PLazeR

a planar laser rangefinder

Robert Ying (ry2242)
“Derek” Xingzhou He (xh2187)
Peiqian Li (pl2521)
Minh Trang Nguyen (mnn2108)

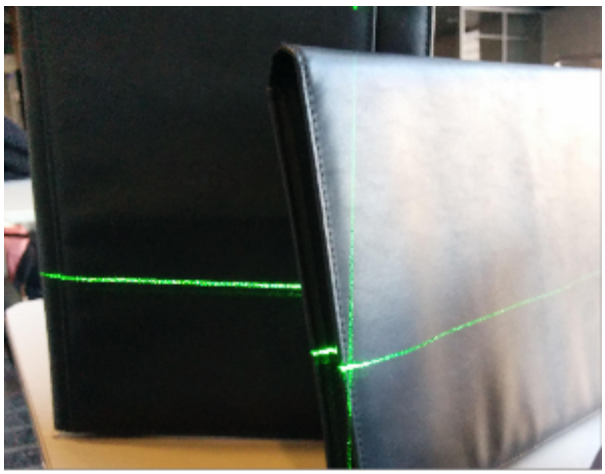
Overview & Motivation

Detecting the distance between a sensor and objects in a scene is a useful tool for machine perception and robotics such as modeling 3-D objects, finding target distance used in military purposes, and using it as a measuring tools instead of tape measuring.

This project is about determining the distance to an object using laser beam. Most rangefinders use either light or sound as their primary media, and then use triangulation or time of flight to determine distance.

We intend to use a planar laser to build a laser rangefinder that can simultaneously determine the distance to various objects in the scene.

As we can see in the image below, the distance of an object is related to the height of the projected laser line. We can detect the laser line at each column in the camera image, and then use that to generate a planar point cloud.



Laser beam visible in scene.

Architecture

This system takes the input data from a USB camera and calculates the distance of the object in question. The system can be divided into three main components: software, software-hardware interface, and hardware. User-space software will be used to read from the camera feed, convert the images to the correct format, and transmit it to the FPGA board that does the algorithm. After the FPGA processes the image to get the laser line at each column, the result is sent back to the software to be processed to a calculated distance. A kernel-module driver will monitor and control the transmission.

Software

The software programming that we use will be mainly on C. The software component of this project include these following steps:

Read the image data from the camera

The image from the camera will be feeded to the software program in the format of three 480 x 640 matrix representing the pixel information, one matrix for each channel color which are red, green, and blue. After that, those matrixes will be transferred to FPGA board which will complete the image processing and laser detecting process before feeding the laser line information back to the software. Having the processed image and the differential vector from hardware, the distance from the camera and the distance from the laser can be calculated using trigonometry.

Reading in the data interpretation from hardware

After converting the camera image into scene coordinates using camera intrinsic matrix, the data we received from hardware are the image profile and the camera parameter. The image profile includes two vectors: one is a vector with a size of 640 representing the horizontal distance of the laser beam to the y-axis, the second one is a vector with the size of 480 representing the height of the laser beam to the x-axis. The camera parameters are the horizontal distance from the laser to the camera, the distance from the camera to the wall, and Gaussian kernel mean and variance.

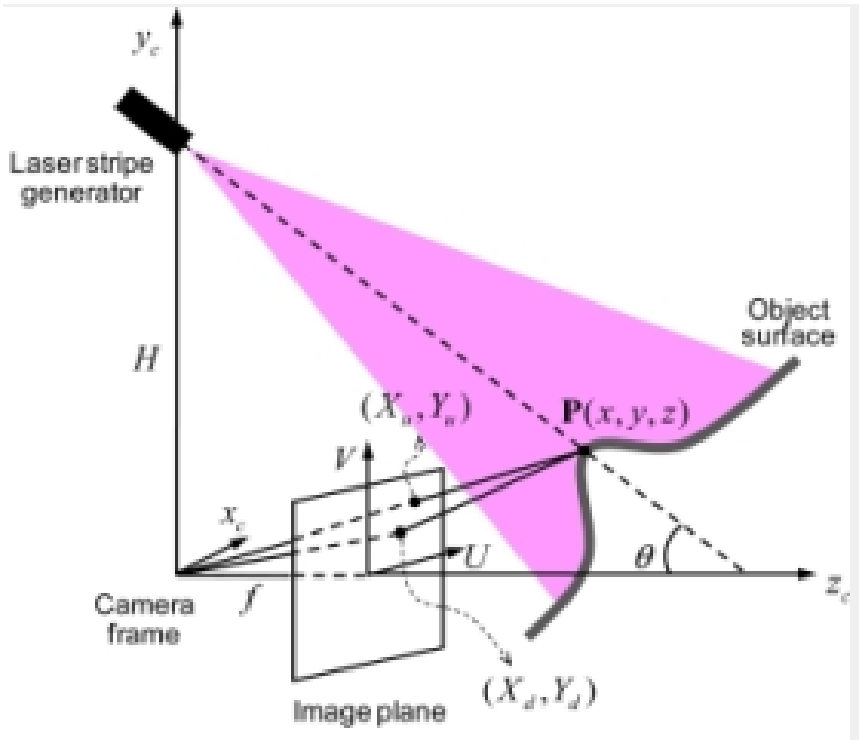
Solve for the angle θ

To calculate θ, we will use the image of the laser beam on the wall, or a flat surface. The angle will be calculated using simple linear interpolation.

Calculating the distance

Having the angle Theta calculated above and the distance from the laser to the camera, the distance of the object can be calculated using trigonometric equation.

$$d = (A^T A)^{-1} x$$



Calculating distance from the image

Hardware/Software Interface

uint8[640*480*3]	three channel RGB image data
bool	r/w flag
uint8	Gaussian filter bandwidth
uint8	laser threshold value
uint16[640]	y-axis differentials
uint16[480]	x-axis differentials
bool	ready flag

Timing:

```
vKernel modulev
[set r/w][write 640x480x3 bytes][unset r/w]------------------------------[read differentials]
------[unset ready]---------------[gaussian filter][threshold][average][set ready]----------
^FPGA^
```

Hardware

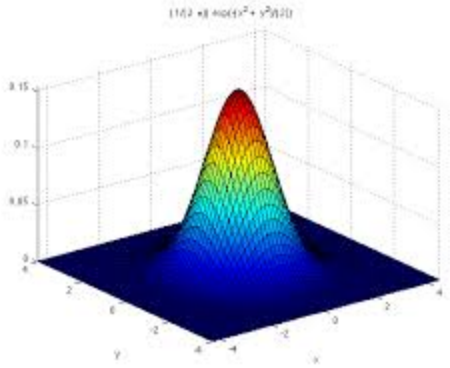
The primary purpose of the FPGA in this system is to implement the function

$$f: I \times \sigma \times \eta \rightarrow L$$

where I is the image space $\mathbb{Z}_+^{640x480}$, σ is the bandwidth of a Gaussian kernel, and η is the thresholding value for the desired value. The output of the function L is then the pixel-wise displacement between the calibrated laser line and the detected line position, represented as a vector in the space $\mathbb{Z}_+^{640}$.

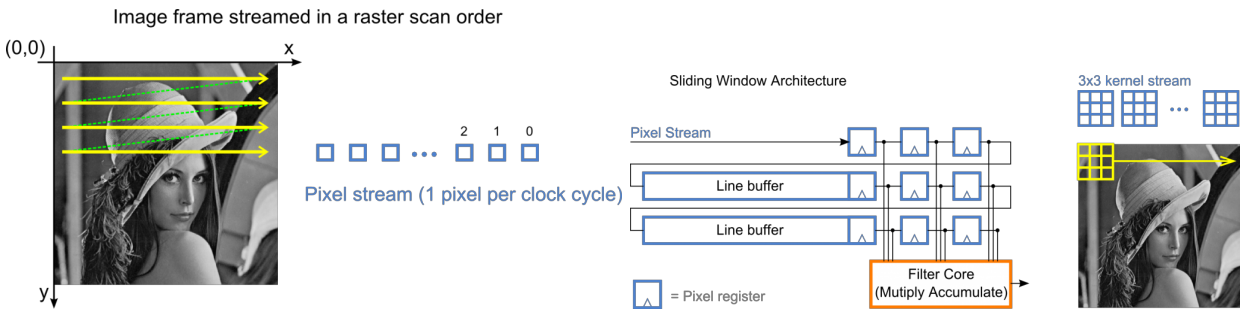
We implement this function as a two-dimensional convolution and other operations between the image and a fixed-size k by k kernel. In particular, we will only need to buffer the k^2 data points for all of the operations.

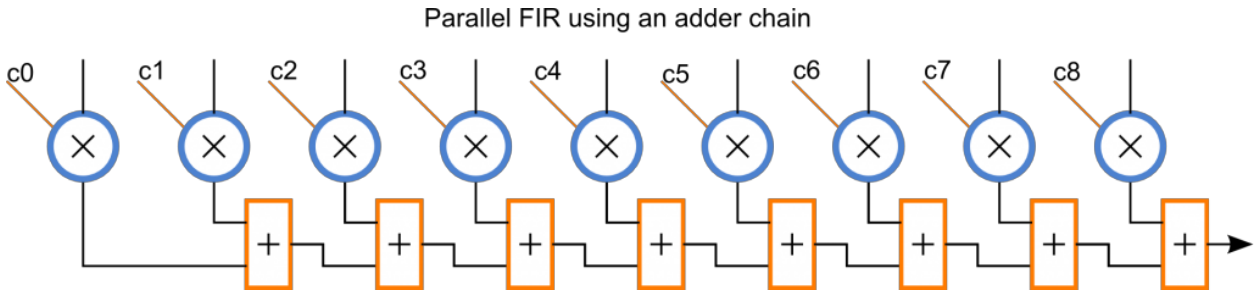
Step 1: Gaussian Blurring



As the camera is fairly low-cost, we will need to apply some preprocessing before the data can be analyzed. In particular, there is nontrivial sampling error and measurement error that results in a speckled noise pattern on the captured image. In hardware, we can efficiently eliminate the effect of this noise by convolving the image with a Gaussian kernel.

We can implement the convolution as follows (example given as a 3x3 kernel, though we would likely use 15x15 or other larger windows):





In essence, we will implement k rolling buffers of length k across the image, so that we can raster scan a k by k matrix to perform the convolution with.[1] This is fairly efficient, as we need only use k^2 multiplications; moreover, we can do these multiplications with integral values only, saving floating-point multipliers.

Step 2: Thresholding

The strength of the laser is such that it will saturate the camera. Therefore, both the horizontal and the vertical lines will appear as white in the image feed. We therefore can specify a threshold (its value to be calibrated and provided for the run) and run thresholding for each pixel to get the area of the lines. For this purpose, each of the pixel is simply compared with the thresholding constant stored in DRAM and provided by the software, resulting in a black & white binary signal for each pixel.

Step 3: Averaging

For each column of the above-mentioned binary image, we can determine its average point now by calculating:

$$n_j = \frac{1}{n} \sum_i i * m_{j,i}$$

on all columns *j*. This should be easy to implement on a FPGA board with arithmetic tools. The resulted array is returned to the software for processing.

---

[1] http://blog.teledynedalsa.com/2012/05/image-filtering-in-fpgas/

Milestones

Milestone 1

- Get software prototype working
- Design and test thresholding and averaging hardware

Milestone 2

- Design and test Gaussian convolution hardware
- Write the kernel module

Milestone 3

- System integration done
- Use for actual measurements and report the result

Reference

How to calculate distance
http://www.cse.unr.edu/~bebis/CS791E/Notes/CameraParameters.pdf
http://www.seattlerobotics.org/encoder/200110/vision.htm

Hardware implementation
http://www.rgshoup.com/prof/pubs/FPGA93.pdf