CS-308-2014 Final Report

Pathfinder

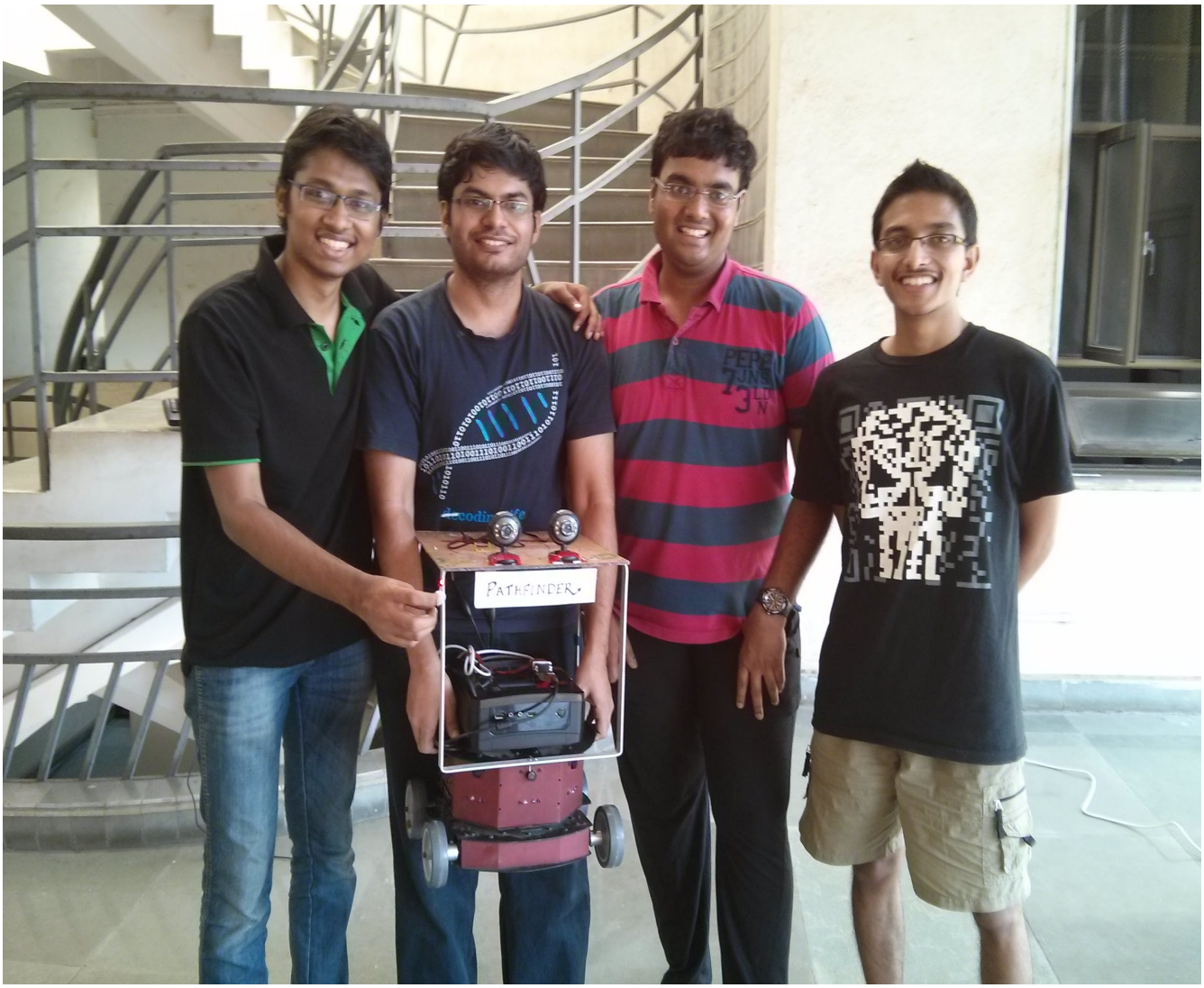
TU-05

Devendra Singh Chaplot (100050033)

Ashwin Paranjape (100050056)

Piyush Kumar (100050023)

Umang Mathur (100050012)



The PathFinder Team

Table of Contents

1. Introduction

Several automated tasks require the bot to be present at a precise location to perform the task. However, there is no direct method of transmitting interaction with environment directly to the bot. Indirect methods include indicating the object through a camera, explicitly mentioning the coordinates, wheel encoder data or triangulation techniques which might not be accurate enough or practical under those settings. We aim to change this by allowing the user to interact directly with the environment and making the bot capable of recognizing the interaction and reacting accordingly. We use a bright spot produced using a laser beam or Infrared LED and imaging techniques using two cameras to accurately compute location of a the target and move the bot to the target.

2. Problem Statement

The bot navigates to a given location in the arena specially demarcated using a bright spot produced using a laser beam or LED. The system uses imaging techniques using two cameras to accurately compute location of a the target to move the bot to the target.

As defined in the introduction, the system is a high-accuracy navigation bot driven by a well-lit target. Consequently, the primal system functionality and performance would be bench marked on :

1) precision of identification of the target location
2) precision in calculating the 3D coordinates of target location
3) precision in movement to the target location

3. Requirements

3.1 Functional Requirements

- **Camera Module** : The two cameras individually identify the target location (bright red point) with high accuracy.
- **Calculation of distance** : From the two images, identification of target's location is accurate accounting to relative angle between cameras' line of sight, and subpixel level of accuracy.
- **Serial Communication :** The CPU (linux box) attached to the FB-VI bot processes the coordinates of the target location using the image processing module. After this, an appropriate message needs to be sent to the motor drivers of the mechanical part . This is done using serial communication (USB to UART0) . An appropriate protocol has been designed that ensures that the bot moves the exact amount of distance
- **Motion of the bot** : Bot is able to move to a close vicinity of the target location computed by the bot based on two images captured in the previous step.

3.2 Non-Functional Requirements

- **Camera Resolution and Image Quality** - The resolution of the camera needs to be good enough so as to detect the hue and the brightness of the red point. At the same time, it should not be too high. A very high resolution leads to a large sized captured image, thus adversely affecting the processing time.
- **USB bandwidth** - The USB bandwidth in the associated hardware should be sufficient. A low bandwidth implies slow speed of processing, and thus slow debugging. Note that, at a time, the bot should be capable of handling 2 cameras, a keyboard and a mouse (for the purpose of debugging the code)
- **Weight balancing** - A good balance of weight must be maintained on the 2 wheels. An improper balance leads to different magnitude of frictional forces on the two wheels. This can lead to different angular acceleration when the same amount of current is supplied to the 2 motor drivers.
- **Friction coefficient of the wheels and the surface** - The friction force on the 2 wheels must be sufficient so as to drive the bot with the require speed. Also, a different coefficient of friction leads to a difference in speeds of the 2 wheels.
- **Laser v/s LED** - An informed choice must be made between the different choices of lighting media, namely LEDs and Laser pointer. The decision must take into account various parameters such as price, brightness, re-usability, lifetime, environment and the application.
- **Robustness of Image Processing Module** - The Image processing module should be robust enough so as to incorporate minor changes in the environment, flickering and change in the brightness values. Also, the module should allow for calibration of the threshold values in a different environment, and with different hardware.
- **Robustness of the Serial Communication Module** - The serial communication module must allow for customizing the protocol used to communicate between the image processing module and the motion control module.
- **Code documentation and modularity** - The code must be well documented in an easy-to-read fashion. Also the code must be modular, so as to allow for customization and calliberation.
- **Free and OpenSource** - The IDE ‘LPCExpressp’ is freely available. Also, the OpenCV module is open-source
- **Ease of installation** - The LPCExpresso IDE is quite easy to install (by clicking on the setup executable). OpenCV module can be installed easily using a suitable package manager for the Operating System (for ex ‘Synaptics Package Manager’ in Ubuntu)

3.3 Harwdare Requirements

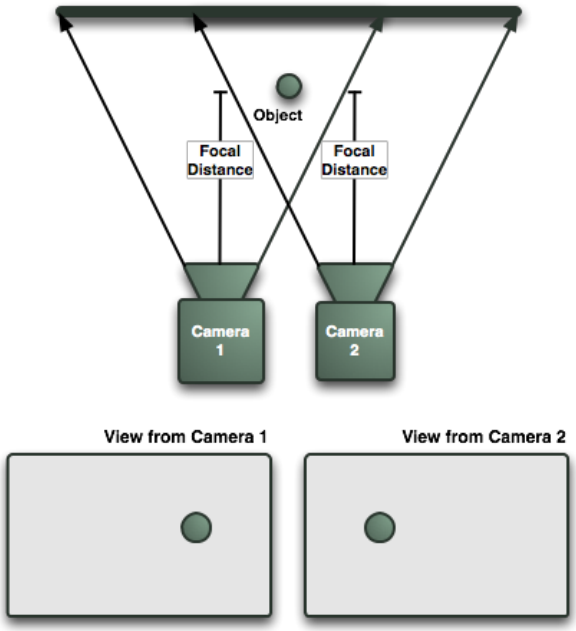
2 USB cameras
InfraRed LEDs and button cell
Laser Pointer
Firebird VI
Camera mounting rig for placing the camera at a height.
Serial to USB cables
Cable for loading binary file in FB-VI
Display Monitor (for debugging)

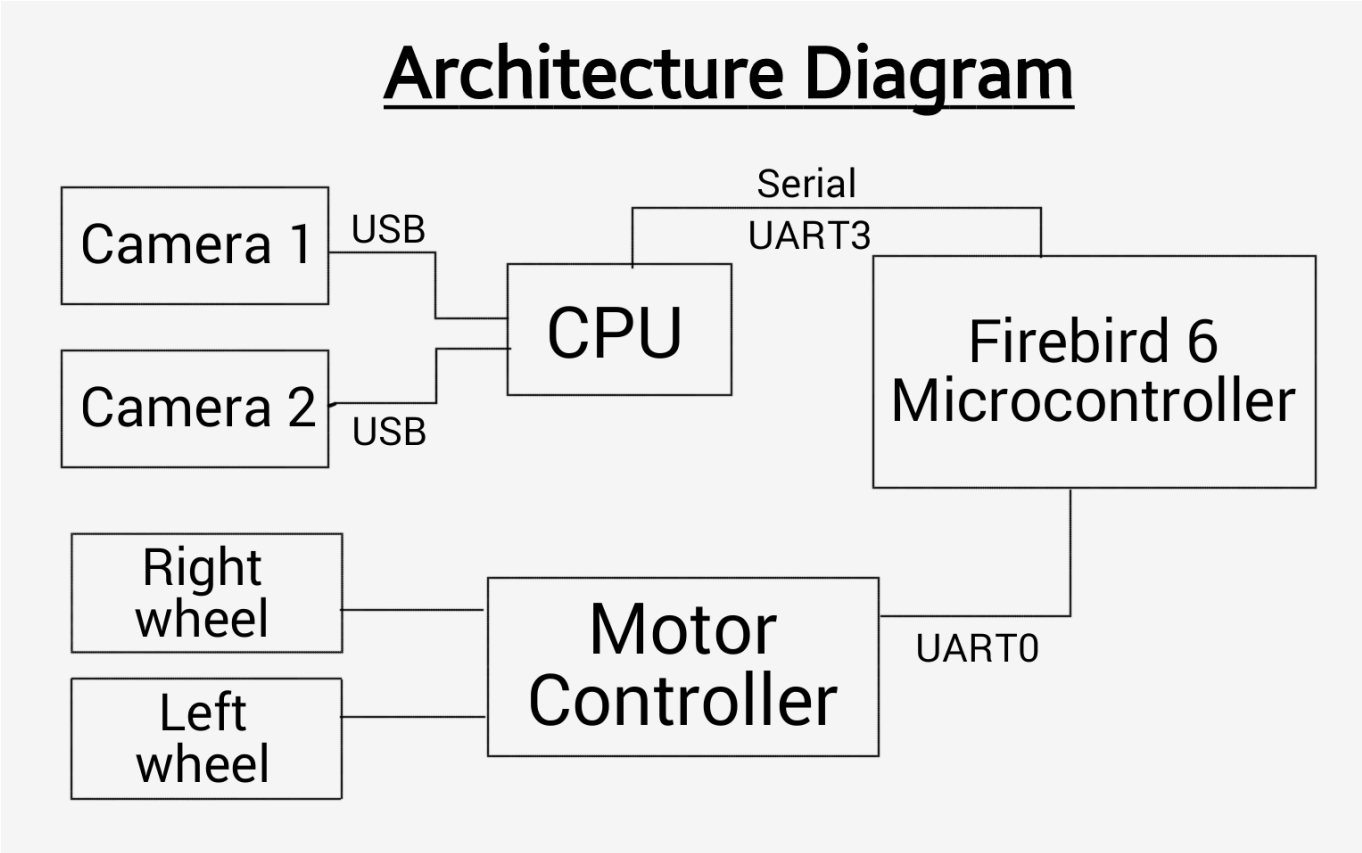
Keyboard (for debugging)
Mouse (for debugging)

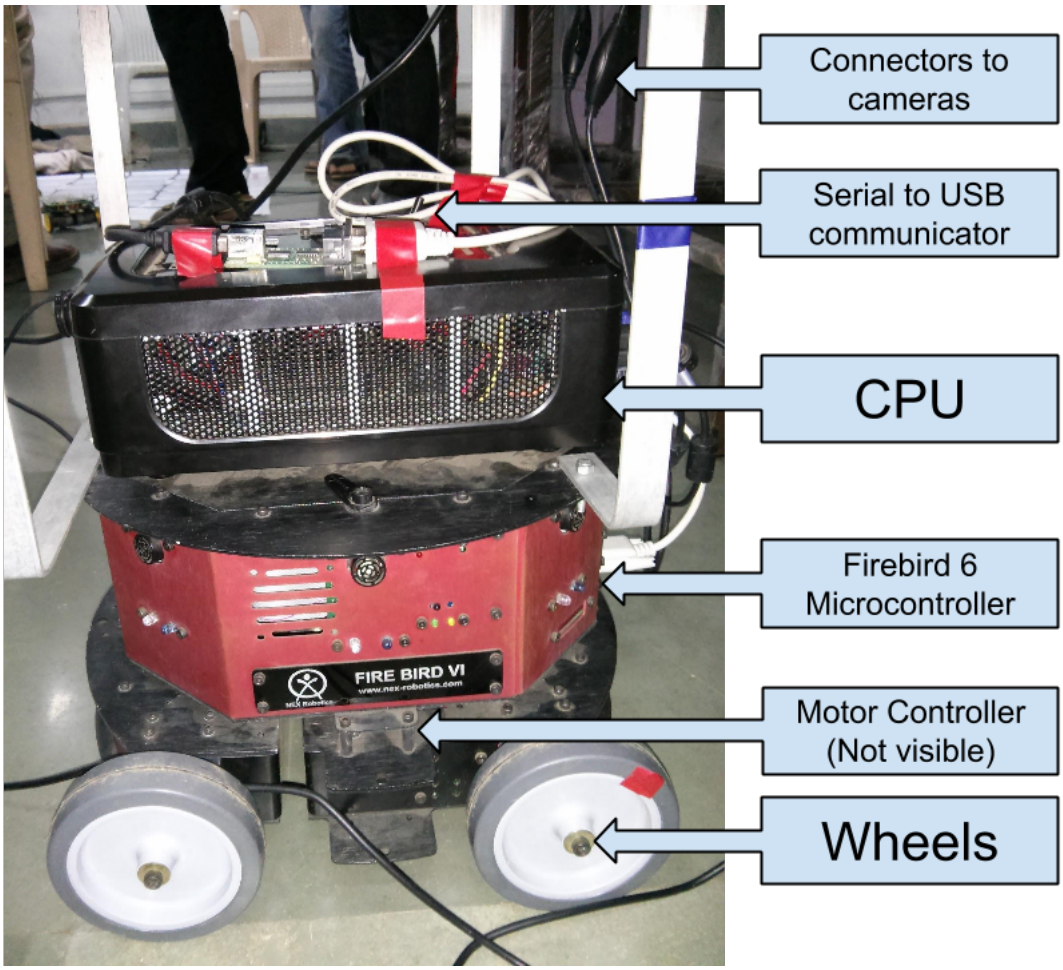
3.4 Software Requirements

OpenCV for C++ (g++)
Serial communication module in C (termio in Linux)
LPCExpresso IDE

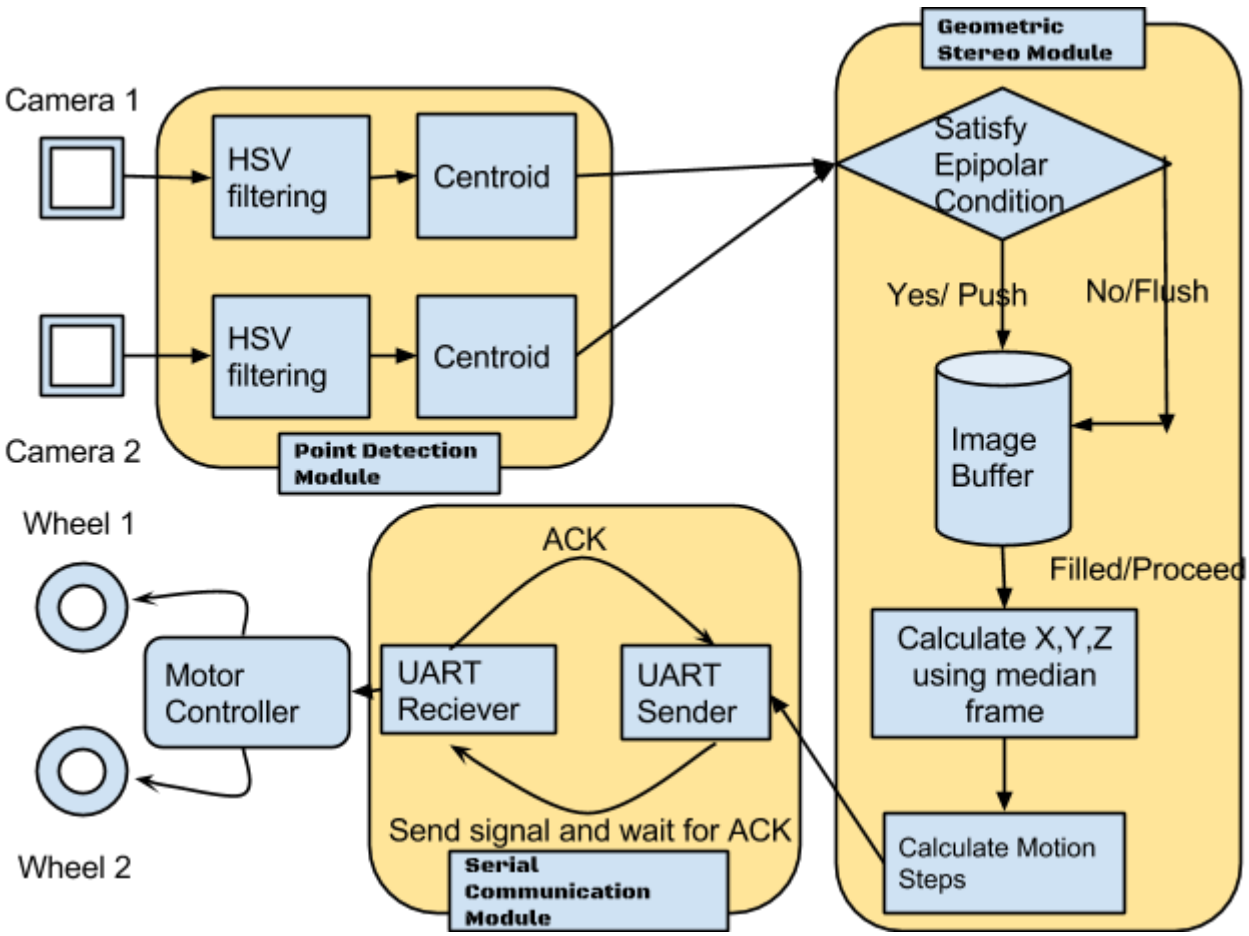
4. System Design




Architecture Diagram
Camera 1
USB
Serial
UART3
CPU
Firebird 6
Microcontroller
Camera 2
USB
Right
wheel
Motor
Controller
UART0
Left
wheel


Connectors to cameras
Serial to USB communicator
CPU
Firebird 6 Microcontroller
Motor Controller (Not visible)
Wheels
FIRE BIRD VI

FSM



5. Working of the System and Test results

Working of the system:

The procedure can be divided into following parts:

- Calibration of two cameras
- Marking the location where the task is to be performed
- Identification of target location by the bot
  - Identification of the target location in the video stream of the two cameras
  - Calculation of the 3D coordinates of the target location
- Moving the bot to target location

Calibration of two cameras

The two cameras are calibrated using a number of training points with known location. The intrinsic parameters of both cameras are also determined in this section. After calibration,

given a pair of corresponding points in images of two cameras, we find the 3D coordinate of that point.

Testing Strategy - In standard methods for camera calibration a few points are left out of the calculation. The ability of the camera to predict the lines in 3D on which they lie is a test of near perfect calibration.

Marking the location where the task is to be performed

Using LEDs - The bot detects the LED using cameras and move to that location. The LEDs are turned on using a remote control. The user is provided with a GUI showing locations where the LEDs are installed. User can choose one among these locations. The advantage of using LEDs is bright point in the image which essentially means easy identification of target location in the image. The disadvantage is that the LED needs to preinstalled at the target location manually with power requirements

Using Laser Pointer - The user points to the desired location using a laser pointer. The advantage of Laser pointer is that the target location can be easily marked even after fixing the location of Laser Pointer. The downside is difficulty in identification of this location in camera image.

Testing Strategy - The LEDs should appropriately light up and be visible in the camera and/or the laser pointer should be bright enough to be visible in the camera. The points should not be farther than 5m.

Identification of target location by the bot

Identification of the target location in the video stream of the two cameras

We first defocus the cameras a little to get the effect of gaussian blur without the computational overhead. Then we filter based on HSV ranges in HSV space. We have sliders which can be used to tune the filtering for a given environment. Once we have a reasonable filter, we take the centroid of the filtered point as the target location for each camera

Testing Strategy - The image processing code should be able to identify the image coordinates of the marked point with reasonable accuracy.

Calculation of the 3D coordinates the target location

The calibration gives us a function F which takes (x,y) coordinates of target location in two images and give us 3D coordinates of the target location.

$$F (p1 (x , y) , p2 (x , y)) = (X , Y , Z)$$

where p1 and p2 are coordinates of target location in images of two cameras.

The final formula obtained is the following:

$$Z = \frac{t}{\left|\tan\left(\theta - \tan^{-1}\left(\frac{x_2}{f}\right)\right) + \frac{x_1}{f}\right|}$$

where t is distance between 2 cameras

theta is angle between 2 cameras

x1 and x2 are image coordinates of target location

f is focal length of both cameras

X is then calculated as

$$X = 0.5\,(x1 + x2)\,Z\,/\,f$$

The bot has two cameras to pinpoint the exact location of the LED including the depth. In other words the X,Y,Z coordinates relative to the bot.

Testing Strategy - The bot should be able to output the X,Y,Z coordinates of the marked point in the bot's coordinate system with reasonable accuracy. The point should not be further than 5 meters and should be clearly visible given the resolution of the camera provided. The bright spot should also be in the line of sight of the bot.

Moving the bot to target location : Serial communication and motion control

The robot is not expected to do path planning, however should dodge minor obstacles and reorient itself toward the objective.

After the first calculation of target's (X,Y,Z) coordinates, the bot first moves Z distance forward, then turns 90 degrees (right or left depending upon whether X is positive or negative) and then moves X distance forward.

The FB-VI bot is supposed to manouver to the target lit point. For this, the exact computed coordinates of the target location must be appropriately communicated to the motor driver in terms of the distance to be moved and the angle of turning. In essence, the target coordinates can be communicated as a tuple (d1,d2), where d1 represents the distance (in inches) to be moved before taking a turn, while d2 is the distance after taking the turn (moving from (x1,z1) to (x2,z2) is equivalent to moving from (x1,z1) to (x1,z2) followed by (x1,z2) to (x2,z2). A single message has been encoded as a 8-bit unsigned integer . The MSB of the instruction specifies whether to move or to rotate. The remaining 7 bits encode the amount of motion. For forward motion, the 7-bit value is the distance in inches. For rotation, the 7th bit represents the sign while the last 6 bits represent the magnitude (in degrees) of the angle to be rotated.

For example, the message (move forward 24 inches) will be encoded as 00011000

[0-0011000 : forward-24]

Message (rotate by -30 degrees) will be encoded as 11011110
[1-1-011110 : Rotate-negative-30]

Testing Strategy - The bot should be able to navigate a few small obstacles of a size comparable to the bot and reach the desired location. The location should should be reachable from the bot. The bot reaches that location and displays on screen to indicate task completion

Test Results:

Table for test results

Sr. No.	Functional Requirement	Testing Strategy	Test Results
1.	Calibration of cameras	In standard methods for camera calibration a few points are left out of the calculation. The ability of the camera to predict the lines in 3D on which they lie is a test of near perfect calibration	Focal Length = 800 px and departure from parallel alignment is 1.1 degrees.
2.	Marking the location where the task is to be performed	The LEDs should appropriately light up and be visible in the camera and/or the laser pointer should be bright enough to be visible in the camera. The points should not be farther than 5m	Points even farther than 5m were satisfactorily marked using both LED and laser
3.	Identification of target location by the bot	The image processing code should be able to identify the image coordinates of the marked point with reasonable accuracy	The bot was able to successfully identify the target location for both LED. Although there were slight discrepancies in the point correspondence between the two cameras we corrected for it using the centroid of the detected points and taking a median of centroids over many frames
4.	Calculation of the 3D coordinates the target location	The bot should be able to output the X,Y,Z coordinates of the marked point in the bot's coordinate system with reasonable accuracy. The point should not be further than 5 meters and should be clearly visible given the resolution of the camera provided. The bright spot should also be in the line of sight of the bot.	Both LED and laser points were detected with a maximum error of about 7% (average error of 5 inches in 80). Points beyond 5 meters were also detected. Only caveat is that the bot needs to be recalibrate in different lighting conditions (which is reasonable). Also due to original hue and absorption of many surfaces laser did not work when shined on a few surfaces.
5.	Moving the bot to target location	The bot should be able to navigate a few small obstacles of a size comparable to the bot and reach the desired location. The location should should be reachable from the bot. The bot reaches that location and displays on screen to indicate task completion	The bot reaches the desired location with an error of around 5 inches. Obstacle avoidance was not the primary aim and hence was not focussed upon. Task completion status couldn't be displayed as the LCD of the bot was not working.

6. Discussion of System

a) What all components of your project worked as per plan?

1. Identification of laser point in images and calculation of its distance from the bot
2. Serial communication between linux box and Firebird VI master microcontroller
3. Bot was able to both move and turn in order to reach the target location.

b) What all components of your project did not work as per plan and how were they corrected?

1. The cameras were assumed to be parallel and that deviation from this assumption would not cause problems. However it turned out that deviation from this assumption even by a single degree caused a lot of discrepancies. Hence we corrected our model and took into account the deviation from this assumption in the form of a slider which can be used to caliberate every time the cameras might have moved slightly
2. The detection of LED was not working properly as we were filtering based on red color and finding the brightest point. We realized that filtering based on HSV was much more robust as it allowed us to filter based on a range of colors. With this we could detect LEDs as well as many instances of laser shined on surfaces.
3. Motor-microcontroller - The only way of controlling motors is to send messages through UART to microcontroller which in turn controls the motors. The communication protocol only allowed us to set the speed and the encoding of the wheels. However under increased weight it one of the wheels started spinning wildly, which was unexpected.
4. LED-display and serial communication - The connector between CPU and Firebird microcontroller stopped working ang we had to resort to an external serial communication cable
5. Loose connection and resets in the new serial to USB cable
6. Inexact motion estimates - As can be expected the motion estimates through wheel encoders were inexact and did not always correspond to the actual motion of the bot

c) What we added more than discussed in SRS?

1. In the SRS we were considering that the target point would be lit by a red LED, owing to the fact that a red LED when directly pointed to the camera will have substantial brightness and will be easily detected. But due to the detection algorithm being robust, we were able to take it to the next level and pointing to objects using a laser also works.

d) Changes made in plan from SRS:

1. Obstacle avoidance was not followed through - Primary reason was lack of time.
2. Auto-detection of the bright point : the target identification mentioned in the SRS was irrespective of its position wrt the field of view, so bot should have been able to turn and find target's outside its field of view (by rotation and sampling, etc). This aim was not achieved and the bot does not try to find targets that outside its view field.

7. Future Work

Possible extensions:

1. Obstacle Avoidance - The project can be extended to incorporate minimal obstacle avoidance. This means that advanced object detection be used to detect obstacles in the path and the bot plans optimal way of avoiding them on its way to the target.
2. The laser spot is not detected on all kinds of surfaces even with a high powered laser. To correct we need to use the difference of hues between the proposed spot and its surroundings.
3. Instead if a bot moving on the ground, we can make a drone moving in the air and tracking the laser pointer (and attack :P)
4. Functional extensions: Appropriate hardware can be used to extend the functionality of the bot. This means one can attach a programmable arm to the bot. Then, the laser can be pointed to a fruit in a green-house, towards which the bot moves and cuts the fruit. Similarly, one can extend the functionality to paint a spot on a wall.

8. Conclusion

The project was a very engaging task with challenges all along the way. There were many challenges which we anticipated and many others which we did not. For example we realized while integrating that the bot becomes very heavy on one side and the wheel starts behaving in an arbitrary manner. The micro controller which controls the motors can only be sent the speed and the number of counts and we were unable to control the motors separately given the amount of time we had. Had we integrated the bot earlier we would have realized the problem and taken corrective steps. Thus the biggest learning was never not to stop anticipating problems which an arise in real world modelling

9. References

1. Camera Calibration
    a. http://en.wikipedia.org/wiki/Camera_resectioning
    b. http://docs.opencv.org/doc/tutorials/calib3d/camera_calibration/camera_calibration.html
2. Geometric Stereo
    a. http://en.wikipedia.org/wiki/Epipolar_geometry
    b. http://docs.opencv.org/trunk/doc/py_tutorials/py_calib3d/py_epipolar_geometry/py_epipolar_geometry.html
3. Bright Spot Detection
    a. Hough Circle Transform in OpenCV http://docs.opencv.org/doc/tutorials/imgproc/imgtrans/hough_circle/hough_circle.html
    b. Laser spot detection demo http://www.youtube.com/watch?v=5Yq1t0SQdjg
4. Serial Communication in Linux (termio)
    a. http://en.wikibooks.org/wiki/Serial_Programming/termios
    b. http://en.wikibooks.org/wiki/Serial_Programming/Serial_Linux