The Pennsylvania State University

The Graduate School

College of Engineering

HYBRID VISION SYSTEM FOR

ROBOTIC INDOOR NAVIGATION

A Thesis in

Aerospace Engineering

by

Anuradha Mangalgiri

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

August 2013

The dissertation of **Anuradha Mangalgiri** was reviewed and approved* by the following:

Lyle N. Long
Distinguished Professor of Aerospace Engineering and Mathematics
Director, Computational Science Graduate Minor Program
Thesis Advisor

Robert G. Melton
Professor of Aerospace Engineering and
Director of Undergraduate Studies

George A. Lesieutre
Professor and Head, Aerospace Engineering
Director, Center for Acoustics and Vibration

*Signatures are on file in the Graduate School.

ABSTRACT

A hybrid vision system was developed for indoor robot navigation. The robot used an all-terrain robot chassis as its base. It was equipped with a minoru stereo camera, a parallax micro controller, sabertooth motor controller and two NimH batteries. A Samsung Q430 laptop was used as the onboard computer. The robot vision system utilized stereo vision for its navigation. Raw images were captured from the minoru stereo camera, in real time and using a feature-based stereo algorithm disparity maps were created. The disparity results were converted into depth results using the camera properties. The depth results were fused with the edge map of the scene. Range values were inferred from this fused image and using the range values in a fuzzy logic algorithm, the intersections were detected. The robot navigation was then based on the detected intersection.

To develop the system, various stereo algorithms were explored. Most algorithms provide excellent results for stereo images but fail with raw images from the camera. A feature based algorithm was found to be the most robust for the present vision system.

The system was tested in various scenarios. It was successful as a close range system. It was able to detect all the intersections in close range. In the tested scenarios it was able to move avoiding obstacles and walls.

Table of Contents

ACKNOWLEDGEMENTS

This thesis has helped me grow both professionally and personally. I started as a complete beginner in the field of computer vision and robotics but with the help of this project I was able to develop skills and tremendous knowledge in this field. I want to thank everyone who motivated and guided me throughout this project.

I want to thank Dr. Lyle N Long, for giving me an opportunity to work on this project. His contribution and guidance was a great help for this project. I want to specially thank him for being patient through all my delays and setbacks.

I would also like to acknowledge Oranuj Janrathitikarn for her assistance in troubleshooting all the hardware issues. She really helped me, when I was stuck.

Also, I would like to thank my family and friends for all their support and guidance. I would like to mention a special thanks to my parents, for always being there.

Chapter 1

Introduction

Mobile robots have multiple applications in many commercial and military systems. Over the years, with developments in hardware and software technology, the scope of mobile robots has improved tremendously. There is a vast range of mobile robots, from the smallest robots to robot planes, helicopters or submersibles, to humanoids and household robots [1]. These robots vary in degrees of autonomy, intelligence, and mobility [1].

Autonomy refers to a system's capacity to operate in the real world environment without any external control, for an extended period of time [1]. Most robots are not fully autonomous and exhibit autonomy only in certain controlled situations [1].

A successful autonomous robot should be able to successfully plan its path and avoid obstacles. Sensors allow a robot to evaluate its environment, and to exhibit autonomy. These sensors help emulate some aspect of cognition and obtain information about the robot's environment. Sensors are used to monitor numerous phenomena such as motion, visual scenes, sounds, smells, electromagnetic radiation, and others [1]. Sensors can be proprioceptive i.e. they are used to sense internal environments or exteroceptive i.e. they sense external environment [1]. Proprioceptive sensors can measure the robot's position, velocity, acceleration, load, internal temperature, battery charge, failures, etc. [1]. Exteroceptive sensors can be vision sensors, proximity sensors, audition, olfaction, touch etc. [1]. Also, two or more sensor data can be fused together to get a more accurate inference about the environment. For autonomous navigation most mobile robots use sonar transducers or laser sensors as their primary spatial sensor [2] [3] but now even vision sensors are being used.

Robot vision is a very broad field and includes areas such as visual surveying, pattern recognition and stereo vision [4]. Stereo Vision refers to the issue of determining the 3 D structure of a scene from two or

more images taken from distinct viewpoints. It is a ranging method that resembles the basic mechanism of human eye to get depth information. Stereo Vision is a very important aspect of robot vision because of its significance for moving in unstructured environments [3]. It is very effective in detecting obstacles. Stereo vision results can be prone to errors as the results are dependent on image quality and light conditions in the environment. Stereo vision results are defined by a measure known as disparity. Disparity is calculated by finding the correlation between homologous pixels in images. This disparity value is inversely related to the depth value. More details on stereo vision and algorithms can be found in Chapter 4.

Vision based navigation is applied in both indoor and outdoor navigation. The following section discusses some previous vision based navigation systems that have utilized stereo vision.

Road Scene Analysis by Stereo Vision

The authors Hautière, N., Labayrade, R., Perrollaz, M. and Aubert, D. [5] used stereo vision to detect the lateral position of vertical objects in a scene (Figure 1) Usually, vision is used with LIDAR to get accurate lateral position of objects. A quasi-dense u-disparity approach was used to calculate the sparse disparity of the scene (Figure 2). It showed good accuracy in detection even under bad weather conditions, without the use of LIDAR [5].

Figure 1: Car Detection

Figure 2: Disparity Map of a road Scene

Stereoscopic Mobile Robot System

This vision system was presented by P. Oskar, at the XIII Industrial Systems Scientific Conference [4].The system allowed a robot to detect a flight of stairs (Figure 4) and determine the stair's orientation. Edge detection is applied on the stereo image and then the analytic form of these edges is calculated using Hough transformation (Figure 5), stereo matching is done using the starting points. The distance of the stairs is derived using triangulation and finally the orientation of the stairs is calculated using geometric equations [4]. Figure 3, shows the Robotic System.

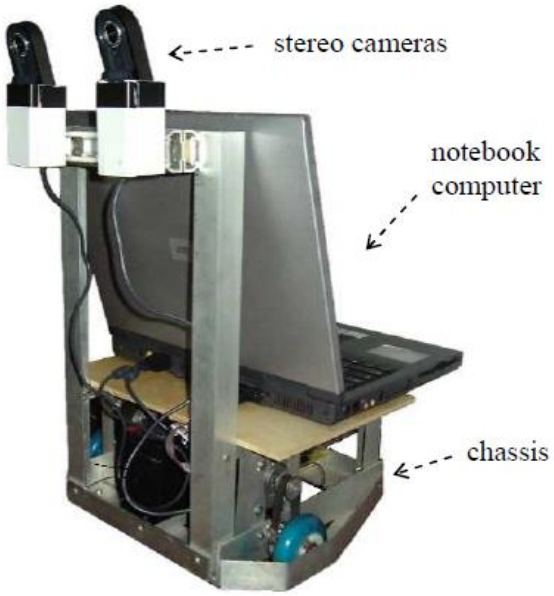


Figure 3: Mobile Robot

Figure 4: Camera Image of Stairway

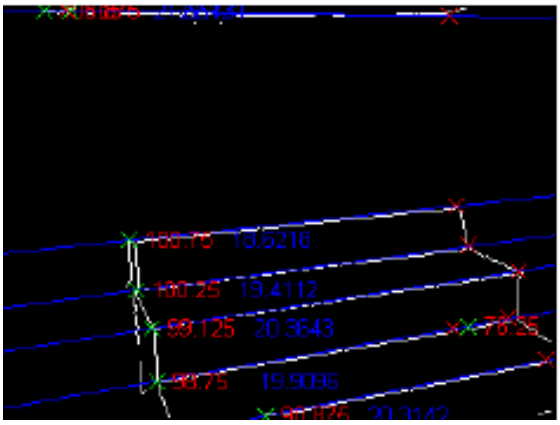


Figure 5: Detected edges, Hough line distance, angles

Jos´e

Jos`e is a Real World Interface (RWI) B-14 mobile robot (Figure 6). It has a Pentium PC running the Linux operating system as its onboard processing [3]. It uses a triclops stereo vision (Figure 7) module for getting depth information of the scene. The stereo vision module has 3 identical wide-angle (90 degrees field of view) cameras. Jos`e also uses a two-dimensional occupancy grid mapping to navigate and autonomously explore unknown and dynamic environments [3].

Figure 6: Jos'e

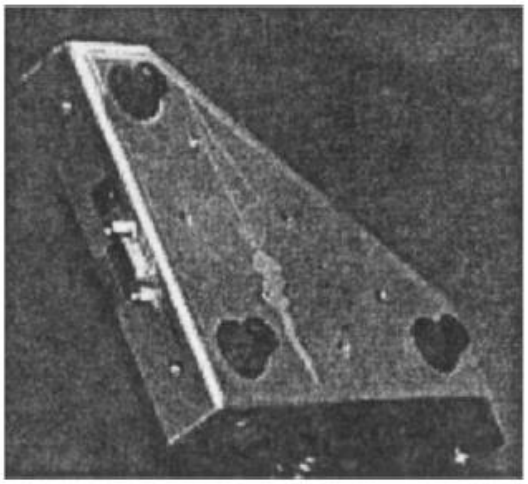


Figure 7: Triclops Camera

Real Time Stereo Vision for Obstacle Avoidance

In this method, presented by Kostavellis et al., disparity maps of the scene are generated using Point Grey's Bumblebee 2 stereo camera and then decision making algorithms are applied to generate the navigational commands for the robot (Figure 8) [6]. There are two methods that have been explored for the decision making process, the mean estimation method and the threshold estimation method. [6]

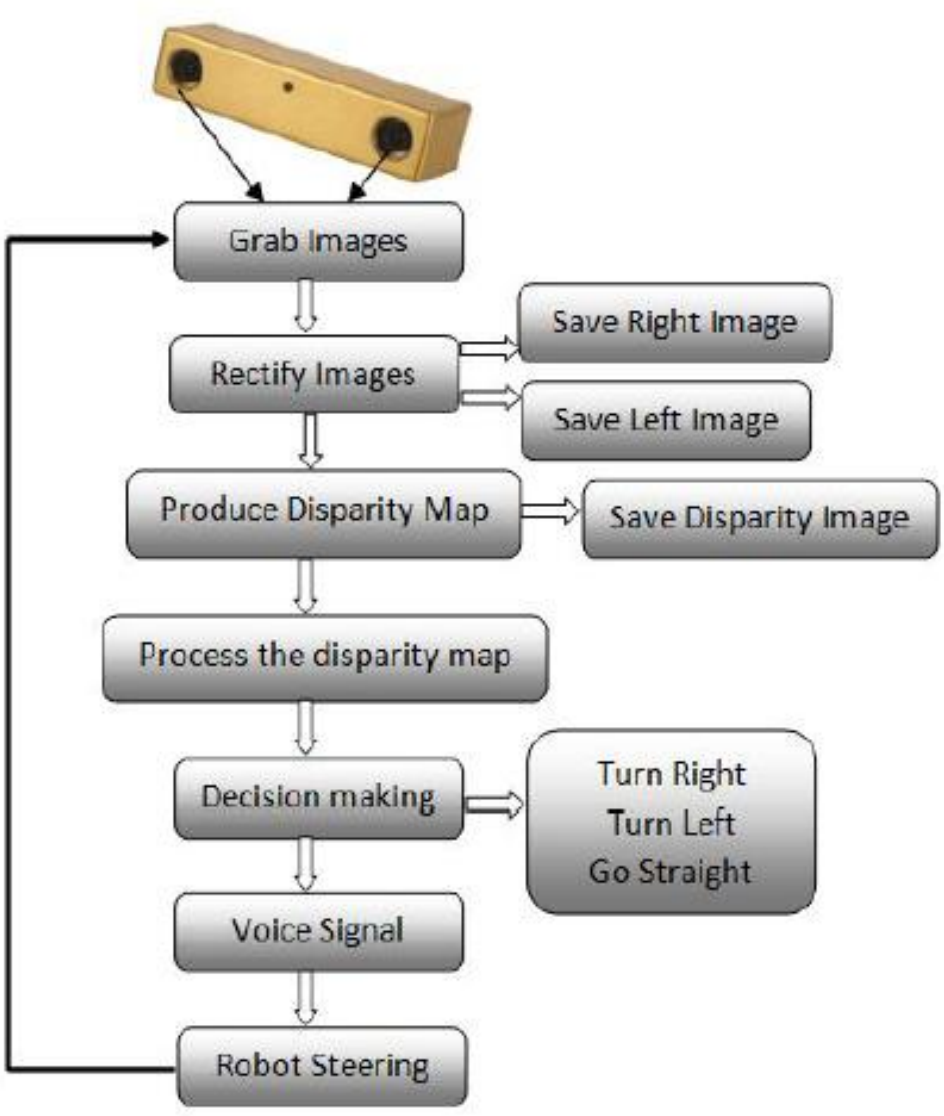


Figure 8: Chart for Obstacle Avoidance Algorithm

The Mean Estimation Method

For estimation by this method, the scene is divided into three windows of pixels as seen at Figure 9, and then a predefined frame is subtracted to avoid mismatches due to the different fields of view of the two images. Then the disparity map is divided into left side window, central window and right side window (Figure 9). For each window the pixels' disparity is aggregated and the smallest value of three indicates the smallest possibility of an obstacle in the window. Thus, that window becomes the preferred direction of navigation. This method can lead to problems because in the presence of noisy disparity maps, the robot can falsely detect obstacles and fail to move even though there are no obstacles. [6]

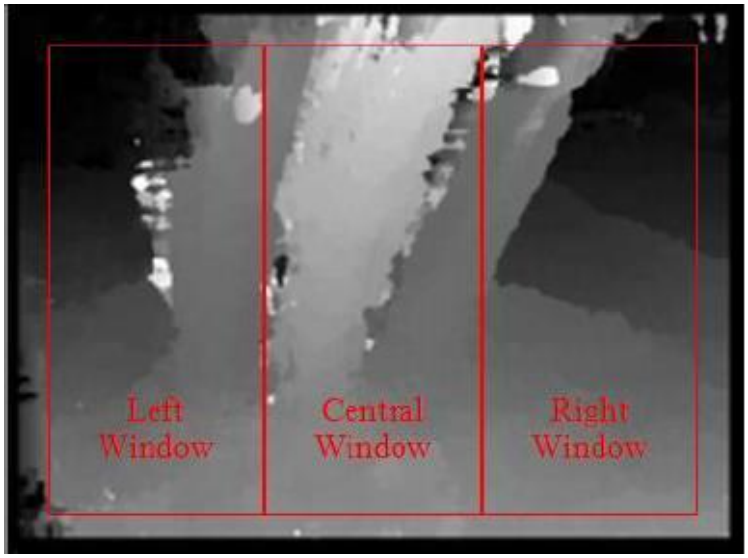


Figure 9: Separation into 3 windows

The Threshold Estimation Method

In this method too, the disparity map is separated into three windows; left, center and right. In the central window, the pixels p whose disparity value is greater than the threshold value T (T=120) are enumerated. If the enumeration rate is smaller than a predefined rate r (r=20%) then there is no obstacle in front. If there is an obstacle in front then the presence of obstacles in other windows is checked. Also, the average of pixel values in the left and the right value are compared to detect obstacles; after this the final decision is made. [6]

Chapter 2

Hardware

Robot

The robot base used here is a 4 wheel drive All Terrain Robot Base (figure 10). It is an extremely rugged and a well-built robot [7]. The frame is made of 1/8” thick aluminum and the dimensions of the frame are 12.25’’ wide x 17.25’’ long x 2.125’’ high. The wheels of the robot are Tamiya 6.75’’ diameter, All-Terrain Tires. The gear motors (figure 11) are 32 mm in diameter with permanent magnetic reversible DC motors [7]. They are 24 VDC and have a range of 8 to 265 RPM.

Figure 10: Robot Base [7]

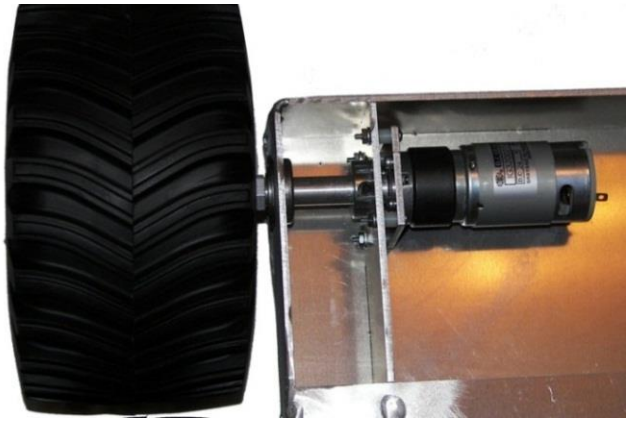


Figure 11: Robot motor and wheel [7]

Parallax Servo Board

The Parallax Servo board (Serial) (Figure 12) can control up to 16 servos at the same time. The pulses generated by the microcontroller are sent to the motor controller, which in turn controls the motion of the wheels of the ATR. The Baud rate can be set from 2400 to 380000 baud. For the present system the baud rate was set to 2400 because it allowed communication between the microcontroller and the motor controller [8]. The PSC supports several commands that are sent to it via RS-232 serial protocol. The voltage swing of this serial line is 0-5 VDC (TTL level). Each serial command must be preceded with an exclamation point, "!", and the pair of letters, "SC". The syntax for communicating with the motor controller is:

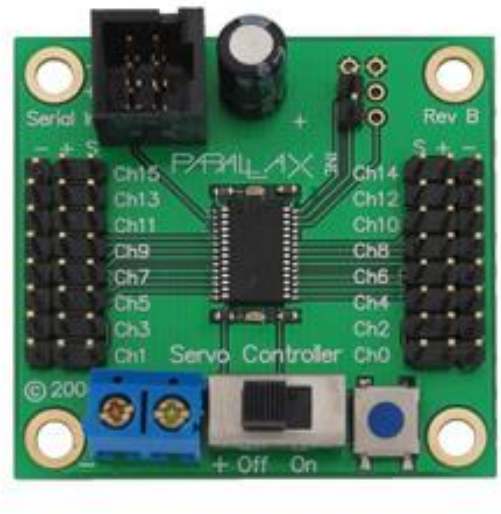


Figure 12: Parallax Servo board [8]

"!SC" C R PW.LOWBYTE, PW.HIGHBYTE, $0D

The pulse generated is a position command. A position command is composed of a header, three parameters: C, R, and PW, and a command terminator. C is the channel number. R is a binary number that controls the ramp speed of each channel; it takes values from 0-63. If value 0 is chosen, the signals are sent immediately without any delay. Ramp values of 1-63 correspond to speeds from ¾ of a second up to 60 seconds for a full 500 µs to 2.50 ms excursion, for standard servos. PW is a 16 bit word that corresponds to the desired servo position; it takes values from (250-1250). Based on the value of PW, the rotation of the wheels can be controlled, and thus the desired motion can be generated on the robot [8].

Sabertooth Motor Controller

The Sabertooth 2x 12 motor controller (Figure 13) is used to control the motors, which steer the robot. The Sabertooth features mixed modes designed especially for differential drive robots, where two motors provide both steering and propulsion. It also has independent options in all operating modes. This is useful if there are two motors for control, but the motors aren't necessarily being used to drive a differential drive robot. The motors do not need to be matched or even similar, as long as they both are within Sabertooth's operating limits [9].

The motorcontroller features a PWM frequency of 32 kHz, which is well above the maximum frequency of human hearing. Unlike some other motor drivers, there is no annoying whine when the motor is on, even at low power levels. It features dual temperature sensors and overcurrent sensing. It will protect itself from failure due to overheating, overloading and motor shorts [9].

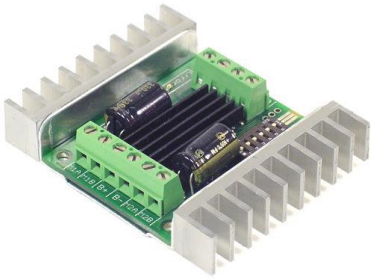


Figure 13: Sabertooth Motorcontroller [9]

Motor 1 is connected to terminals M1A and M2B, and Motor 2 is connected to terminals M2A and M2B. The input signals that control the Sabertooth are connected to terminals S1 and S2.These are connected to the channels on the Parallax motor controller. The battery is connected to terminals B- and B+. The negative side of the battery is connected to B- and the positive side to B+ [9].

The motorcontroller can be used with analog, R/C and serial input modes, as well as dozens of other operating options. For the present system the R/C input mode is chosen. R/C input mode takes two standard R/C channels and uses those to set the speed and the direction of the motor. The operating modes and options are set with DIP switches. There are six switches that control the operating modes [9].

The setting shown in Figure 14 was used for the robot. To operate the in R/C mode, switch 1 is set to the DOWN position and switch 2 to is set to the UP position. Switch 4 is in the DOWN position, to select Independent mode. In Independent mode, the signal fed to S1 input directly controls Motor 1 and the signal fed to S2 controls Motor 2. Switch 6 is set in the DOWN position, to select the Microcontroller mode [9] .

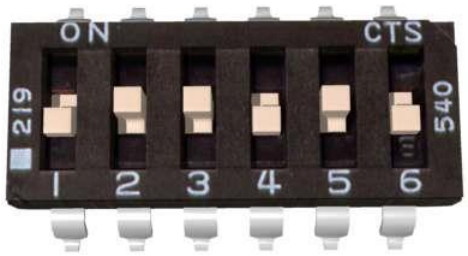


Figure 14: Dip Setting [9]

Batteries

There are two batteries on the robot, one to power the motors and the other to power the microcontroller. A 24V 4500 mAHr NiMH 2x10 Battery Pack [10] (Figure 15) is used to power the motors for rotating the wheels. A Venom Racing 7.2v 3000mah NiMH Battery Pack 9 (Figure 16) was used to power the parallax microcontroller.


Figure 15: Venom Racing 7.2v Battery Pack 3000mah NiMH Battery Pack [10]

Figure 16: 24V 4500 mAHr NiMH 2x10 [10]

On Board Laptop

A Samsung Q430 laptop was used on the robot. The laptop has Intel Core i5 450M / 2.4 GHz processor and a Data Bus speed of 1066 MHz. It has a 4GB RAM, 500GB Hard Drive and 4 hours of battery life.

Minoru

The Minoru Stereo Camera (Figure 17) is used as the vision sensor for the system. It has two standard web cameras placed at a baseline distance of about 60 mm from each other. Minoru connects to the PC's USB port like any other webcam. The resolution used on the cameras is 320x240 pixels. [11]

The frame rate is 15 or 30 fps for most resolutions. 30 fps may not be achieved as the frame rate depends on factors such as USB transfer rates and exposure time. 15 fps is usually reliable [12]

The cost of the camera is $40. The effective stereo range of the Minoru, using 320x240 pixel images, is between 35cm and 2 meters. The range is limited mainly by the baseline distance between the cameras and the image resolution. Thus, minoru, works as a reliable close range sensor.

Figure 17: Minoru Stereo Camera [11]

A greater effective range of approximately 3-4 meters can be obtained by using a higher resolution. However, this requires a slower frame rate. At this resolution, the difference in frame capture times due to unsynchronized capture becomes a far greater issue. Also, for the resolution of 640x480 pixels the principle point is off by 10 pixels.

Chapter 3

Software

OpenCV

OpenCV (Open Source Computer Vision Library) is an open-source BSD-license library that includes hundreds of computer vision algorithms [13]. It was originally built by Intel [14]. It provides interfaces to Intel's Integrated Performance Primitives with processor specific optimization [15]. OpenCV has a modular structure, which means that the package includes several shared or static libraries. The following modules are available [13].

core - a compact module defining basic data structures, including the dense multi-dimensional array Mat and basic functions used by all other modules.

imgproc - an image processing module that includes linear and non-linear image filtering, geometrical image transformations (resize, affine and perspective warping, generic table-based remapping), color space conversion, histograms, and so on.

video - a video analysis module that includes motion estimation, background subtraction, and object tracking algorithms.

calib3d - basic multiple-view geometry algorithms, single and stereo camera calibration, object pose estimation, stereo correspondence algorithms, and elements of 3D reconstruction.

features2d - salient feature detectors, descriptors, and descriptor matchers.

objdetect - detection of objects and instances of the predefined classes (for example, faces, eyes, mugs, people, cars, and so on).

highgui - an easy-to-use interface to video capturing, image and video codecs, as well as simple UI capabilities.

gpu - GPU-accelerated algorithms from different OpenCV modules. [13]

Some features of OpenCV:

- Image data manipulation (allocation, release, copying, setting, conversion).
- Image and video I/O (file and camera based input, image/video file output).
- Matrix and vector manipulation and linear algebra routines (products, solvers, Eigenvalues, SVD (Support Vector Decomposition)).
- Various dynamic data structures (lists, queues, sets, trees, graphs).
- Basic image processing (filtering, edge detection, corner detection, sampling and interpolation, color conversion, morphological operations, histograms, image pyramids).
- Structural analysis (connected components, contour processing, distance transform, various moments, template matching, Hough transform, polygonal approximation, line fitting, ellipse fitting, Delaunay triangulation).
- Camera calibration (finding and tracking calibration patterns, calibration, fundamental matrix estimation, homograph estimation, stereo correspondence).
- Motion analysis (optical flow, motion segmentation, tracking).
- Object recognition (Eigen-methods, HMM (Hidden Markov Models)).
- Basic GUI (display image/video, keyboard and mouse handling, scroll-bars).
- Image labeling (line, conic, polygon, text drawing)

OpenVis3D

OpenVis3D is an efficient 3D computer vision library that provides routines for image and video processing. It includes routines for dense stereo matching, optical flow estimation, occlusion detection, and egomotion (3D self-motion) estimation. [16]

The benefits of this package are: [16]

- Provides symmetric dense stereo matching with occlusions, symmetric dense optical flow with occlusion, probabilistic 3D egomotion estimation. (egomotion is a method to capture the motion of the camera)
- Designed as a template library
- It uses adapters that are compatible with Matlab, OpenCV, and can be tailored to be used with any other image processing library.
- Modular structure of Stereo and Optical flow code make it a way to add new algorithms for local matching, global matching, pre-and post-processing.

Chapter 4

Vision System

A vision based image processing application consists of five steps. [17]

1. Acquiring the images
2. Digitized representation of image. This is denoted by a two dimensional function I(x,y) that is described with an array, x marks a column and y a row of the array. The domain for x and y depends on the maximal resolution of the image. Every possible discrete function value represents a gray value and is called a pixel.
3. The image is divided into segments based on the similarity of pixels.
4. Feature values are assigned to the segments.
5. The features determined in the previous step are classified.

Stereo Vision

Stereo Vision refers to the issue of determining the 3 D structure of a scene from two or more images taken from distinct viewpoints. It is a ranging method that resembles the basic mechanism of the human eye to get depth information. In binocular stereo, the images are simultaneously obtained from the left and the right camera and a unique mapping is found between the points in one image to the points in another image [18]. The points that don't have corresponding points in the other image are called occlusions [18]. The result of mapping provides a disparity map, which indicates the shift in the pixel distance between two corresponding points in the images. The disparity result gives the depth information. Disparity and depth are inversely related, which means that the higher the disparity between the two points in the images the closer the object represented by the point relative to the camera.

Using the camera calibration and epipolar geometry, the raw images from the camera are rectified to generate stereo images. For rectified stereo images the points can be matched along the same epipolar line. If the images are not rectified the stereo correspondence becomes an optical flow estimation problem [18].

The critical issue in stereo matching is to measure the similarity between correspondences, which is calculated as a matching cost [19]. A matching cost is computed at each pixel for all disparities under consideration [20]. The simplest matching costs assume constant intensities at matching image but more robust costs model (explicitly or implicitly) certain radiometric changes and/or noise [20]. Radiometric difference can be caused by the camera(s) due to slightly different settings, vignetting, image noise, etc. Further differences may be due to non-Lambertian surfaces, which make the amount of reflected light dependent on the viewing angle. Other differences can be caused because the strength or the positions of the light source may change when the images are acquired at different times [20].

The stereo algorithms can be classified as local methods or global methods [19]. Local methods utilize local measurements such as image intensity and phase, and aggregated information from multiple pixels using smoothness. The method of aggregation is to minimize the matching error within a rectangle window. The window size can be fixed, or there can be multiple windows, adaptive window sizes or predictive windows. The varying window sizes give performance improvements at discontinuities [18] [20]. Global stereo algorithm approaches rely on the extremization of a global cost function or energy. The function aggregates the terms associated with local matching property, smoothness constraints and sometimes the penalties for occlusion. Some of the energy minimization schemes that have been used are dynamic programming, simulation annealing, relaxation labeling, non-linear diffusion, maximum flow, graph cuts and belief propagation [18].

A typical stereo algorithm follows the following 4 steps [21]:

1. Matching cost computation
2. Cost (or support) aggregation
3. Disparity optimization
4. Disparity refinement

Stereo Algorithms that compare the intensity level of each pixel are grouped as Correlation based methods and the methods that aggregate similar pixels in each image and then compare that in the two images are called Feature based methods. Feature based methods are more reliable when dealing with real time systems.

Correlation Based Similarity Measure

Depth maps are produced by calculating the disparity at each pixel within a neighborhood. A square window of fixed size around the pixel of interest in the reference image is taken. Then the corresponding pixel within the window in the target image, along the scanline or epipolar line, is found [22].

Sum of Absolute Difference (SAD): This is one of the simplest of the similarity measures, which is calculated by subtracting pixels within a square neighborhood between the reference image I_1 and target image I_2 followed by the aggregation of absolute differences with the square window, and optimization with the winner-take-all (WTA) strategy. If the left and the right images match, the resultant will be zero [22].

$$\sum_{(i,j)\in W} |I_1(i,j) - I_2(x+i, y+j)|$$

Sum of Squared Differences (SSD): In this approach the differences are squared and aggregated and later optimized using the WTA strategy. [22]

$$\sum_{(i,j)\in W} |I_1(i,j) - I_2(x+i, y+j)|^2$$

Normalized Cross Correlation (NCC): SSD and SAD methods obtain high accuracy in radiometrically similar images but are sensitive to radiometric differences that violate the brightness constancy, i.e. in case the brightness is not constant then these methods lack accuracy. In such cases of radiometric difference NCC obtains robustness by removing or relaxing the brightness constancy assumption. [19]

$$\frac{\sum_{(i,j)\in W} I_1(i,j).I_2(x+i,y+j)}{\sqrt[2]{\sum_{(i,j)\in W} I_1^2(i,j).\sum_{(i,j)\in W} I_2^2(x+i,y+j)}}$$

Sum of Hamming Distance (SHD): SHD is employed for matching census-transformed images by computing bitwise-XOR of the values in the left and right images, within a square window. This step is followed by a bit-counting operation which results in the final Hamming distance score [22].

$$\sum_{(i,j)\in W} I_1(i,j)\ bitwiseXOR\ I_2(x+i,y+j)$$

Graph Cuts and Belief Propagation

Belief Propagation and Graph cut stereo algorithms model a disparity image as a Markov Random Field (MRF). Steps 1 to 3 of a stereo algorithm process can be accomplished by modeling the disparity map as a MRF, the disparity refinement step is accomplished by using Graph Cuts or Belief Propagation. Labeling an MRF is an NP-hard problem (Non-deterministic Polynomial-time-hard), both Graph Cuts and Belief Propagation approximate the optimal solution [21].

Modeling Disparity as MRF: The matching cost is reflected in the compatibility function $(\boldsymbol{x_p}, \boldsymbol{y_p})$, where $\boldsymbol{x_p}$is the disparity at pixel location p and $\boldsymbol{y_p}$ is the intensity difference between two pixels. The aggregate support for the candidate disparities can be accomplished by using the correlation based similarity measure but an MRF approach aggregates support by introducing a second compatibility function$\boldsymbol{\Psi}(\boldsymbol{x_p}, \boldsymbol{x_n})$, where n is the location next to p. This is known as pair wise MRF

and it is generally used for stereo problems because considering more neighbors makes inference on the field computationally intractable [21]. The joint probability of MRF can be written as [21]

$$P(x_1, x_2, x_3, \dots\dots\dots, x_N, y_1, y_2, y_3, \dots\dots\dots, y_N) = \prod_{(i,j)} \Psi(x_i, x_j) \prod_{p} \varphi(x_p, y_p)$$

Here N is the number of nodes (i, j) represent a pair of neighboring nodes.

The disparities are found by maximizing the above probability function. Taking the log of the probability function, it is seen that MAP (Maximum A posteriori) estimator is equivalent to minimizing a function of the form.

$$E(x_1, x_2, x_3, \dots\dots\dots, x_N, y_1, y_2, y_3, \dots\dots\dots, y_N) = \sum_{(i,j)} -\log \Psi(x_i, x_j) \sum_{p} -\log \varphi(x_p, y_p)$$

This can be expressed as,

$$E(x_1, x_2, x_3, \dots\dots\dots, x_N, y_1, y_2, y_3, \dots\dots\dots, y_N) = \sum_{(i,j)} V(x_i, x_j) \sum_{p} D(x_p, y_p)$$

V (.) and D (.) are energy functions. Maximizing the probability is equivalent to minimizing the energy functions.

MRF is defined in terms of the energy function. The function $D(x_p, y_p)$ is computed using Birchfield-Tomasi matching cost [23]. The cost function $V(x_i, x_j)$ is given as a Potts model:

$$V(x_i, x_j) = \begin{cases} 0, & if\ x_i = x_j \\ \rho_I(\Delta I), & otherwise \end{cases}$$

The function $\rho_I(.)$ is defined in terms of image gradient between pixels

$$\rho_I(\Delta I) = \begin{cases} P \times s, & if\ \Delta I < T \\ s, & otherwise \end{cases}$$

Here T is a threshold, s is penalty term for violating the smoothness constraint and P is a penalty term that increases the penalty when the gradient has a small magnitude. T,P and s are constant over the whole image [21].

Belief Propagation: The max product algorithm computes the MAP estimate of the whole MRF [21]. At each iteration, each node uses the message it has received in the previous iterations from neighboring nodes to calculate messages to send to those neighbors. The message update schedule can be synchronous or asynchronous. In the synchronous method each node first computes the message for each node and after that the messages are delivered to each node and next round of messages are calculated. In asynchronous method, messages are propagated in one direction and nodes are updated immediately. Asynchronous method propagates quickly compared to synchronous method. When the max-product algorithm converges on a graph with loops, it returns an approximate solution for the most likely labeling of the graph [21].

Graph Cuts: Like BP Graph Cuts finds a local minimum by making local improvements. The ‘’swap” algorithm is used to make the local improvements by choosing two possible states α and β, then finding α the nodes whose label should be changed to β, or vice-versa, in order to minimize the energy in the field as much as possible [21].

Comparison between Graph Cuts and Belief Propagation: The results returned by the two are comparable. The solutions provided by Graph cuts are smoother but not necessarily close to ground truth. The efficiency of the two methods lies in the formulation of MRFs [21].

Shape and Stereo Correspondence

This method was developed by Abhijeet S. Ogale and Yiannis Aloimonos [18]. The method uses scene shape as a major contributing factor for establishing stereo correspondence. In this method correspondence, segmentation, occlusion detection and shape estimation are done in concert.

Correspondence and Segmentation: Correspondence results can be improved if segmentation information about the image is known. Image segmentation can be easily deduced if correspondence is known. Also, scene shape critically affects correspondence and again correspondence information is needed to understand scene shape. Thus, correspondence, shape and segmentation are interrelated and we need one to solve the other. The stereo algorithm needs to find them using iterative cycles involving feedback loops to arrive at the best results. This is a feature based stereo algorithm.

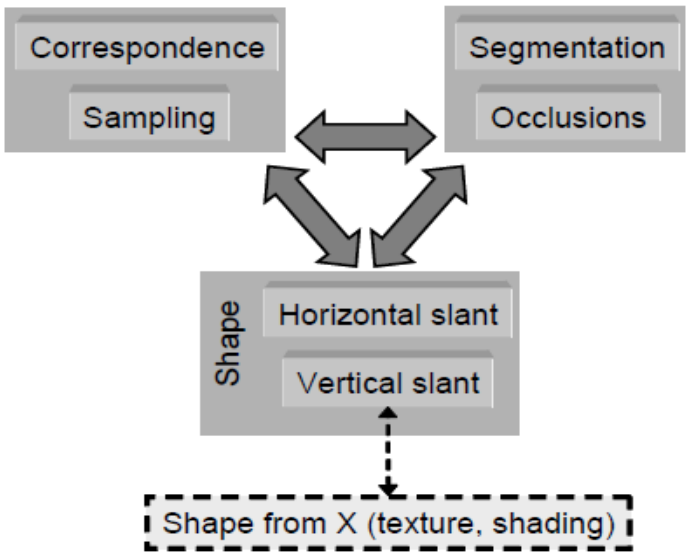


Figure 18: Shape and Stereo Correspondence

Finding results for flat lands: Flat lands consist of flat front or parallel surfaces only. The flat land stereo algorithm involves finding correspondence and segmentation in concert [18].

The absolute intensity difference between two images is found using:

$$\Delta I(x, y, \delta_x) = |I_1(x, y) - I_2(x + \delta_x, y)|$$

The goal is to find the true shift δ_x, which gives the disparity results for the stereo problem. The correct shift can be determined by finding large connected sets of matching pixels. Two pixels are a match if the intensity difference is below a certain threshold. Two pixels can be a match even if they don't correspond to a same point. Thus, a pixel may be part of a connected region even when the shift does not correspond to true shift, but the true shift value maximizes the area of the connected matching regions [18].

Another issue that is faced with this correspondence approach is that if two single colored regions with different disparity are separated by a horizontal edge, they will match regardless of the shift value. To solve this, connections must be severed along horizontal edges to avoid errors in finding the true shift values.

Algorithm for flatland

For every shift $\delta_x \epsilon \{\delta_1, \delta_2, \ldots\ldots, \delta_k\}$, do

(a) Shift the left image I_L horizontally by δ_x to get I'_L, and match I'_L with I_R

(b) If a horizontal edge separates a pixel (x; y) and its vertical neighbor (x; y - 1), their connection must be severed.

(c) Build connected components taking into account the vertical connections established in the previous step.

(d) Find the sizes or weights of the connected components.

(e) For each pixel, if the connected component containing it is larger than the previous shifts, update left and right disparity maps, while enforcing uniqueness.

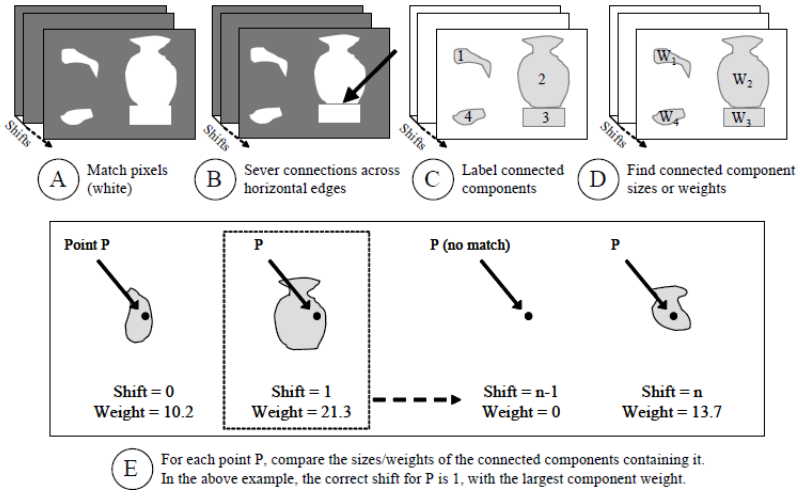


Figure 19: Algorithm for flatland

Horizontal Slant: For horizontal slanted surfaces the disparity changes along the X-axis and not along the Y-axis. The stereo algorithm assumes that the cameras are separated only by translation along the X-axis. The system provides rectified images, from the raw images of the camera. For raw images from the cameras the horizontally slanted lines in one image will always project into segments of different lengths into the other image. Consequently, N pixels on a scanline on the first image will correspond to M pixels on a scanline on the other image. Thus for horizontal slant, interval matching is more feasible than pixel matching. One of the images is stretched first and then the matching is done between the two images (Figure 20).

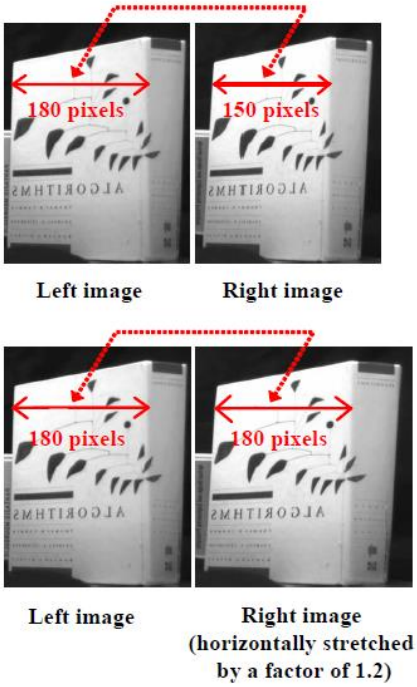


Figure 20: Stretch and Match

Algorithm for Horizontal Slant

1. For all $m_L \in M, \Delta_L \in \lfloor \Delta_1, \Delta_2 \rfloor$, do

(a) stretch I_L by m_L to get I'_L

(b) find range for d_L

(c) for every d_L, match I'_L and I_R and find connected matching segments and their sizes; update correspondence map while enforcing the uniqueness constraint.

2. For all $m_R \in M, \Delta_R \in \lfloor \Delta_1, \Delta_2 \rfloor$, do

(a) stretch I_R by m_R to get I'_R

(b) find range for d_R using given range.

(c) for every d_R, match I'_R and I_L and find connected matching segments and their sizes; update correspondence map while enforcing the uniqueness constraint

3. $m_L = m_R = 1$

(a) for every $d_L \in \lfloor \Delta_1, \Delta_2 \rfloor$, match I_R and I_L and find connected matching segments and their sizes; update correspondence map while enforcing the uniqueness constraint

Vertical Slant: A disparity change along the vertical can be caused due to a depth discontinuity or by a vertically slanted surface. Thus, the real reason for the disparity change cannot be determined. For this, additional assumptions must be made like vertical neighbors separated by a horizontal edge or by no edge, should not be connected. Disparity cannot be assumed to be constant when there is no change in color or intensity. If there is a non-horizontal edge running across an image, it will cause occlusions to appear if it is a discontinuity.

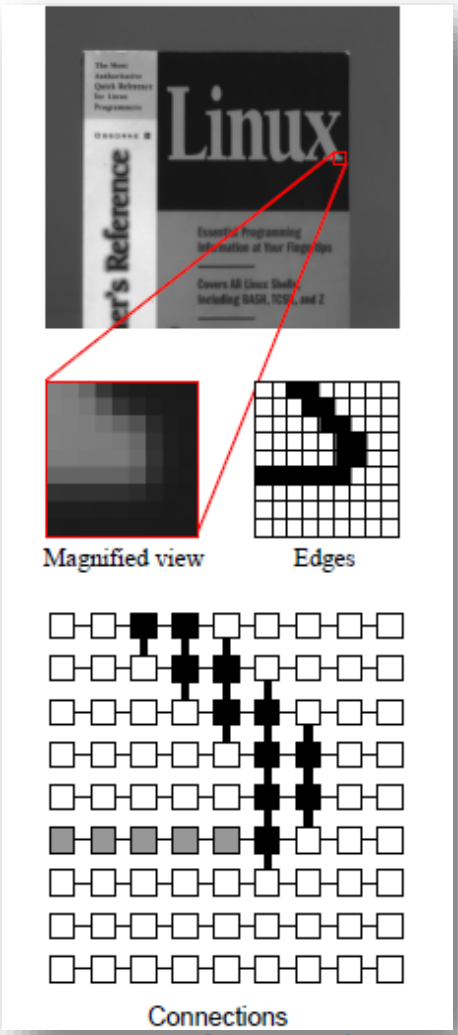


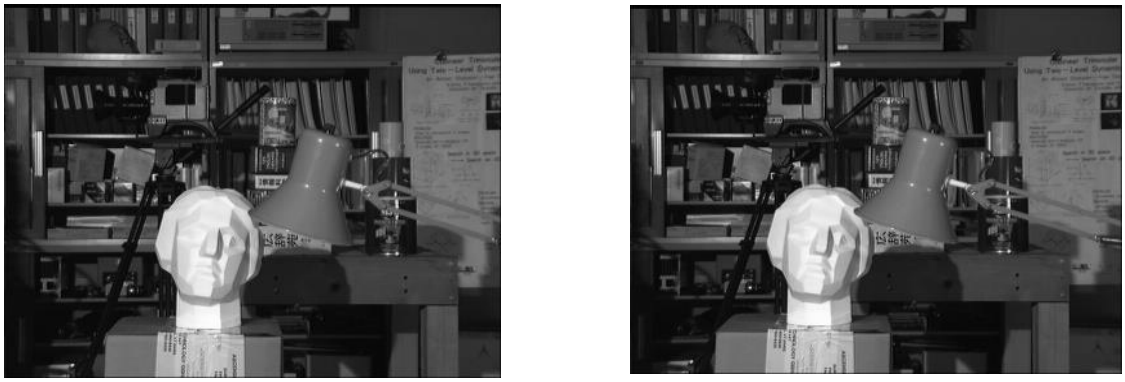
Figure 21: Vertical Connections between Pixels

There won't be any occlusions if both sides of it lie on the same surface. Thus, vertical neighbors lying on non-horizontal edges should be connected (Figure 21). This can be implemented by modifying the scanline algorithm. If it is assumed that each pixel is connected by links to pixel directly above it and pixel directly below it, then the only links left intact are the ones on non-horizontal edges. Then connected component labeling can be performed as before.

Comparison

Three types of stereo algorithms were tested and their results were compared to get the optimal method for the scope of the project. The methods were tested on Stereo images available at http://vision.middlebury.edu/stereo/data/scenes2001/ [24] and raw images from Minoru stereo cameras. The accuracy and computational efficiency of the methods was evaluated.

Tskuba Image: The algorithms were first tested on the Tskuba image (Figure 22), which is a rectified stereo image.



Left Right

Figure 22: Tskuba Image

(Figure 23) is the true disparity map for the image. The hotter the color the closer the object represented by the color is relative to the camera.

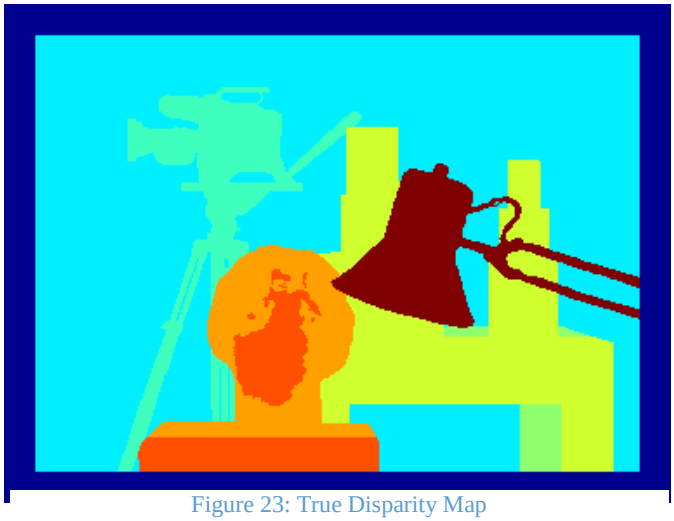


Figure 23: True Disparity Map

Correlation Based Similarity Measure: The disparity range for the method was chosen from 0 to 16 and the window size was chosen as 9. The code for the methods can be found at [23]. The above results (Figure 24) are noisy but are very close to the original ground truth.

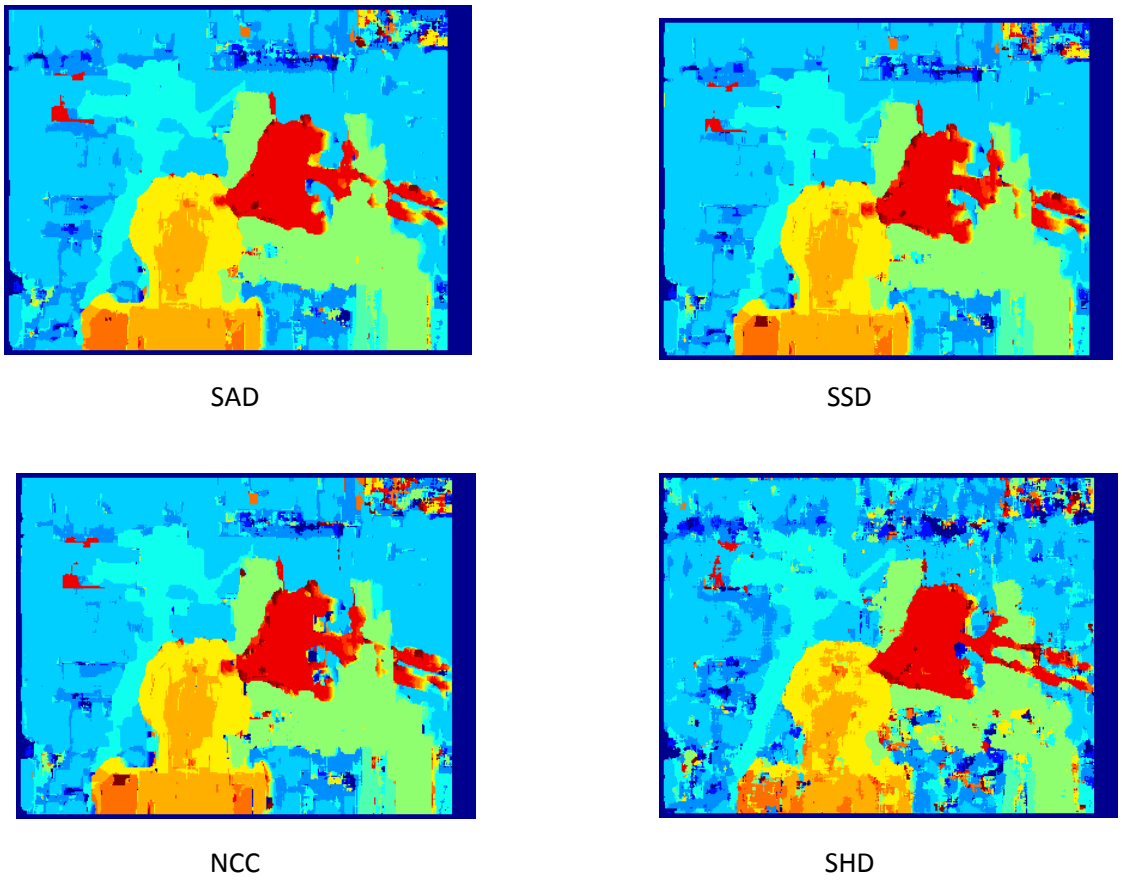


Figure 24: Correlation Based Similarity Measure

Energy Minimization: This method uses Belief Propagation to get a basic disparity map and then uses Mean Shift Segmentation to get a cleaner disparity Map. In Mean Shift Segmentation, the reference image is segmented and for each segment the associated pixel disparity is averaged to get the final disparity of the segment [25] . The code for this can be found at [25]. The disparity range was chosen to be 0 to 16. The above result (Figure 25) is close to the original disparity map (Figure 23) and is less noisy compared to correlation based similarity measure (Figure 24)

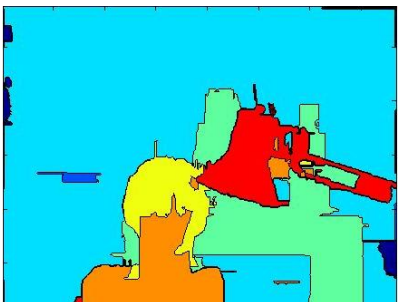


Figure 25: Energy Minimization

Shape and Stereo Correspondence**:** The disparity map range was chosen from 0 to 16. The stereo result (Figure 26) was close to the original ground truth (Figure 23) and was not noisy like the correspondence based similarity measure result (Figure 24) but was not as accurate as the energy minimization result (Figure 25). The code for this can be found at [16].

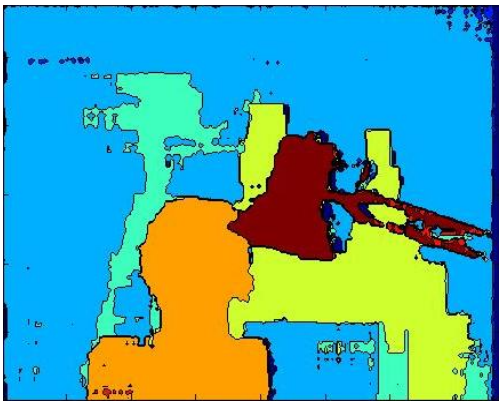


Figure 26: Shape and Stereo Correspondence

Room: A scene image (Figure 27) was taken from the Minoru stereo camera and the above discussed stereo algorithms were tested on the images. The left and right images have not been rectified to get stereo images.

Left

Right

Figure 27: Room Image

Energy Minimization: The disparity range was set from 0 to 70. The disparity result of energy minimization method was inaccurate on non-rectified images. In Figure 28, the segmentation result is accurate but the corresponding disparity for the segments is inaccurate.

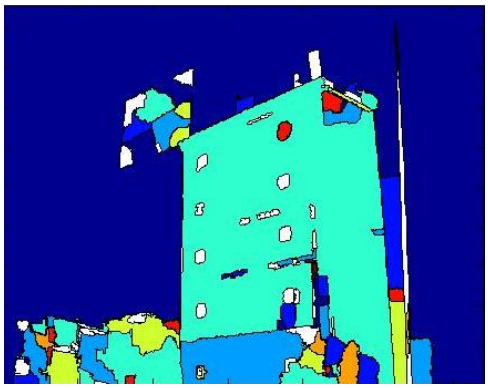


Figure 28: Energy Minimization (Room Image)

Shape and Stereo Correspondence: The disparity range was set from 0 to 70. As seen in the above figure (Figure 29), the stereo result is closest compared to the other two algorithms but it is still very noisy. For the scope of the project the results from the shape and stereo correspondence method were sufficient. Improvement in the stereo result would require more computation.

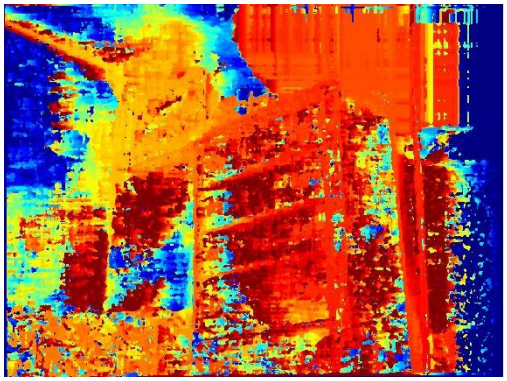


Figure 29: Shape and Stereo Correspondence (Room Image)

Disparity to Depth

Disparity is inversely related to depth. The higher the disparity value the lower is the depth value. Depth and disparity can be related by the following formula [26]

$$\text{Depth}=\frac{bf}{disparity * x}$$

b is the distance between the two cameras in length units

f is the focal length of the image sensor in length units

x is the pixel size of the image sensor

Edge Detection

Edges are significant local changes of intensity in an image [27]. They typically occur on the boundary between two different regions in an image [27]. The basic edge detection process involves estimating the gradient of the image, i.e. computing at each pixel (*x, y*) the values $u(x,y) \triangleq \frac{\partial I(x,y)}{\partial y}$ and $v(x,y) \triangleq \frac{\partial I(x,y)}{\partial y}$ by using the values of *I* (*x, y*) over a neighborhood *N* (*x, y*) of (*x, y*) and designating as edges the places where the length of the gradient vector estimate [*u, v*] = (∇*I*) exceeds some threshold value [28].There are two major categories of methods for edge detection: Gradient based edge detection and Laplacian based Edge Detection. The gradient based edge detection detects the edges by looking for the maximum and minimum in the first derivative of the image. The Laplacian method searches for zero crossings in the second derivative of the image to find edges [29].

Sobel Edge Detection

The Sobel operator consists of a pair of 3x3 convolution perpendicular kernels .These kernals are applied separately to the input image to produce gradients in each direction (G_x and G_y). These are then combined to get absolute gradient value for each point and the orientation of that gradient. The gradient magnitude is: [28]

$$|G| = \sqrt[2]{{G_x}^2 + {G_y}^2}$$

And the orientation is:

$$\theta = \tan^{-1}\left({G_y}/{G_X}\right)$$

Canny Edge Detection

Canny Edge Detection is a 5 step process of [30]:

1. ***Smoothing:*** Blurring of the image to remove noise.
2. ***Finding gradients:*** The edges should be marked where the gradients of the image have large magnitudes.
3. ***Non-maximum suppression:*** Only local maxima should be marked as edges.
4. ***Double threshold:*** Potential edges are determined by threshold.
5. ***Edge tracking by hysteresis:*** Final edges are determined by suppressing all edges that are not connected to a very certain (strong) edge.

Comparison

Sobel is computationally simple and provides detection of edges and their orientations but it is sensitive to noise and comparatively less accurate. Canny provides better accuracy especially in noise conditions but it is time consuming and computationally complex [29]. In the figure below, it can be seen that for a non-noisy image, the performance of Sobel and Canny is comparable but for a noisy image, Canny outperformed Sobel.

Lena Image

Sobel

Canny

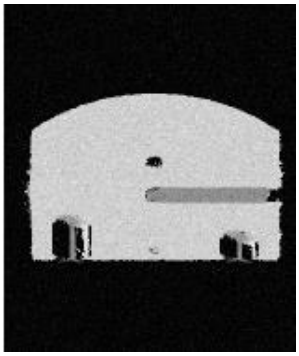
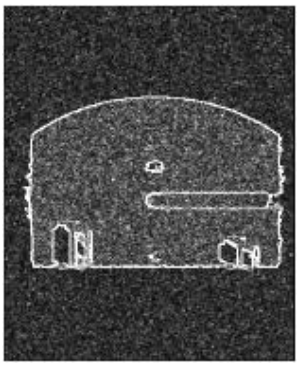


Image with Noise



Sobel

Canny

Figure 30: Comparison of Edge Detection Algorithms

Fused Image

The stereo disparity map is fused on top of the edge map of the left image to generate the fused image. This is done by replacing the pixels representing the edges on the edge map with corresponding pixels on the disparity map, to generate the fused image. The fused image gives us the disparity result of the edges in the image. If the depth map is used to generate the fused image, the depth values for the edges can be deduced. Figure 31 and Figure 32 gives the fused map for the tskuba and room image respectively.

Figure 31: Fused Image (Tskuba)

Figure 32: Fused Image (Room)

Chapter 5

Algorithm

The robot used the Minoru camera (Figure 17) to capture stereo images. The dense Stereo Correspondence algorithm is used on the images to get depth information of the scene. The depth data is then used in a fuzzy logic code to detect intersection, based on which the robot navigates. The following flow chart illustrates the algorithm used on the Robot.

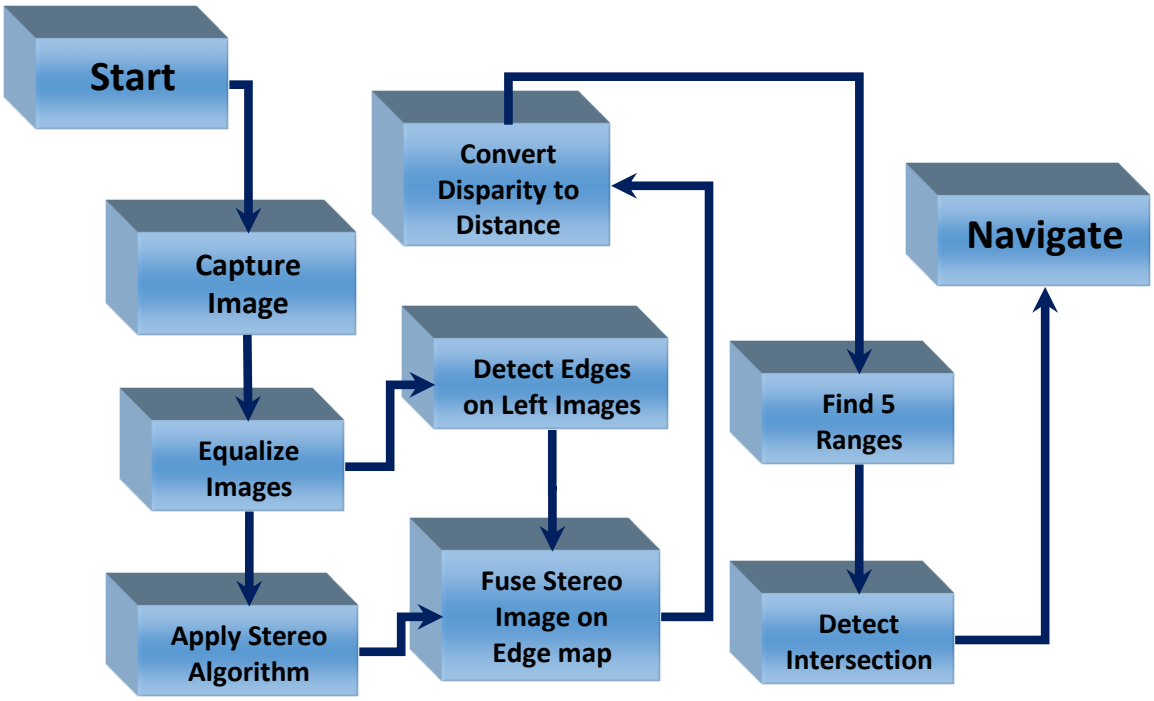


Figure 33: Robot Algorithm

Capture Image

The images are captured using the minoru stereo camera (Figure 17). The resolution used for the captured images is 320X240, and the frame rate used is 30 fps.

Equalize Images

The low image quality of the camera and flickering can result in the difference in the pixel intensity for the left and the right image. This can lead to errors in the stereo result because the corresponding pixels in the two images might refer to different intensities. To resolve this issue the intensities of the two images should be equalized. This is done by using the following formula

IR [i] [j] = IR [i] [j] x $\frac{\sum i \sum j \text{ IR[i][j]}}{\sum i \sum j \text{ IL[i][j]}}$

Where;

IR [i] [j] = right image intensity at pixel i,j

IL [i] [j] = left image intensity at pixel i,j

Stereo Algorithm

The dense Stereo Algorithm is used to get stereo results of the scene [30]. The algorithm works on images that have not been rectified and is fast and accurate for a real time system. The disparity range used for the stereo algorithm is 9 to 55. The values were chosen by using the following formula:

$$disparity_{min} \cong \frac{focallength \times baselinedistance}{range_{max}}$$

$$disparity_{max} \cong \frac{focallength \times baselinedistance}{range_{min}}$$

$focallength = 320\ pixels$

$baselinedistance = 60mm$

$range_{max} = 2\ m$

$range_{min} = 35\ cm$

The range values are the distance values that can be measured by the minoru camera for a resolution of 320x240 pixels. The focal length for the camera was found using MATLAB's camera tool box.

Detect Edges

The edges on the left image were detected using the canny edge detector because of its better performance under noise.

Fused Image

The edge image and the stereo image are fused to get an edge disparity map. The resulting image displays the stereo result only for the edges. Thus, the relative depth of the edges can be comprehended.

The edge image in OpenCV is a binary image, the pixels representing the edges have an intensity value 1 and the pixels representing the background have an intensity value of 0. To get the fused image the intensity value of all the edge pixels were replaced with the corresponding pixel intensity value on the disparity map.

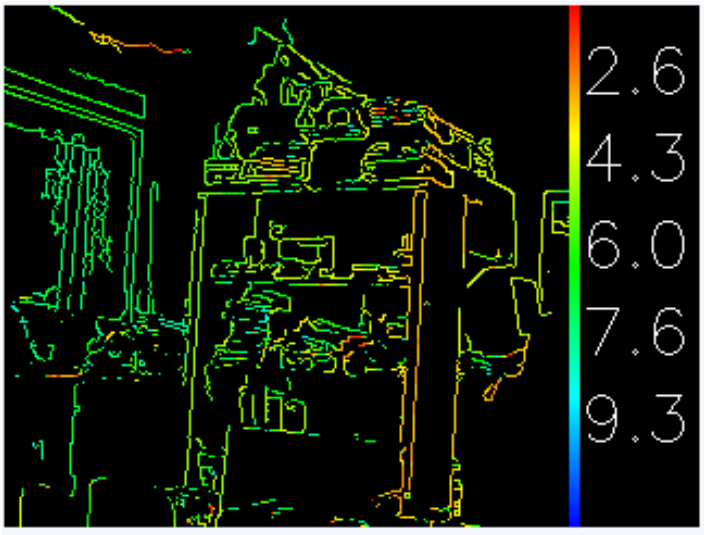


Figure 34: Fused Image

Disparity to Distance

The disparity can be converted into distance using the following formula

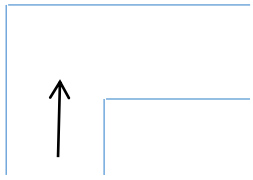
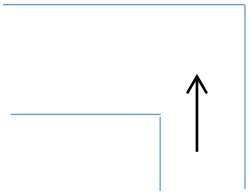
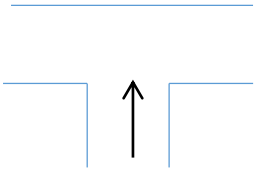
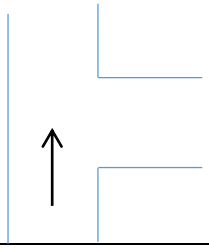
$$distance \cong \frac{focallength \times baselinedistance}{disparity}$$

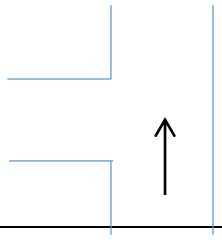
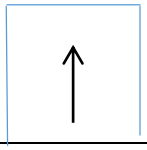
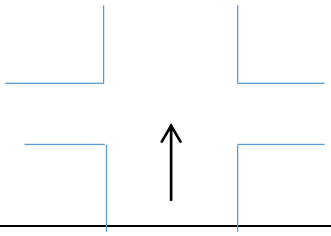
Find Ranges

The range values can be found using two methods. The first method the fused image is divided into 5 windows and the average distance value for each window is calculated. The first, third and the fifth value is later used to detect intersection. The other method is to divide the fused image into three windows and the average distance value of each window is used for intersection detection. At a lower resolution, dividing the frame into three windows yields better results.

Detect Intersections

There are 9 major intersections as shown in the table below:

Right Turn I	
Left Turn (L)	
T-intersection-bottom (T-b)	
T-intersection-right (T-l)	

T-intersection-left (T-r)	
Straight Hallway (S)	
Big Room(B)	
Dead End (D)	
X intersection (X)	

The first step in identifying the intersection type is to determine the presence of walls. The walls can be present in front, to the left or to the right of the robot. Of the five range values that were found earlier, the first, third and the fifth value are used to check the presence of walls. The three values are compared with threshold values a and b, using the logic given below (Figure1).

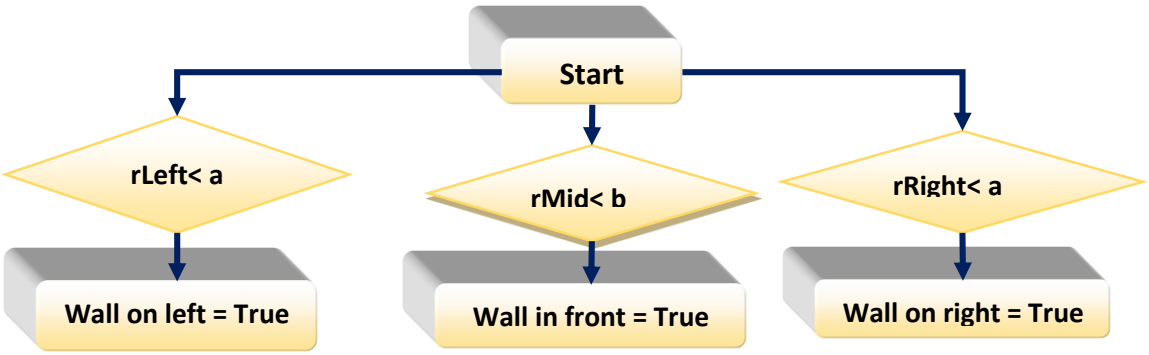


Figure 35: Presence of Wall

Here a is the distance of the walls in the sides and b is the distance of the wall in front. The values were chosen as a=2.35 feet and b=2.4 feet. Each frame cycle takes about 1890 ms to detect the intersection and move the robot. The robot can move up to 10.8 inches in 1 second, thus in one cycle it can move up to 20.4 inches i.e. about 1.7 feet. So the wall threshold values were set at a value higher than 1.7 feet.

After the presence of walls has been determined. The following logic can be used to identify the type of intersection.

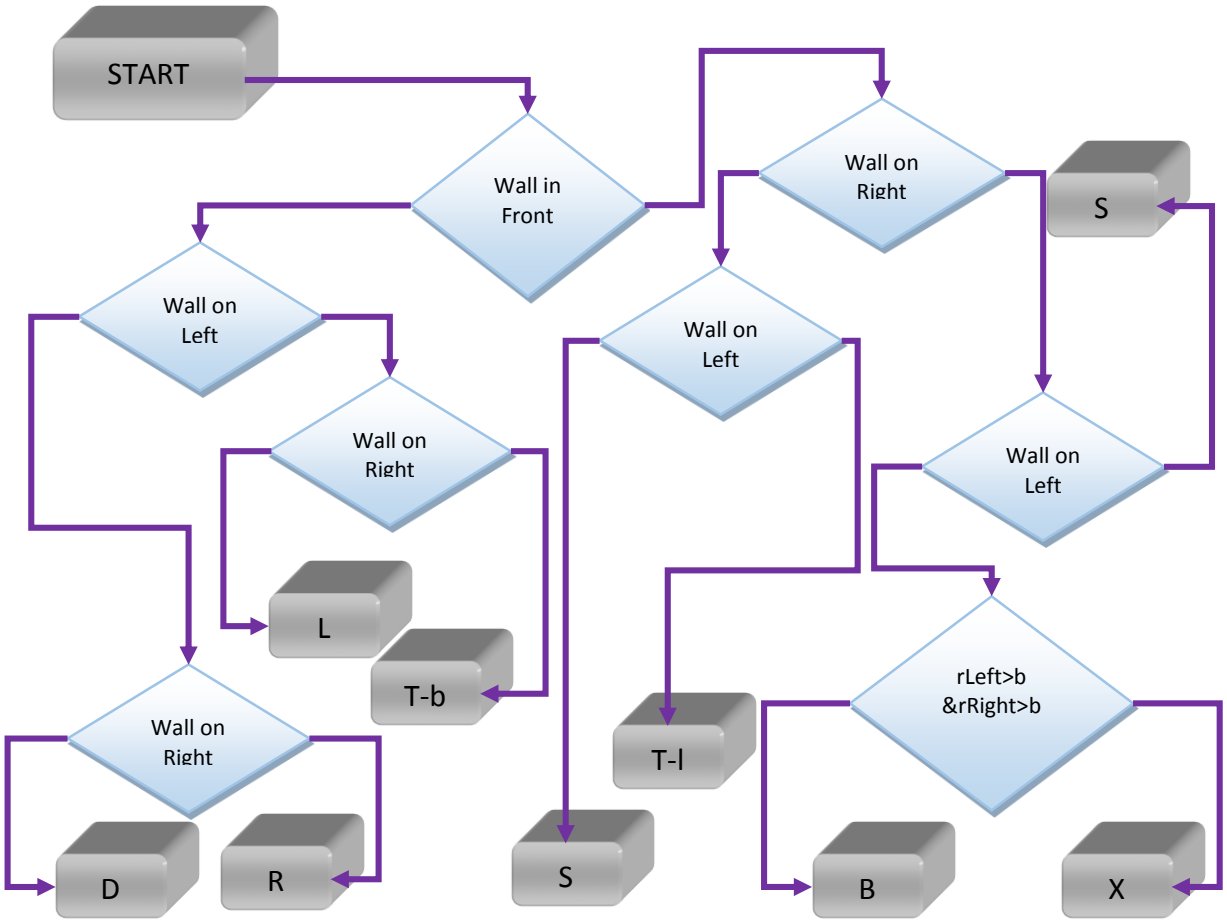


Figure 36: Intersection Detection

Navigation

For the 9 intersection types, the following table shows the possible navigation decisions.

Dead End	Reverse
Right Turn	Right
Left Turn	Left
T Intersection1-bottom	Left, Right, Stop, Reverse
Straight Hallway	Straight
T Intersection2-right	Left, Straight ,Stop ,Reverse
T Intersection2-left	Right, Straight, Stop, Reverse
B intersection	Straight
X intersection	Right, Straight, Left, Reverse

For multiple navigation options the first option has been set as the navigation command for the robot. If the systems involves path planning then the best option for reaching the destination can be chosen.

Motor Control

On the motorcontroller, channel 5 controlled the right wheels and channel 1 controlled the left wheels. The neutral value for left wheels was found to be 760 and the neutral value for right wheels was found to be 758. To steer the robot forward the right channel value was set 800 and the left channel value was set to 720. The ramp value was set to a constant value of 0 for forward steering. To generate a 90 degree turn in the left direction the right wheel value is set to 840 and the left

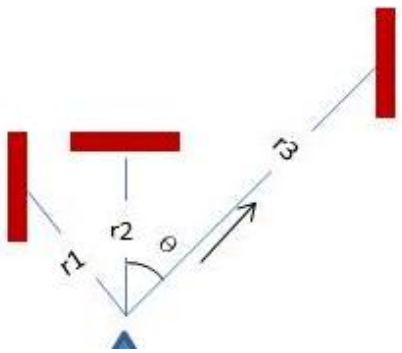


Figure 37: Turn Angle

wheel value is set to 920. To generate a 90 degree right turn the right wheel value is set to 510 and the left wheel value is set 600. Since, the ramp value controls the delay between the signals, it can be used to control the turn angle.

In the above scenario the robot is expected to turn right. The farthest an obstacle/ wall has been detected is at a distance of r_3. The turn angle Θ can be approximated as $\cos^{-1}\frac{r_2}{r_3}$. . The ramp value of 15 causes the least rotation, thus the ramp value function can be set as:

$$ramp = 20 - 15 * \theta/90^{\circ}$$

Chapter 6

Results

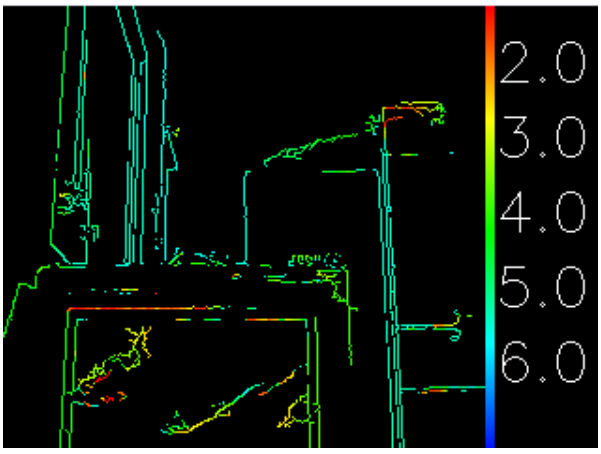
Depth Result

Box Detection

A box was placed at approximately 4 feet away from the robot and following results were observed

Robot Placed 4 feet away from a box

Right Camera Captured Image



Fused Map

Figure 38: Box Detection

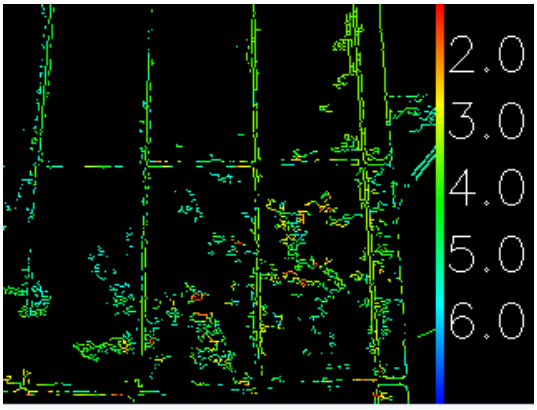
The robot was able to detect the distance of the box as approximately 4 feet, with some noise.

Detection of Wall

Most stereo algorithms fail to get disparity results for scenes with lower differences in pixel intensity, i.e. when the scene has predominately the same color. Thus, they cannot get good disparity results for walls which are usually the same color throughout the scene. Since, the stereo algorithm used is a feature based algorithm, good disparity results for walls can be found. To verify this, the robot was placed 4 feet away from a wall and the following results were found.

Robot Placed 4 feet away from a box

Right Camera Captured Image



Fused Map

Figure 39: Detection of Wall

As seen above the wall surface color is homogeneous, but the robot was able to determine its presence at a distance of 4 feet.

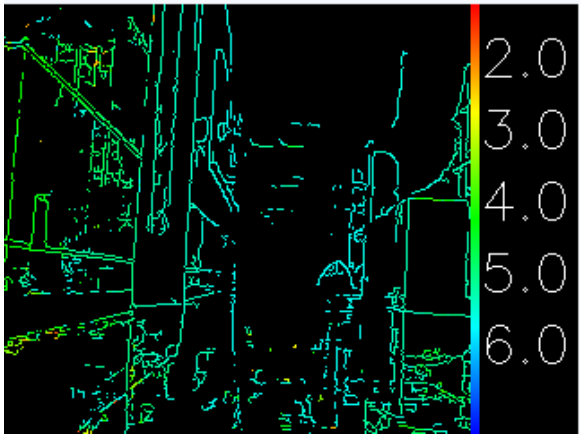
Intersection Detection

The robot was tested for the detection of all the major intersections. The following results were observed for all the scenarios.

Straight Hallway

Robot

Camera Captured Image



Fused Map

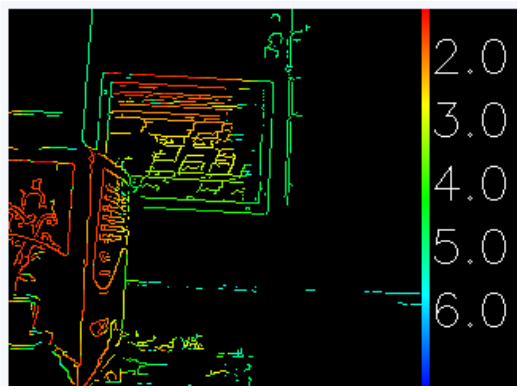
range (2.67519,3.05356,2.91005)

Figure 40: Straight Hallway

Right Turn

Robot

Captured Image



Fused Map

range (1.48992,1.92825,2.58562)

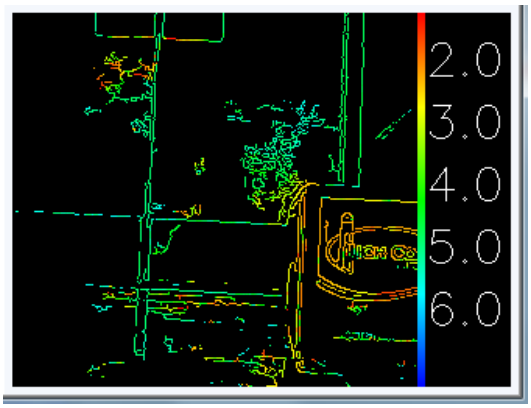
Figure 41: Right Turn

Left Turn

Robot

Captured Image

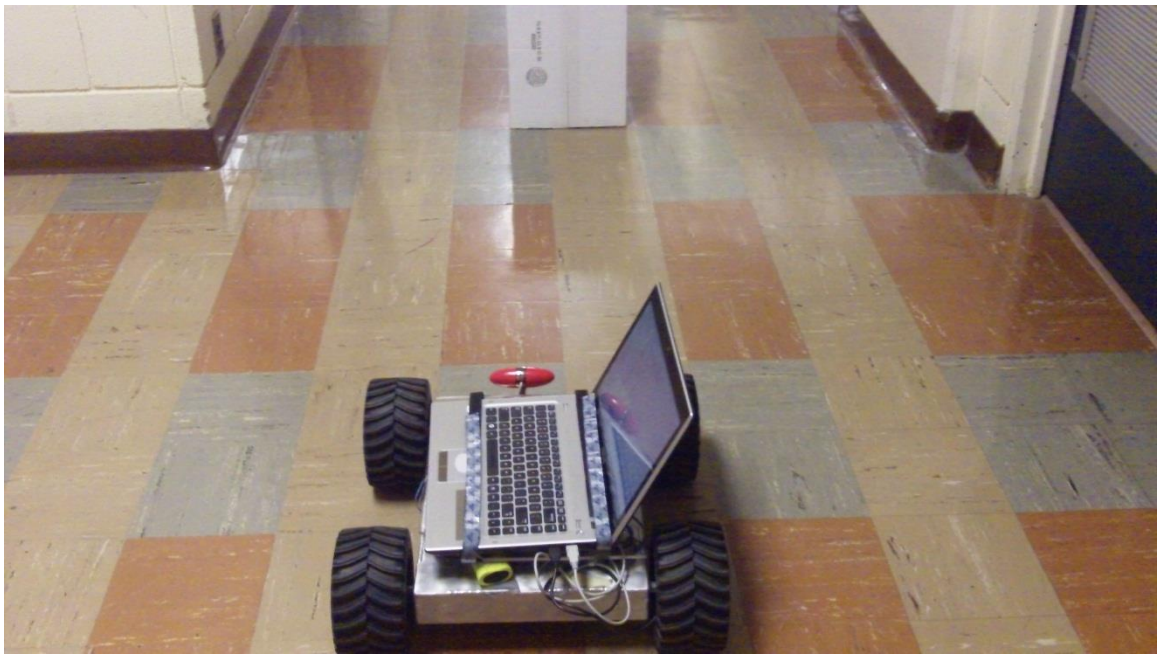
range (2.56923,2.25796,1.75998)



Fused Map

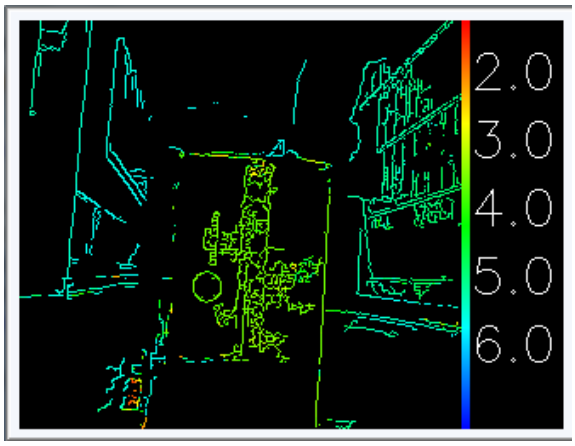
Figure 42: Left Turn

T intersection-type 1



Robot

Captured Image



Fused Map

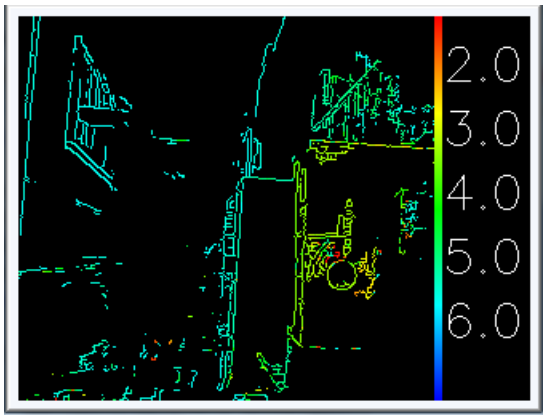
range (2.82836,2.20558,2.45762)

Figure 43: T Intersection type 1

T intersection-type 2

Robot

Captured Image



Fused Map

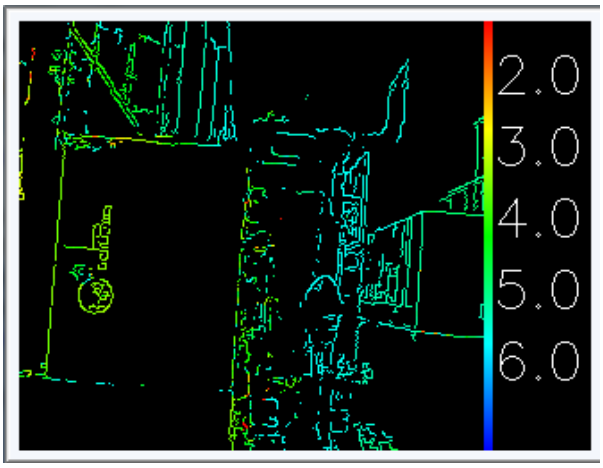
range (3.15416,2.97446,1.89046)

Figure 44: T intersection type 2

T intersection-type 3

Robot

Captured Image

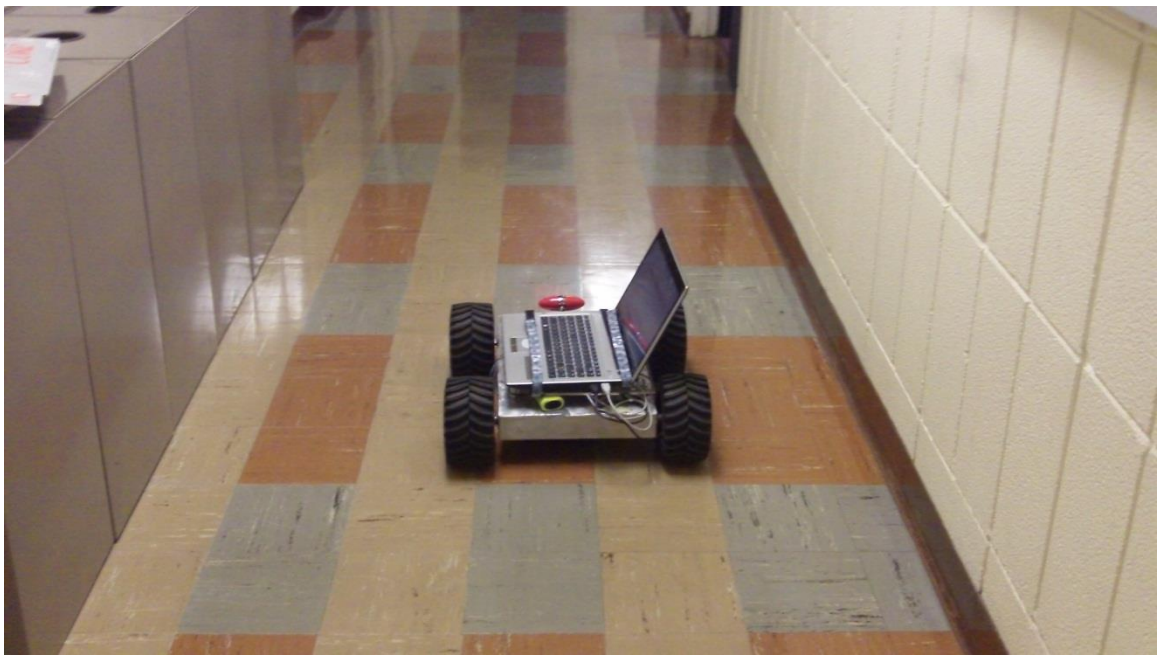


Fused Map

range (2.32005,2.81291,2.85479)

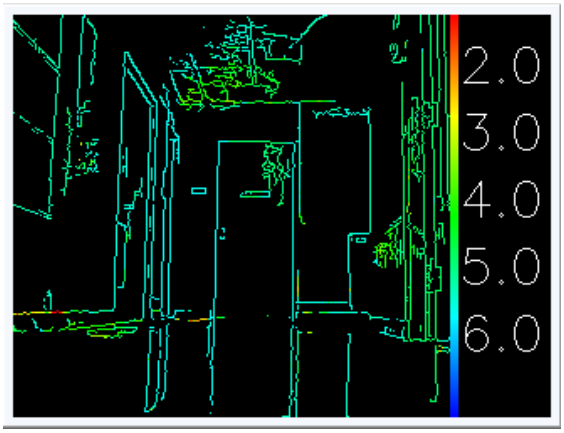
Figure 45: T intersection Type 3

X intersection



Robot

Captured Image

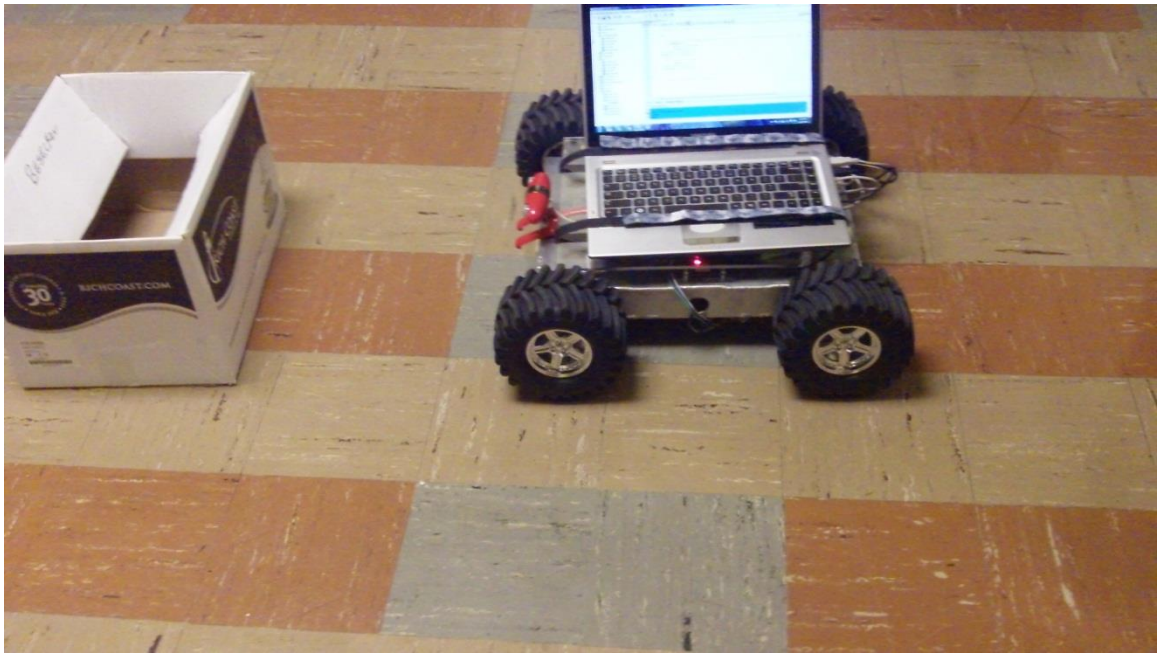


Fused Map

range (2.72041,2.63201,2.44729)

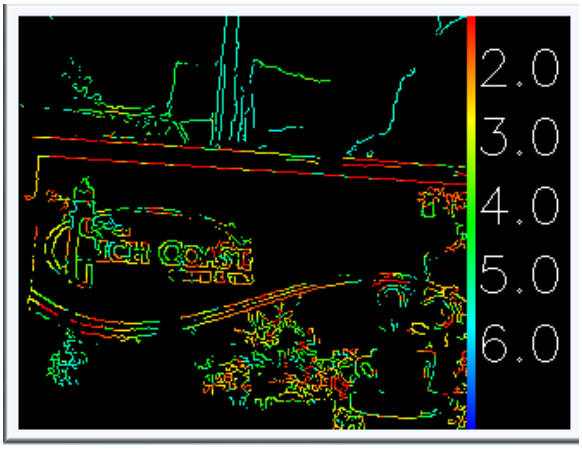
Figure 46: X intersection

Dead End



Robot

Captured Image



Fused Map

range (1.76518,1.84297,1.96085)

Figure 47: Dead End

Obstacle Avoidance

range (2.97487,2.86753,2.26535)
navigate
left
intersection type is 2

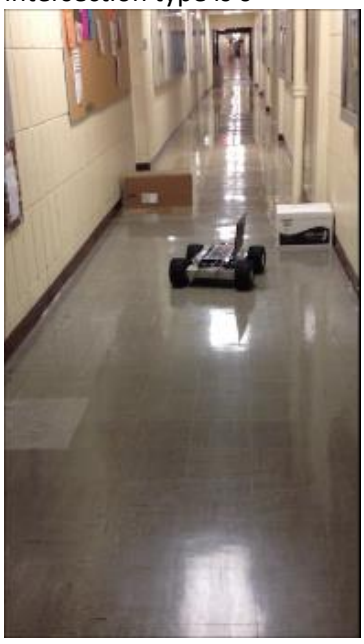
range (2.71313,2.88023,2.5124)
navigate
straight
intersection type is 8

range (2.34844,2.67778,2.6919)
navigate
Straight
intersection type is 3

range (2.30829,2.38454,2.85787)
navigate
right
intersection type is 0

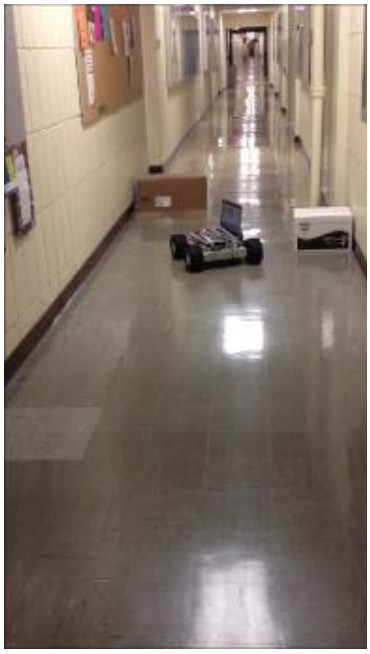
range (2.34033,2.36359,2.89018)
navigate
right
intersection type is 0

range (3.45602,2.72929,1.73259)
navigate
left
intersection type is 2

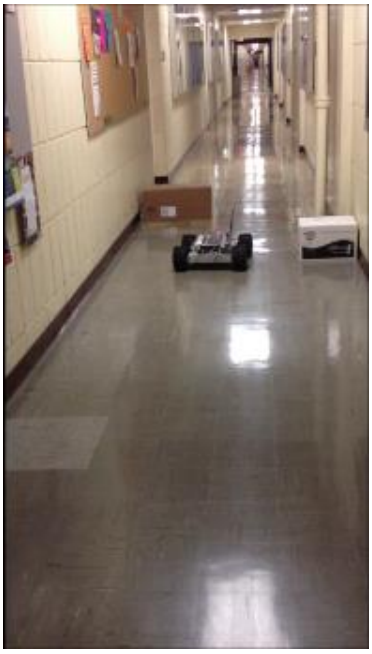


range (1.99534,2.26112,1.95972)
navigate
reverse
intersection type is 7

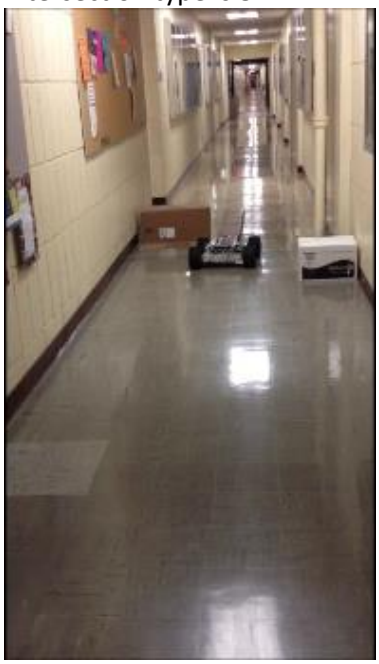
range (2.89045,2.43332,1.81817)
navigate
left
intersection type is 1



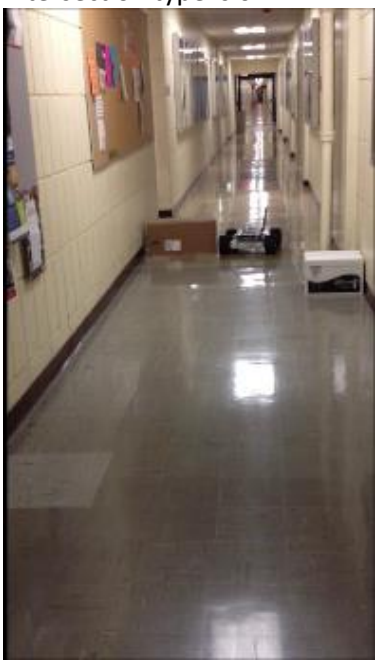
range (2.686,2.96693,2.52877)
navigate
straight
intersection type is 8



range (2.00306,2.25456,2.45371)
navigate
right
intersection type is 0



range (2.05456,2.12253,2.53867)
navigate
right
intersection type is 0



range (2.45314,2.99173,2.40491)
navigate
straight
intersection type is 6

Figure 48: Obstacle Avoidance

Wall

range (2.45907,2.45586,2.34468)
navigate
left
intersection type is 2

range (2.84244,2.529,2.80177)
navigate
straight
intersection type is 8

range (2.73115,2.3055,2.60127)
navigate
left
intersection type is 4

range (2.01428,2.33301,2.30866)
navigate
reverse
intersection type is 7

range (2.45708,2.16785,2.33543)
navigate
left
intersection type is 1

range (2.57782,2.00825,2.29083)
navigate
left
intersection type is 1

range (1.73746,1.75346,1.72)
navigate
reverse
intersection type is 7

range (3.05126,2.35944,1.80833)
navigate
left
intersection type is 1

range (inf,1.50772,2.09715)
navigate
left
intersection type is 1

range (2.76098,1.99549,3.18434)
navigate
right
intersection type is 4

range (3.29186,2.19869,2.35572)
navigate
left
intersection type is 1

range (2.60479,3.00969,1.90729)
navigate
Straight
intersection type is 2

range (2.90312,2.84283,1.68461)
navigate
left
intersection type is 2

range (2.90814,2.97449,2.00054)
navigate
Straight
intersection type is 2

Figure 49: Wall

Higher Resolution

At resolution of 640 x 480 the depth range is much higher than that of 320 x 240. The robot can detect objects as far as 14 feet but using higher resolution can be computationally very taxing. Each frame cycle takes about 10 seconds for computation; this can cause errors in synchronization of real position of the robot and the one evaluated by the camera. The advantage is that that due to higher range the presence of walls or obstacle can be detected earlier during the course.

In this scenario, a box was placed at an approximate distance of 10 feet from the robot and the course of action taken by the robot was recorded. The range values for the fuzzy logic code for detecting intersection was set as 7 feet as the range for the wall in front and 6.5 feet as the range for wall in left or right direction. The course of action taken by robot can be seen below.

Frame 1
Range= (6.8,7.1,7.5,7.1,6.3) ft.
Left turn
T intersection type 2 (left)

Frame 2
Range= (6.9,7.1,7.8,7.2,6.1) ft.
Left turn
T intersection type 2 (left)

Frame 4
Range= (7.2,7.3,7.1,7.1,6.1) ft.
Left turn
Left

Frame 5
Range= (7.2,7.0,6.9,7.0,6.7) ft.
Right turn
T intersection bottom

Frame 5
Range=(6.7,7.1,7.3,6.8,6.5) ft.
Left turn
Left

Frame 6
Range= (7.6,7.9,8.3,7.6,6.6) ft.
Left turn
T intersection type 2 (left)

Frame 7
Range=(7.3,9.0,7.1,7.6,6.2) ft.
Left turn
Left

Frame 8
Range= (7.1,6.6,7.7,7.4,6.5) ft.
Left turn
T intersection type 2 (left)

Frame 9
Range (7.2,6.6,7.2,7.4,6.8) ft.
Right turnT intersection bottom

Frame 10
Range= (6.7,7.7,7.3,7.3,7.1) ft.
Right turnT intersection bottom

Frame 11
Range= (7.2,6.6,6.6,6.6,6.8) ft.
Right turn T intersection bottom

Frame 12
Range (7.1,6.8,6.7,7.0,7.0) ft.
Right turn T intersection bottom

Figure 50: Course of Action for Obstacle on Right

Discussion

The vision system showed excellent results for obstacle detection and the system is a good choice as a close range sensor. The system was able to detect all the major intersections in close range. On lower camera resolution the obstacles/ walls are detected much later in comparison to higher resolution system. This is problem, when the robot is trying to navigate to avoid colliding into large walls. It needs to be aware of the wall much earlier, so that it can make the necessary amount of turn in time. In case of multiple navigation options, the robot behavior is erratic. It avoids colliding into walls and obstacles but it keeps going in loops. If we mix a path planning algorithm or a map generating module with the present system, we can get the desired robot motion. For increasing the range of the vision system, a stereo camera with greater baseline distance will be beneficial. The vision system can become more robust, by adding other sensors such as LIDAR or IR, etc. Since the camera view angle is limited to only 45 degrees, adding other range sensors can help the robot infer more about its surroundings and it will have the capacity to detect intersections at a greater range.

Chapter 7

Conclusion

A low cost feasible stereo system was developed for an indoor robot. The robot was successful in maneuvering through hallways under constant light. A Minoru stereo camera was used as the vision sensors and it was concluded that it is an efficient low cost close range sensor. Most stereo algorithms show excellent results for stereo images but exhibit ineffective results for raw images from camera. Also, algorithms can be computationally extremely taxing for real time systems. Feature-based algorithms work better on non-rectified images as they compare features rather than individual pixels. For the present system, the shape and stereo correspondence method was found to be the most accurate and computationally easy. As an edge detector the Canny edge detector is more accurate compared to Sobel in noisy conditions. A camera resolution of 320x240 pixels gives a lower range of measured distance compared to what is obtained at the resolution 640 x 480 pixels. Thus, at higher resolutions the obstacles can be detected from a farther distance. The processing speed for resolution at 640 x 480 pixels is about 10 seconds per frame and for resolution at 320x240 pixels is less than 2 seconds. Thus, selection of resolution is a tradeoff between computational power and expected range. The robot can move up to 1.7 feet on lower resolution system and rotate maximum 90 degrees in either left or right direction in each frame cycle.

Based on the test cases, the system is an excellent choice for close obstacle detection. In presence of large walls, the robot avoided collision with the walls but because of lack of predefined path, the robot moved in loops.

The system can be improved by adding path planning and map generating modules as described by S. D. Hanford, O. Janrathitikaran and L. N. Long for development of a robotic system to use the Soar cognitive architecture for the control of unmanned vehicles [31]. A greater range can also be achieved by using a

stereo camera with baseline cameras greater than 60 mm. Bumblebee stereo cameras have a baseline distance of 120 mm [32]. This gives a better range and can be used for detecting intersections earlier. Use of other sensors such as LIDAR (Light Detection and Ranging) and IR (Infra-Red) with the present system can increase the robot's awareness about its environment. This is because Minoru has a field view of 45 degrees, which limits its capability to measure distance of walls.

Thus, close ranged feature based stereo systems are efficient systems for autonomous indoor robot navigation.

Appendix A

C++ Code

CODE

Servo Configuration

This function gets and sets the properties of the serial port and sets the baud rate. The function takes the input of the Handle H, which is the handle representing the serial port and the function also takes the desired baud rate as the input. The function is called in the main function and the baud rate is set to 2400, because it allowed for communication between the microcontroller and the motor controller. The Parity is set to NOPARITY, the ByteSize is set to 8 and Stop Bits is set to ONESTOPBIT.

shortIntLowByte (short intmyInt)

This function separates the first byte of an 8 bit integer. This is done by using the following method:

myInt = myInt& 0x00FF

shortIntHighByte (short intmyInt)

This function separates the third byte of a 8 bit integer. This is done by the following method:

myInt = (myInt>> 8) & 0x00FF

voidsetServo(HANDLE serial_port, char ch,charramp,shortint pw)

voidsetServo(HANDLE serial_port, char ch,charramp,shortint pw)

This method sends the byte signals as an array to the motorconroller. Channel number (ch), ramp value (ramp) and 16 bit word that corresponds to the desired servo position (pw). Using the functions lowByte and highByte the first and the third byte of pw are obtained. An array with the following elements [‘!’, ‘S’, ‘C’, ch, ramp, lowbyte, highbyte, ‘\x0D’] is sent to through this method to the microcontroller. This further controls the microcontroller and subsequently the navigation of the robot. The following table shows the

PW values for each motion. The values were generates using trial and error. Also, due to the alignment and orientation of the wheels on the robot, the values for pw for both channels are not same

inthallwaylogic(HANDLE H, float rLeft, float rMid, float rRight)

This function uses the three range values to determine the intersections and calls the setServo() function for the different intersections. The function takes the input of the HANDLE representing the serial port, the first, third and the fifth range values.

The first step for detecting intersections is to determine the presence of walls, which is done using the logic given in chapter 5.

void gray2color(HANDLE H,IplImage*image, IplImage*imagecolor,IplImage*legendbar)

This function converts the black and white disparity map into a colored map. It also converts the disparity values into distance and finds the five ranges.

In the stereo algorithm the disparity ranges 9 to 55 for resolution of 320 x 240 and 18 to 110 for resolution 640x 480. . In OpenCV, the value for the intensity of pixels is between 0 and 1. Using this disparity to distance conversation can be given as:

Distance=(380*0.1982/(46*(bwpixel [i] [j])+9))

To find the ranges, the image was divided into five regions along the width. The distance values were averaged in each of the regions, to obtain a range value in each of the region.

Fused (IplImage*img,IplImage*disp,IplImage*cannyImage,IplImage*fuse)

Edges were detected on the left image from the stereo camera, using the Canny Edge detector method in the OpenCV library. The stereo result was super imposed on the edge map to generate a fused image.

int main (intargc, char** argv)

The main function initialized the image windows and variables; it started the camera and the microcontroller port. In the camera loop the stereo matching code was called and functions to create fused image, detect intersections and navigate, etc. were called.

Bibliography

[1] G. A. Bekey, Autonomous Robots: From Biological Inspiration to Implementation and Control, The MIT Press.

[2] S. Trihatmo and R. Jarvis, *Short-Safe Compromise Path for Mobile Robot Navigation in A Dynamic Unknown Environment,* 1994.

[3] D. R. Murray and J. J. Littele, "Using Real-Time Stereo Vision for Mobile Robot Navigation," *Autonomous Robots,* vol. 8, no. 2, pp. 161-171, April 2000.

[4] P. Oskar, "Stereoscopic Robot Vision System," in *XIII Industrial Systems Scientific Conference '05*, Herceg Novi, Serbia and Montenegro, 2005.

[5] N. Hautière, R. Labayrade, R. Perrollaz and D. Aubert, "Road Scene Analysis by Stereovision : a Robust and Quasi-Dense Approach," in *IEEE International Conference on Control Automation Robotics and Vision (ICARV '06)*, Singapore, 2006.

[6] I. Kostavelis, L. Nalpantidis and A. Gasteratos, "Real-Time Algorithm for Obstacle Avoidance Using a Stereoscopic Camera Production.," in *Third Panhellenic Scientific Student Conference on Informatics*, 2009.

[7] "Product Description All Terrain Robot," [Online]. Available: http://www.superdroidrobots.com/ATR.aspx..

[8] Parallax, Inc., "Parallax Servo Controller – USB (# 28823) Rev B 16-Channel Servo Control with a USB Interface," 2008. [Online]. Available: http://www.parallax.com/dl/docs/prod/motors/PSCusbManBv3_3.pdf.

[9] SuperDriod Robots, "Sabertooth 2x12 User ’ s Guide Mixed and independent options," Nov 2010. [Online]. Available: http://www.dimensionengineering.com/datasheets/Sabertooth2x12.pdf.

[10] Battery Description, *Product Description.*

[11] alibaba Group, "Minoru 3D Webcam User ' s Guide," alibaba Group, [Online]. Available: http://www.voipvoice.com/downloads/Minoru/userguide_en.pdf.

[12] E. Chan, *Presentation on Minoru Webcams.*

[13] OpenCVDev. Team, "OpenCV Documentation," [Online]. Available: http://docs.opencv.org/modules/core/doc/intro.html..

[14] Intel Corporation, "OpenCV processing and Java Library," Intel, [Online]. Available: http://ubaa.net/shared/processing/opencv/.

[15] G. Adam, *Introduction to Programming with OpenCV,* Illinois, 2006.

[16] A. Ogale and J. Domke, "OpenVIS3dOpen Source 3D Vision Library," [Online]. Available: http://code.google.com/p/openvis3d/.

[17] S. Florczyk, Robot Vision: Video-based Indoor Exploration with Autonomous and Mobile Robots, KGaA, Weinheim: WILEY-VCH Verlag Gmbh & Co, 2005, p. 216.

[18] A. S. Ogale and Y. Aloimonos, "Shape and the stereo correspondence problem," *International Journal of Computer Vision,* 2005.

[19] K. Zhang, L. Jiangbo, G. Lafruit, R. Lauwereins and L. V. Gool, "Robust Stereo Matching with fast Normalized Cross Correlation over Shape-Adaptive Regions," in *16th IEEE Internation Conference on Image Processing (ICIP)*, 2009.

[20] H. Hirschmüller and D. Scharstein, "Evaluation of Stereo Matching Costs on Images with Radiometric Difference," *IEEE Transaction on Pattern Analysis and Machine Intelligence,* vol. 31, no. 9, September 2009.

[21] W. T. Freeman and M. F. Tappen, "Comparison of graph cuts with belief propagation for stereo, using identical MRF parameters," in *Proceedings Ninth IEEE International Conference on Computer Vision,*, 2003..

[22] S. Ahuja, "SSDR2L," [Online]. Available: http://siddhantahuja.wordpress.com/category/stereo-vision/.

[23] S. Birchfield and C. Tomasi, "A Pixel Dissimilarity Measure that is Insensitive to Image Sampling," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 20, no. 4, pp. 401-406, April 1998.

[24] D. Scharstein, P. Ugbabe and R. Szeliski, "Stereo Data Sets," [Online]. Available: http://vision.middlebury.edu/stereo/data/scenes2001/.

[25] S. M. Lankton, "3D Vision with Stereo Disparity," [Online]. Available: http://www.shawnlankton.com/2007/12/3d-vision-with-stereo-disparity/.

[26] S. Jain, J. Cho, X. D. Pham, K. M. Lee, S. K. Park and M. Kim, "FPGA Design and Implementation of a Real-Time Stereo Vision System," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 20, no. 1, pp. 15-26, Jan 2010.

[27] Trucco and e. a. Jain , "Edge Detection," in *Edge Detection*.

[28] R. Kimmel and A. M. Bruckstein, "Regularized Laplacian Zero Crossings as Optimal Edge Integrators," *International Journal of Computer Vision,* vol. 53, no. 3, p. 225–243, 2003.

[29] R. Maini and H. Aggarwal, "Study and Comparison of Various Image Edge Detection Techniques," *International Journal of Image Processing (IJIP),* vol. 147002, no. 3, p. 1–12, February 2009.

[30] P. Kalra, *Canny Edge Detection,* Delhi, 2009, p. 1–7.

[31] S. D. Hanford, O. Janrathitikaran and L. N. Long, "Control of Mobile Robots Using the Soar Cognitive Architecture," *Journal of Aerospace Computing, Information, and Communication,* vol. 6, no. 2, 2009.

[32] P. Grey, "Bumblebee 2," [Online]. Available: http://www.ptgrey.com/products/bumblebee2/bumblebee2_stereo_camera.asp.