Mitchell Barton

ECE 578

Homework 1

11.1.12

**Parallax Laser Range Finder (LRF) Development**

**Introduction**

For my project, I will be integrating a laser range finder sensor with the MCECS bot for navigation and object detection. The laser range finder has already been selected and is the Parallax LRF, which uses an OmniVision camera cube, an Arima laser diode, and a Parallax Propeller P8X32A-Q44 processor to process the images and make the necessary calculations which are described later. The goal is to use the LRF in conjunction with sonar sensors as well as the Microsoft Kinect. In this paper I will discuss the theory of operation of the LRF, its command set, the test setup and program used for initial testing, the initial testing results, problems encountered and planned solutions, and finally, our plans going forward.

**Theory of Operation**

The LRF works on the principle of optical triangulation. The camera focal plane and laser diode are separated by a fixed distance, h, as shown in figure 1. As the distance, D, to the target increases, so does the laser spot’s distance from the center of the focal plane array, or pixels from center (pfc). A linear relationship between pfc and angle Θ, of the form y=mx+b, is derived from experimental data at the factory. Once the laser spot is seen by the camera, Θ can be calculated and passed to the trigonometric function to determine D.

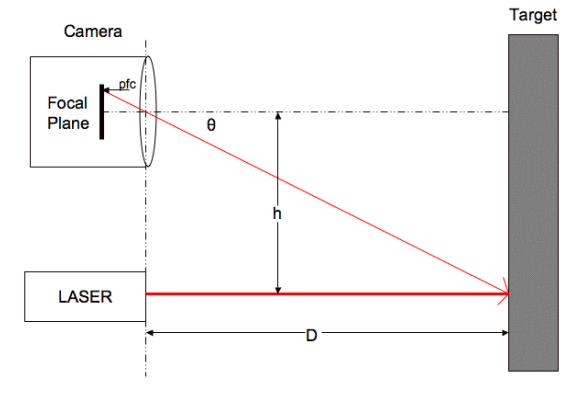


Figure 1 - Optical Triangulation

Additionally, the parallax propeller processor performs several image processing steps to improve the accuracy of the pfc measurement. They are:

1. Background subtraction – the frame grabber function captures a frame with the laser on, and a subsequent frame with the laser off, then subtracts them which results in very little detail other than the laser spot.
2. Thresholding – All pixels that are above a certain value, specifically coded as a luminance value of 0.3, are assigned the maximum luminance, 1, and the rest are set to 0, resulting in a white blob on a black background.
3. Column sum – A one dimensional array is created containing the sums of ‘1’ value pixels from the vertical columns. This is done to speed up the search for blobs.
4. Blob detection – Traverse the 1D array of column sums and determine those columns that are above a threshold value, 2 in this case. This is repeated for every element and more than 1 blob can be detected.
5. Mass and centroid calculation – The mass is the sum of all ‘1’ valued pixels within the previously defined blob. In the case of multiple blobs, the one with the largest mass is selected as the laser spot. The centroid calculation is a weighted average of the ‘1’ valued pixels within the blob, that is:

Sum = 1\*s1 + (2\*s2) + … (n\*sn), where sn is the column sum of blob column n, and

Centroid = Sum/Mass

**Command Set**

The LRF sends a receives data over a serial interface with 5V TTL level signals. The following commands are available, and there are more detailed descriptions in the manual available online:

* R – single range measurement which returns a 4 digit decimal value in mm
* B – single range measurement which returns a 4 byte binary value in mm
* L – repeated range measurement which simply loops the R command
* E – adjust camera for current lighting conditions
* S – reset to default settings
* V – print version info
* H – print list of available commands
* O – display coordinate, mass, and centroid data for all detected blobs
* X – calibrate camera system for range finding
* G – capture and send single frame (greyscale 160p x 128p)
* C – capture and send single frame (YUV422 640p x 16p)
* P – capture and send single background subtracted frame (YUV422 640p x 16p)

**Wiring Diagram and Serial Communication Program**

To communicate with the LRF in real time, I used an Arduino duemilanove to pass serial commands from the PC USB interface to the LRF, and the wiring diagram is shown in figure 2.

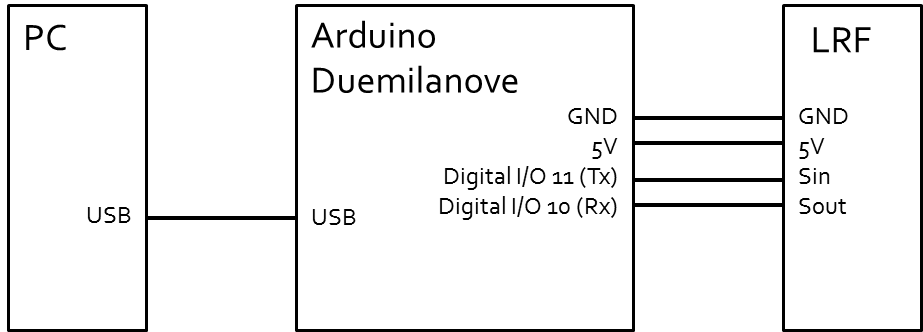


Figure 2 - Test Setup Wiring Diagram

Additionally, I wrote a simple program to accomplish the serial communication tasks using the Arduino soft serial library, and this is shown below:

/\*

Software serial multiple serial

Receives from the hardware serial, sends to software serial.

Receives from software serial, sends to hardware serial.

The circuit:

\* RX is digital pin 10 (connect to TX of other device)

\* TX is digital pin 11 (connect to RX of other device)

\*/

#include <SoftwareSerial.h>

SoftwareSerial LRFSerial(10, 11); // RX, TX

void setup()

{

// Open serial communications and wait for port to open:

Serial.begin(115200);

Serial.println("Hello World!");

// set the data rate for the SoftwareSerial port

LRFSerial.begin(9600);

}

void loop() // run over and over

{

if (LRFSerial.available())

Serial.write(LRFSerial.read());

if (Serial.available())

LRFSerial.write(Serial.read());

}

**Initial Testing**

The LRF is speced to provide ranges between 15-122cm, with an average error of 3%, but not to exceed 5%. I wanted to test this as well as other limitations with the setup shown in figure 3. I found that the minimum measurement was 16.5cm (6.5”). The maximum return I got was 242.3cm (95.4”) but this was very inaccurate. The stated maximum of 122cm was comparatively accurate. The resolution of the measurement appears to get larger as the distance gets larger, i.e. for 20cm the resolution is 0.1cm, and at 140cm it is 3.6cm.

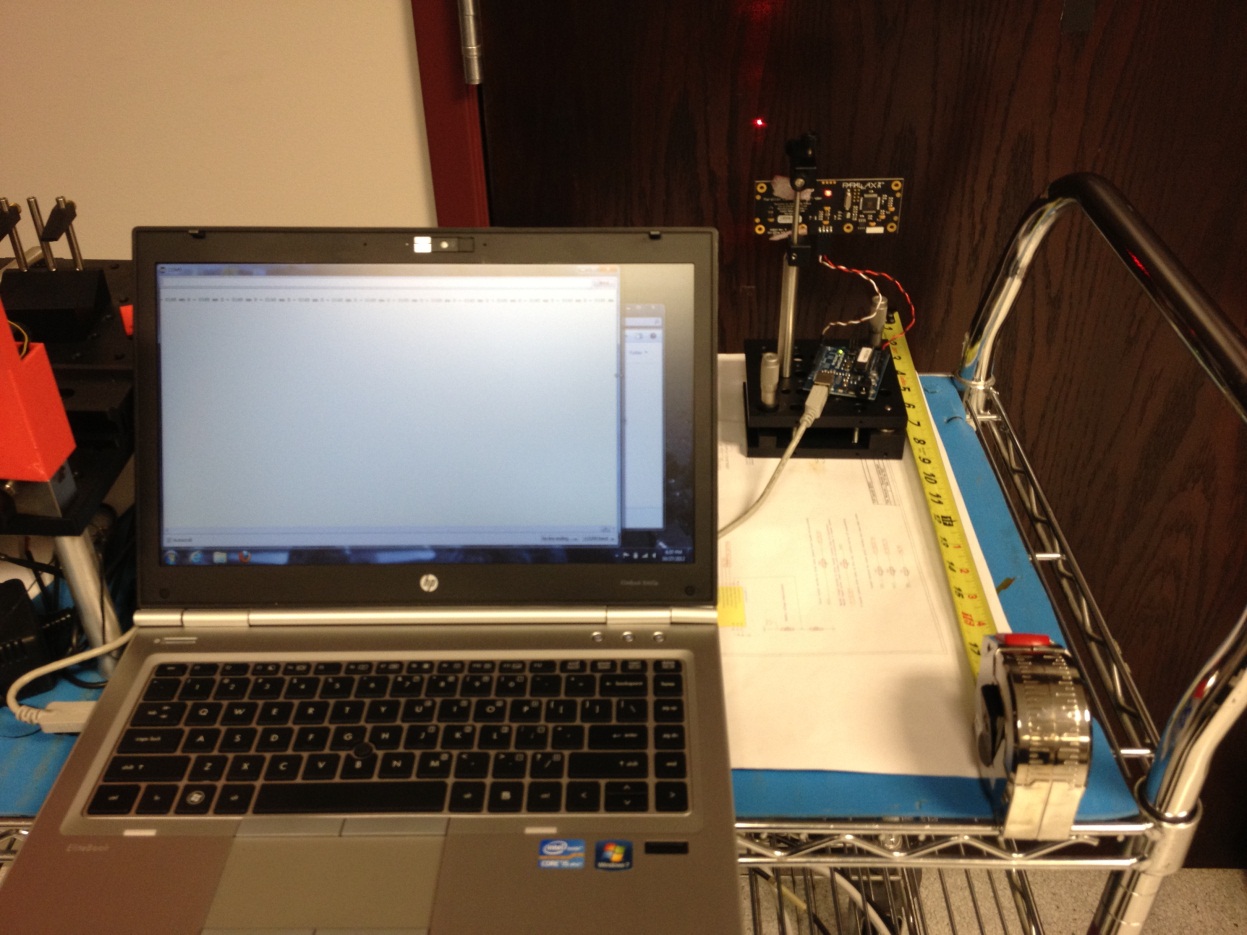


Figure 3 - Test Setup

I took several measurements at each of the 5 positions shown in figure 4, and there was very good repeatability. This was for a wooden door, but when I tried to measure this on a glossy white door the camera had to adjust before getting good repeatability. Also moving during the acquisition causes more errors and inaccuracy, primarily if the ‘laser on’ frame is much different than the ‘laser off’ frame. This means that the robot will need to limit motion during frame acquisition to get accurate distance measurements.

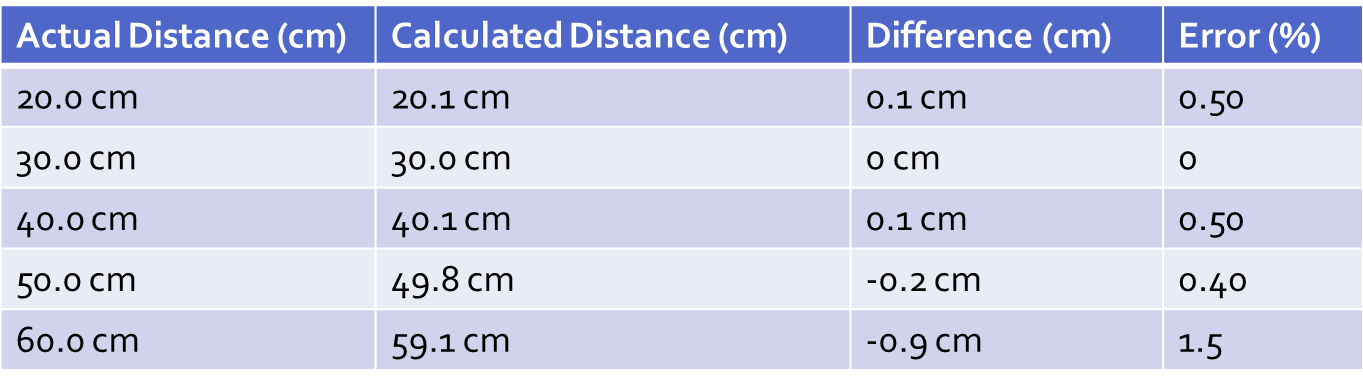


Figure 4 - Accuracy Data

Also, the measured distance did not change as it should have with angle, i.e. there is an insensitivity to angular offset, and it is less accurate in azimuth, that is, the plane in which the camera and laser are the same distance from the floor. In azimuth, I measured the same distance, 59.1cm, through an angle of 13.3°, after which the measurement jumped to 59.8cm. In elevation the angle was only 1.64° before the measurement again jumped to 59.8cm. This leads me to believe that the resolution is worse for angular variations in both elevation and azimuth, while accuracy degrades significantly more for angular variations in azimuth, which is to say when the camera and laser are not equidistant to the target. This requires more testing to understand fully.

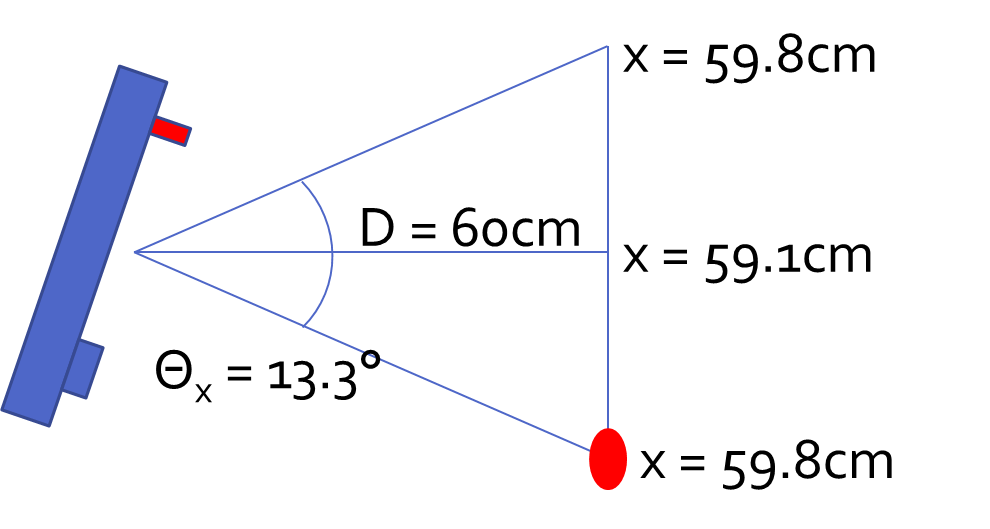


Figure 5 - Angular Accuracy Diagram

**Problems and Solutions**

The first problem I ran into was that the Arduino duemilanove software serial interface had limitations which made it unable to use the image viewer application that is available for the LRF. The software serial limits the data to a 64 byte buffer, and so the image application only gets some of the image when it is expecting the full image. Additionally, the Arduino software serial library seems unable to run at any baud rate other than 9600 with the LRF, however works well at that rate. The planned solution for this is to use an Arduino Mega which is available in the robot lab, because it has multiple hardware serial interfaces. This should allow the LRF viewer to receive all of the image data that it is expecting.

Another problem encountered was the variability in returns from different surfaces, as discussed in the initial testing section. There is some amount of time needed for the camera to readjust it’s gain and level for particular light and surface conditions, and this is stated as 10 seconds minimum in the manual. I did not have a chance to spend a lot of time characterizing this, but I believe it will be important to test the various surfaces in the building which the robot will operate in. I would envision several measurements, followed by a ‘E’ command which will tell the camera to readjust its gain and level, and then several more measurements for many different surfaces, particularly glossy or glass surfaces which will be more likely to generate erroneous blobs.

The last issue to discuss is how to mount and use the sensor. It has become apparent that there are angular inaccuracies and one way to overcome this is to take several different measurements at different angles, by mounting the sensor on a multi-position actuator of some sort. This is the method used by the robot navigation demo on the product homepage, referenced below. This will be something to consider as we continue testing and integrating. We also need to determine where on the robot we can mount this, and this will be one of our goals in the next few weeks as we build the MCECS bot up.

**Future Plans**

The initial testing has shown that this sensor will be very suitable for object detection, but can also be used for more precise navigation at small distances (17-122cm) due to its high accuracy in that range. There is the ability to use the camera as a blob detector as well, however this will require additional hardware, namely a propclip, as well as reprogramming of the propeller processor on the LRF module, which will not be the focus of our efforts this term.

We will be focusing on implementing a Kalman filter to improve the accuracy of range measurements with the LRF, eventually using the sonar sensors as well, for position estimation and mapping with the MCECS bot. We would like to first implement a simple filter which takes only the measurements from the LRF and has one degree of freedom for actuation, perhaps tracing a wall but that would likely have more degrees of freedom. We would like to show that the measurement is improved with the filter, so a before and after type of experiment is our goal. Once we have that working properly, then we could include the readings from the sonar as well so that we have an extended Kalman filter with various sensors as well as a model.

**References**

1. Parallax LRF Homepage; <http://www.parallax.com/Store/Sensors/ObjectDetection/tabid/176/ProductID/774/List/0/Default.aspx?SortField=ProductName,ProductName>
2. Arduino SoftSerial Homepage; <http://arduino.cc/en/Reference/SoftwareSerial>