2. The cameras function the same in the calibration and navigation modes.

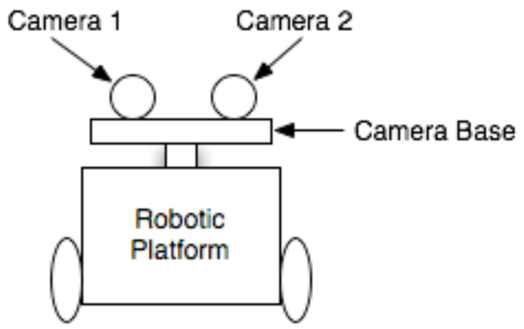

Figure 2: Illustration of Camera Mount

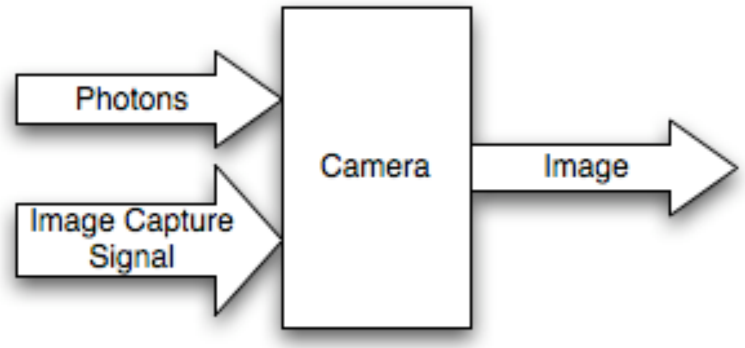

Figure 3: Camera Subsystem

2. Laptop and Software Subsystems

The laptop runs the software necessary for system operation. The laptop sends a signal out via USB to instruct each of the cameras to take a picture. The resulting images are then sent to the laptop via USB. The software on the laptop does the necessary calculations and makes a decision. Once the software has made a decision, signals are sent via a serial interface instructing the robot to move in the appropriate direction. The signals received from the robot contain information regarding robot motion and will be used during calibration mode. The user inputs and outputs are also used during calibration mode.

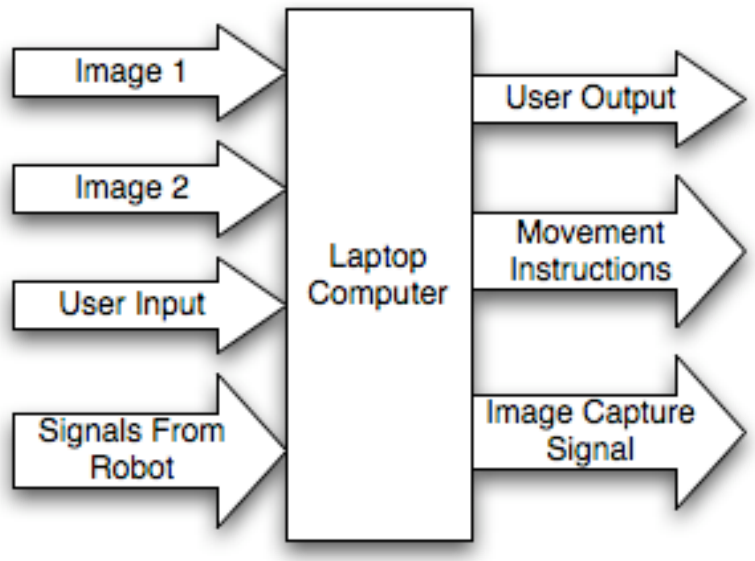

Figure 4: Laptop Computer Subsystem

3. Robotic Platform Subsystem

The ActivMedia robotic platform uses a software package called Aria. With this software, the robot can be controlled using single letters as instructions. For example, to move the robot directly forward, the letter d is outputted from the laptop via the serial interface. The microprocessor on the robotic platform analyzes the movement instructions signal and then sends signals to the motors to turn the wheels in the appropriate directions. The robot sends out other signals via the serial interface, however they are not relevant to this project. The robot functions the same in both modes.

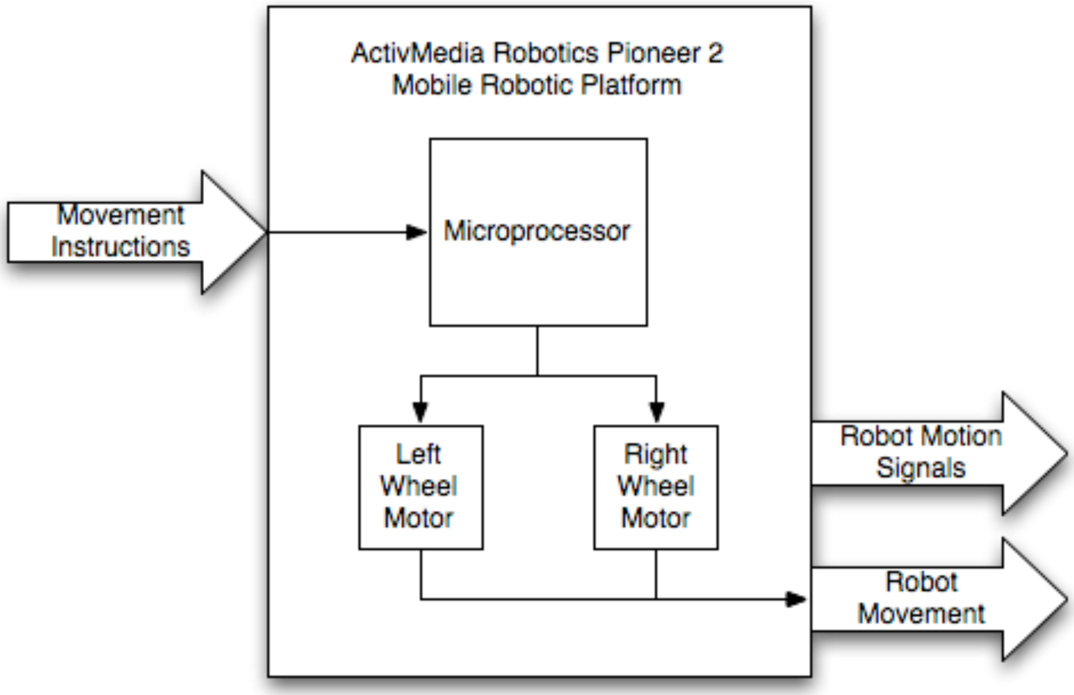

Figure 5: Robot Subsystem

Modes of Operation

There are two modes of operation, calibration mode and navigation mode. Each of these modes will be described in the following sections.

1. Calibration Mode

This mode will be used to ensure that the subsystems are setup and working properly. The user will initiate all of the steps performed in calibration mode manually. To ensure that the lighting is sufficient, a preview window of each camera will be opened in Matlab. The user will observe the preview and determine if the lighting is appropriate. While these preview windows are open, the user will also manually focus each camera if necessary. Lastly, the user will send various motion commands to the robot to ensure that it moves within specifications. If movements are not within specifications, appropriate changes will be made to the software to compensate for any discrepancies. When calibration is complete, the mode will be changed to navigation.

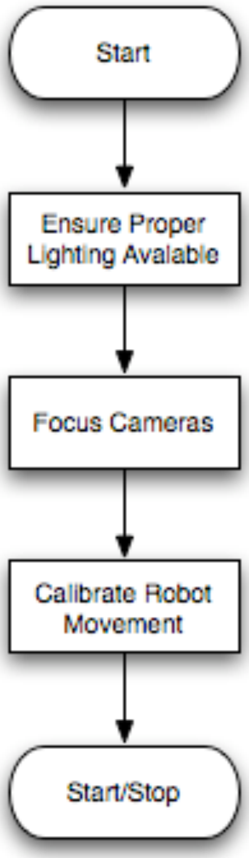

Figure 6: Calibration Mode Flow Chart

2. Navigation Mode

Navigation mode is the primary mode of operation. The software flow for this mode is shown in Figure 7. The first step is to retrieve the images from the cameras. The software will send a trigger to initiate this task. A signal will be sent to each camera to capture an image. The image will be stored in the Matlab workspace as an image array. Since the robot is stationary, the two images will be captured in succession. Unless supported by the cameras, Matlab is unable to process two image captures simultaneously. The Logitech Buddy Cams do not support this feature. Once the images are stored, the software will use them to generate a three dimensional map. The edges of the objects will be detected first. The software will use color space correlation to determine which edges belong to each object. The software will then use the pinhole camera model to calculate the robot's distance from each of the obstacles.

Once the system computes the necessary distances, it will need to determine which way to move. The software will have the robot's physical dimensions hard-coded. The gaps between obstacles will be calculated and compared to Navbot's actual size. The system will determine where a large enough gap is and then move so that it is inline with the gap and facing it. From here, the robot will move forward. If the robot is more than one meter away from the closest obstacle, it will move the appropriate distance to bring it within one meter. From there the robot will move 30 centimeters at a time, reanalyzing its situation after each movement.

The system will continue through this process until it has determined that the navigation is complete. This decision will be based on a timer. A predetermined navigation time will be set and the robot will cease navigation once the time has expired.

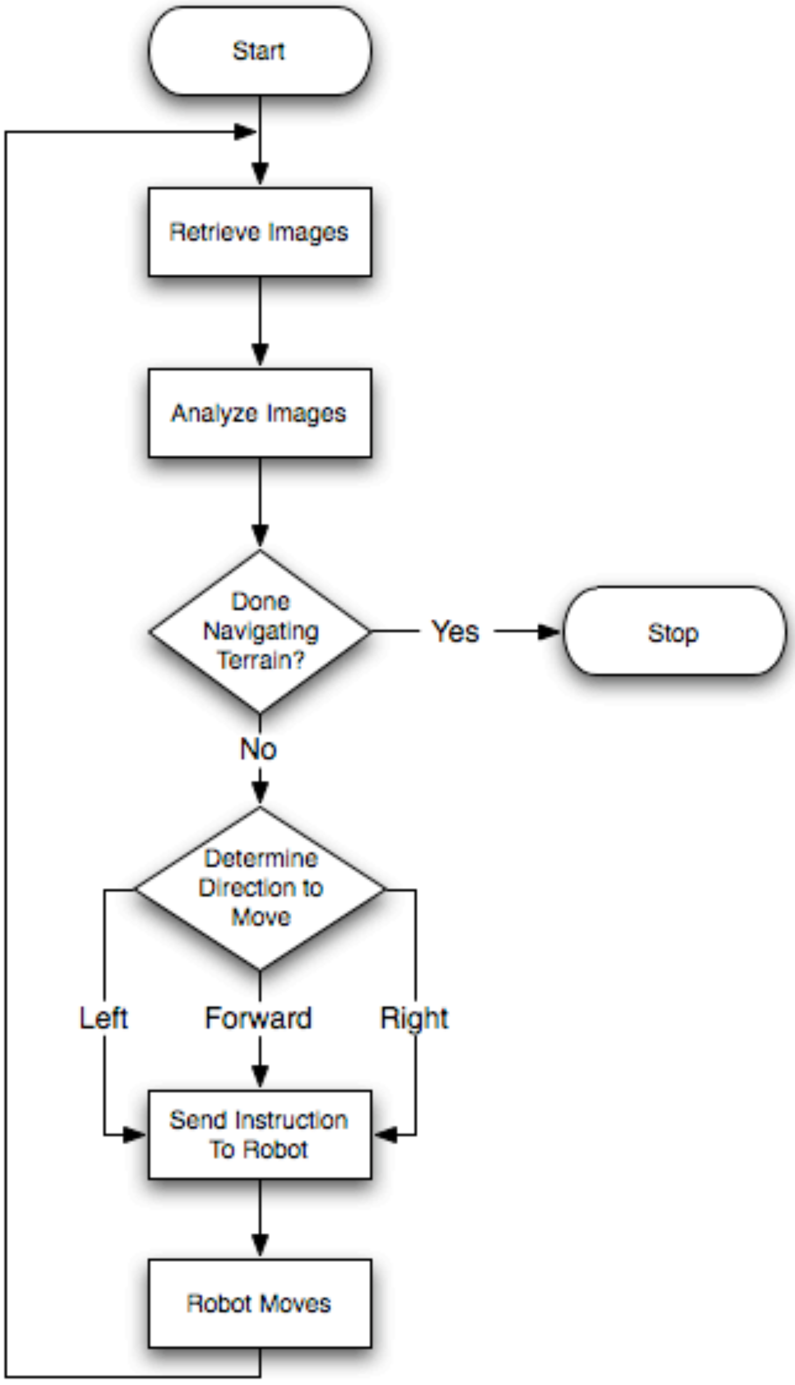

Figure 7: Navigation Mode Flow Chart

Appendix A
Color space Correlation

Color space correlation is a technique used in stereoscopic imaging to correlate images taken from two cameras. The following information was taken from Nick Patrick's paper *Stereoscopic Imaging*. The NavBot system will utilize this technique.

In order to determine the image disparity for each point, the correlation between one pixel and another needs to be defined. Since the images are supplied as RGB matrices, I can determine the distance in the color space for each color and sum each difference. The norm function is used for this purpose.

$$dist = \frac{\sqrt{(RL-RR)^2} + \sqrt{(GL-GR)^2} + \sqrt{(BL-BR)^2}}{3}$$

The square root term in the distance function could be eliminated to make the calculation more efficient; however it is useful in this case for keeping the range of distance between 0 and 255. The terms in the distance function correspond to the intensity of each pixel in each color, i.e. RX, GX or BX. The pixel from each diagram is referred to by XL or XR for the left or right figure.

In order to visualize the color space distance function, it is useful to imagine each color to be an axis. The distance between two points is then defined by the equation above. The image in Figure 9 shows two points in the color space dimensions. The coordinates of the points are R(230), G(100), B(40) for the red point and R(100), G(230), B(200) for the blue point. The distance between these points is 203.3.

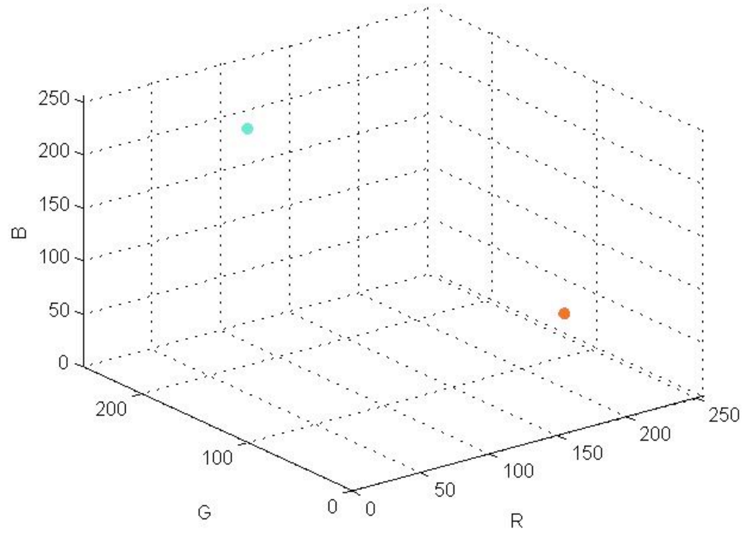

Figure 9: Two points in color space

Now that distance between two colors is defined, this function has to be applied to find corresponding pixels between two figures. Since each figure is aligned vertically, each pixel on the left figure only needs to be checked against the pixels on the same scan line on the right figure. The method that was used to determine the minimum color distance is to slide the right image across the left image from left to right. In order to ensure that noise does not cause too many false positives, each subtracted image is filtered by a small Gaussian filter from the MATLAB image processing toolbox. The filter causes each color distance to be affected by the color distance of its neighbors.

Appendix B
Pinhole Camera Model

The section below concerning the equations to determine the 3D position of objects using stereoscopic imaging is taken from the Project Proposal of the team that previously worked on this project, Brian Crombie and Matt Zivney. These equations are derived for a system with the cameras mounted in line vertically. The NavBot system will have cameras mounted in line on a horizontal axis. The equations will be modified to reflect this difference.

The equations to calculate the 3D position of an object in Cartesian coordinates using locations of the object in two camera images are shown below. This technique is known as stereoscopic imaging. Figure 10 shows the setup of the cameras and the coordinate system to ensure the validity of these equations. The two cameras are placed so there is an upper and a lower camera, and the lower camera must be centered on the x-y axis. The x-axis is assumed to be vertical, the y-axis is horizontal and perpendicular to the line of sight from the center of the cameras, and the z-axis is horizontal and parallel to the line of sight from the center of the cameras. The positive z-axis is pointing toward the objects to be viewed.

$$X = \frac{X_D * d}{X_D - X_U} \qquad \text{(Eq. 1)}$$

$$Y = \frac{Y_D * d}{X_D - X_U} \qquad \text{(Eq. 2)}$$

$$Z = \frac{d * f}{X_D - X_U} \qquad \text{(Eq. 3)}$$

In the above equations, d is the distance between the centers of the cameras and f is the focal length of the cameras. It is assumed f is the same for both cameras. X_D is the distance in the x-axis between the object in the lower camera and the center of the lower camera. If the object in the lower camera is above the center of the camera the distance is positive while if the object is below the center of the camera the distance is negative. X_U is the same as X_D applied to the upper camera. The distance in the y-axis between the object in the camera and the center of the camera will be the same for both cameras, so $Y_D=Y_U$. If the object in the cameras is to the right of the center of the cameras (facing the back of the camera) the distance is positive while if the object is to the left of the center the distance is negative. Figure 10 shows the sign convention for the variables X_D, X_U, and Y_D described above. Equations 1 and 3 were found on the Cooper University website, www.ee.cooper.edu, in the week 5 lecture notes for EE 458. The equation to calculate Y was derived using Equation 3 and the equation for a line in the y-z plane from the origin to an arbitrary point in 3D space.

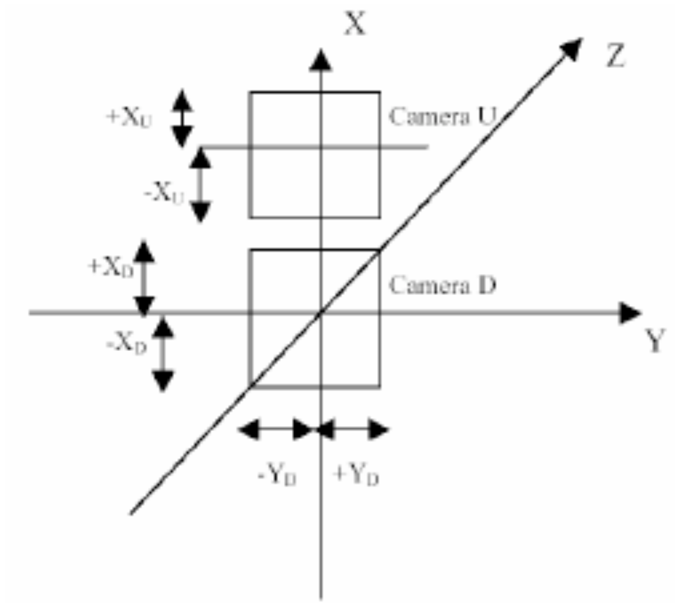

Figure 10: Setup of system to ensure validity of design equations