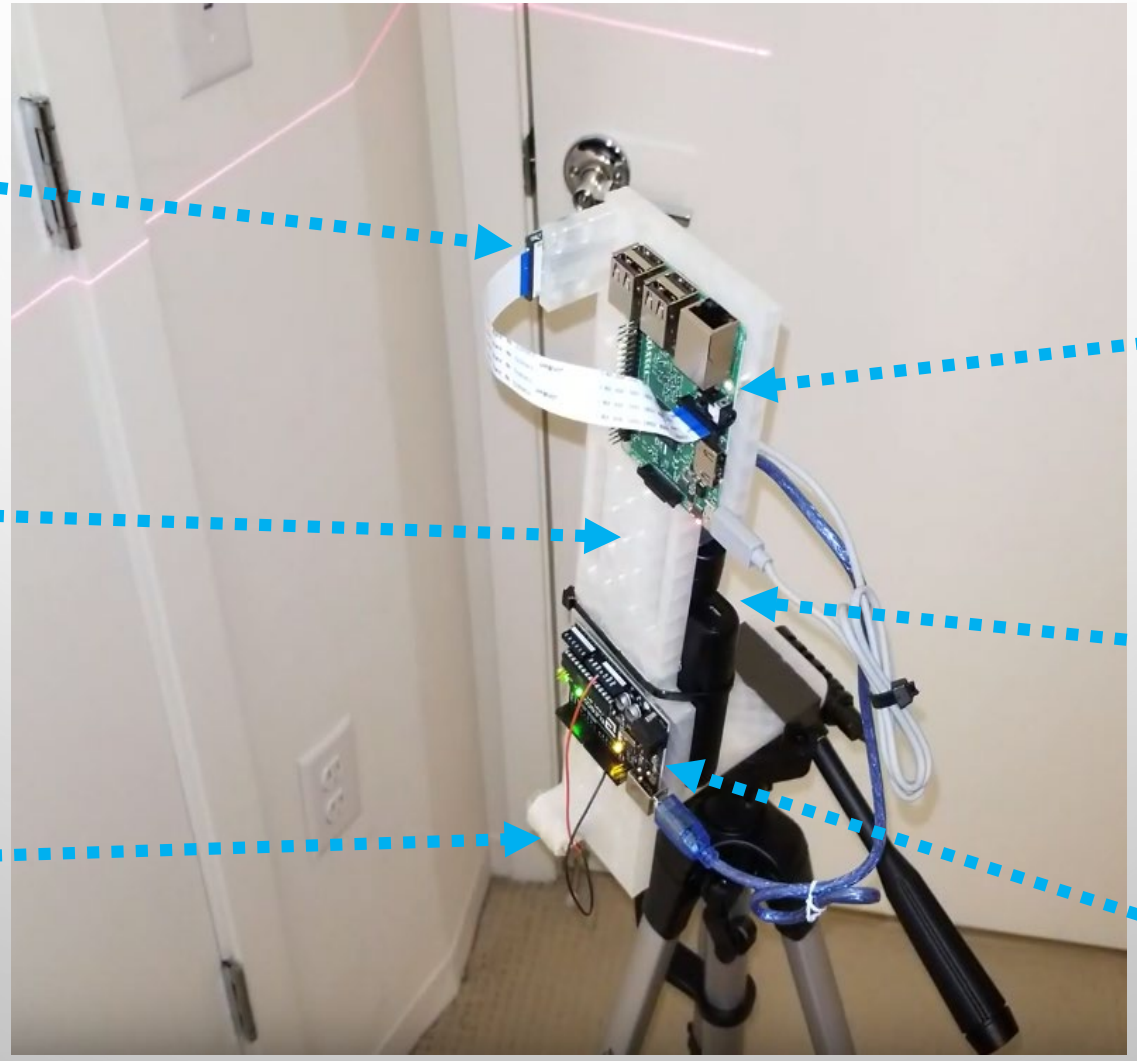
ENME489Y: REMOTE SENSING FINAL PROJECT PRESENTATION

AARON BARLEV

APRIL 30TH, 2019

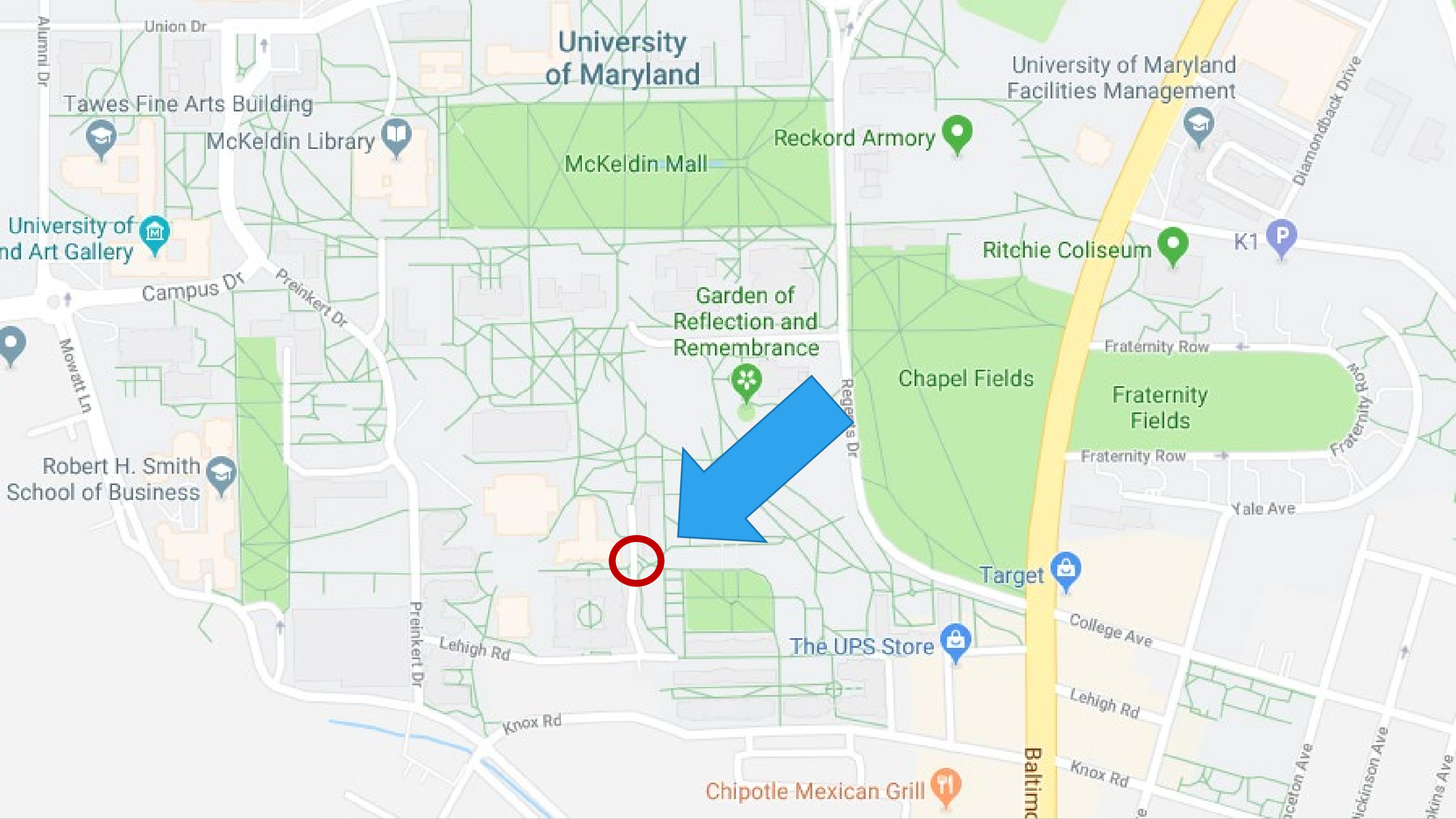
LIDAR SENSOR DESIGN



THE TARGET

THE TARGET

- Black fence with “M” cut-out
- Surrounded by brick pillars
- Back-side is impossible to map (with existing tripod)


University of Maryland
University of Maryland Facilities Management
Tawes Fine Arts Building
McKeldin Library
McKeldin Mall
Reckord Armory
Diamondback Drive
University of nd Art Gallery
Ritchie Coliseum
K1
Union Dr
Alumni Dr
Campus Dr
Preinkert Dr
Garden of Reflection and Remembrance
Regents Dr
Chapel Fields
Fraternity Row
Fraternity Fields
Mowatt Ln
Robert H. Smith School of Business
Yale Ave
Target
College Ave
Lehigh Rd
The UPS Store
Knox Rd
Chipotle Mexican Grill
Baltimo

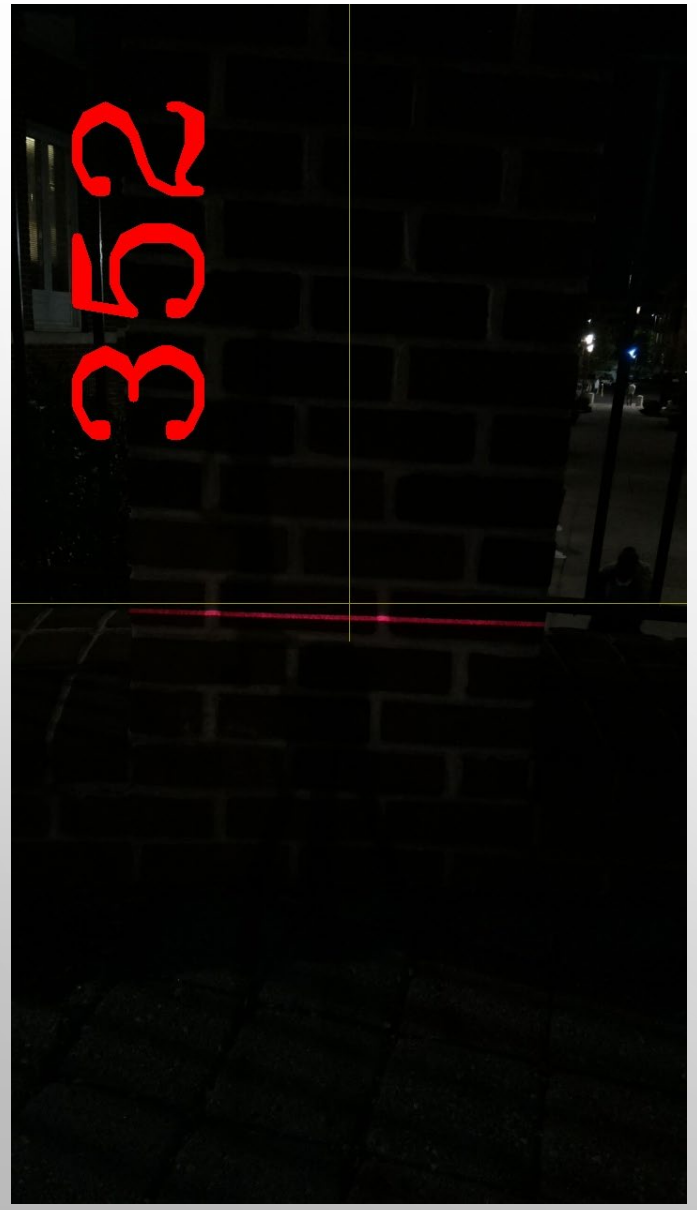
RANGE DATA COLLECTION

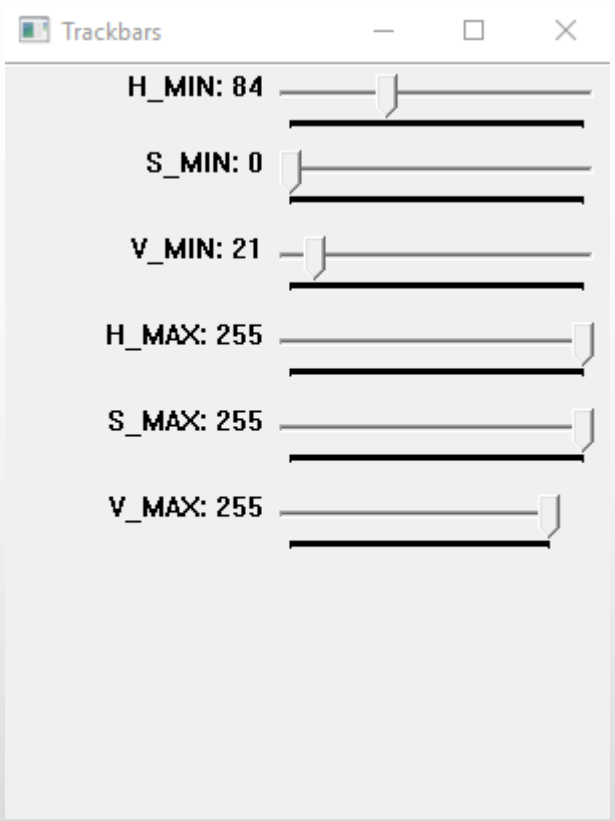
```
22  command = 'raspistill -w 1280 -h 720 -o blank.jpg'
23  os.system(command)
24
25  image = cv2.imread('blank.jpg')
26  image = cv2.flip(image,-1)
```

- Read in image from Pi Camera

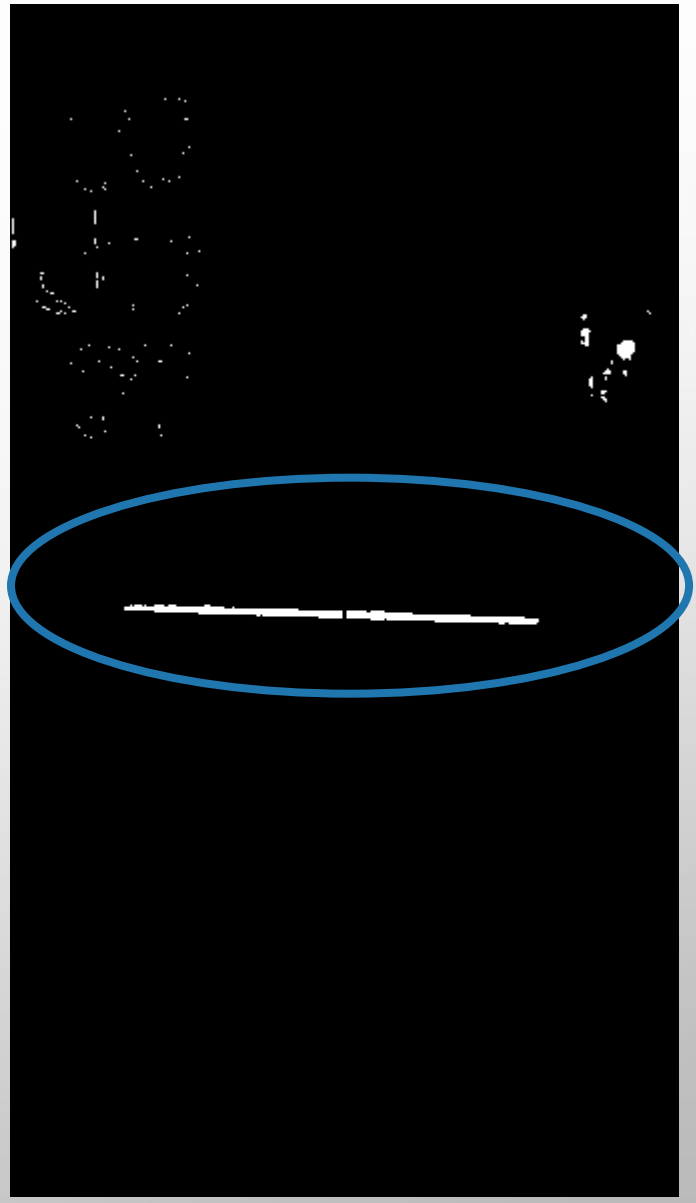
```
39  d = int(d)
40  filename = "%d.jpg" %d
41  cv2.imwrite(filename, image)
```

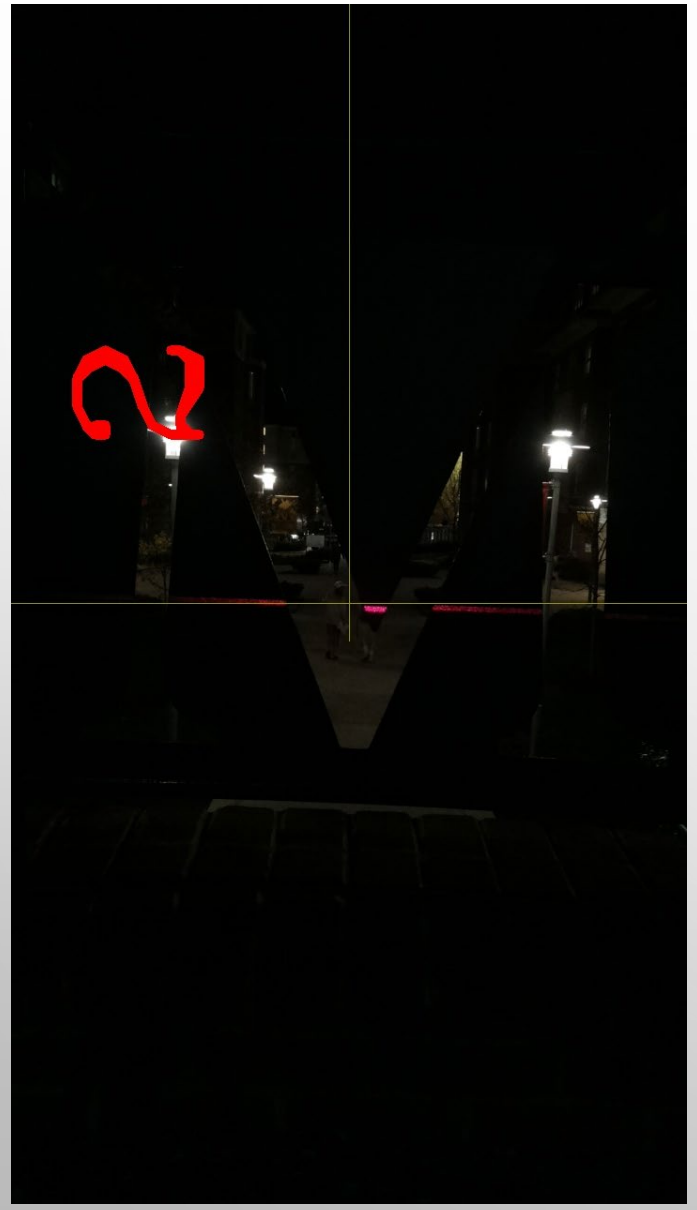
- Save image with the current IMU angle as the filename

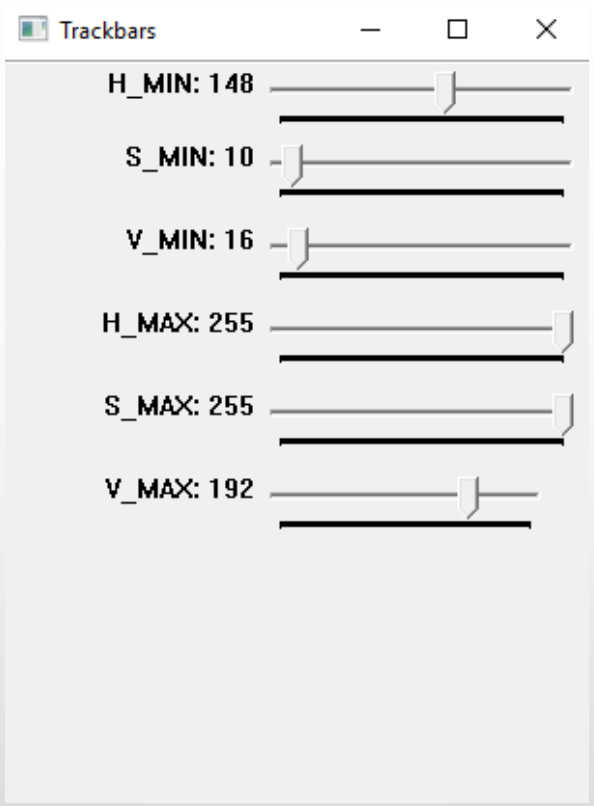


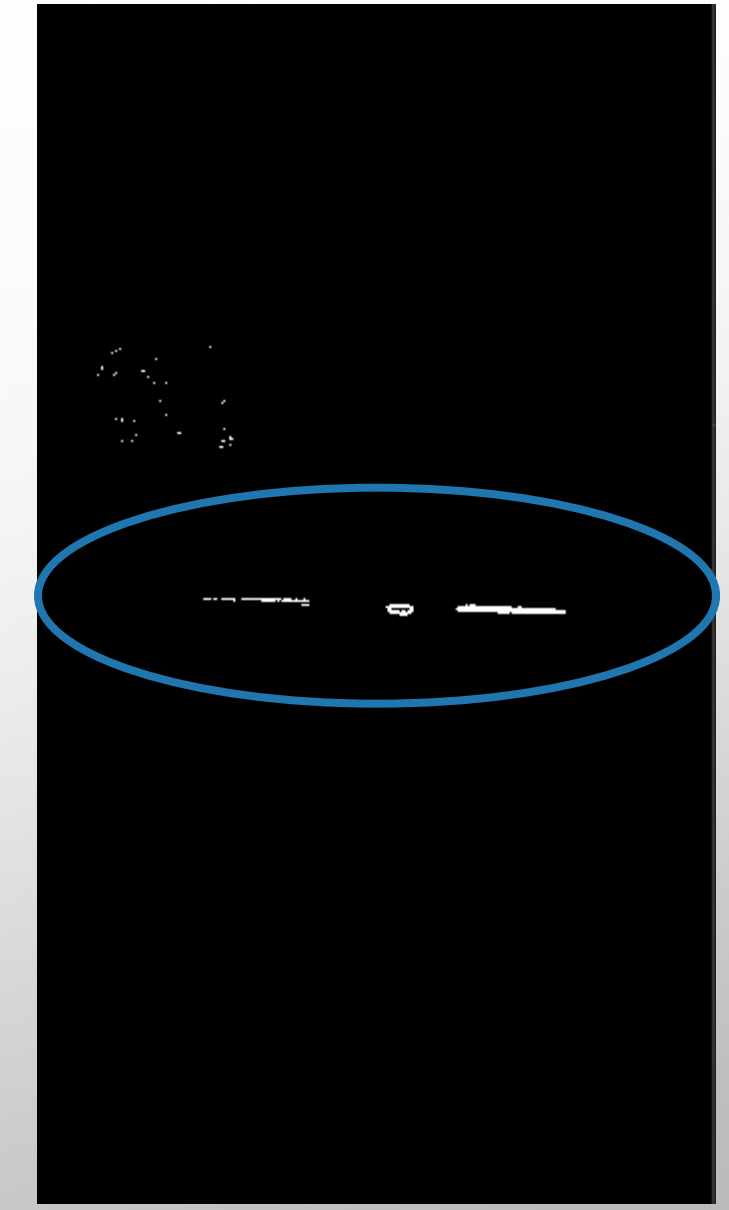


HSV Filtering









HSV Filtering

RANGE DATA PROCESSING

```
22  # M fence
23  colorLower = (148, 9, 15)
24  colorUpper = (255, 255, 190)
```

```
30  # M wall 1
31  colorLower = (84, 0, 21)
32  colorUpper = (255, 255, 255)
```

- Declare the lower and upper HSV thresholds for the image

```
55  # LiDAR parameters
56  ro = -0.51
57  rpc = 0.00064
58
59  # 12 inch separation distance, laser to camera, in meters
60  H = (7.5 / 12) * 0.3048
```

- Declare the ro and rpc LiDAR parameters
- Declare the laser to camera separation distance

RANGE DATA PROCESSING

```
135    mask = cv2.inRange(hsv, colorLower, colorUpper)
136    for x in range(0, 720, 1):
137        for y in range(0, len(mask)):
138            if mask[x][y] > 200:
139                spot.append(y)
```

- Process the image using the preset HSV threshold
- Iterate across vertical lines on the image, storing the location of each “bright” pixel

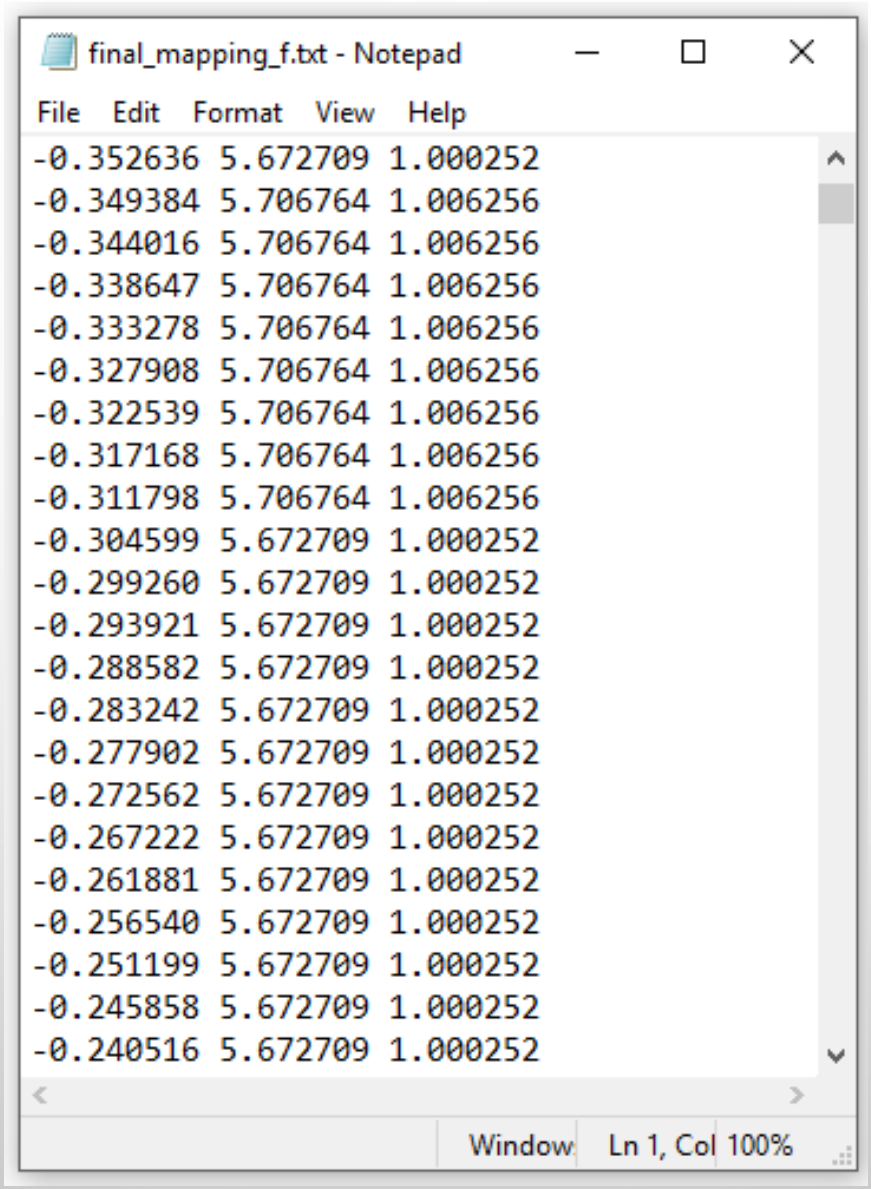
```
144    # Average all row pixels identified as the red laser line
145    if len(spot) > 1:
146        spot_int = int(np.average(spot))
```

- Take the average if multiple “bright” pixels are found

RANGE DATA PROCESSING

```
158   if spot_int >= 1:
159
160       # Ensure proper setup of pixels-from-center
161       spot = 640 - spot_int
162
163       # calculate range based on geometry
164       D = 3.28084*H/np.tan(spot_int * rpc + ro)
```

- Calculate the distance from LiDAR using the preset parameters


final_mapping_f.txt - Notepad
File Edit Format View Help
-0.352636 5.672709 1.000252
-0.349384 5.706764 1.006256
-0.344016 5.706764 1.006256
-0.338647 5.706764 1.006256
-0.333278 5.706764 1.006256
-0.327908 5.706764 1.006256
-0.322539 5.706764 1.006256
-0.317168 5.706764 1.006256
-0.311798 5.706764 1.006256
-0.304599 5.672709 1.000252
-0.299260 5.672709 1.000252
-0.293921 5.672709 1.000252
-0.288582 5.672709 1.000252
-0.283242 5.672709 1.000252
-0.277902 5.672709 1.000252
-0.272562 5.672709 1.000252
-0.267222 5.672709 1.000252
-0.261881 5.672709 1.000252
-0.256540 5.672709 1.000252
-0.251199 5.672709 1.000252
-0.245858 5.672709 1.000252
-0.240516 5.672709 1.000252
Window
Ln 1, Col
100%

3D POINT CLOUD GENERATION

```
33 -     xa = -1*A(:,2);
34 -     ya = -1*A(:,4);
35 -     za = -1*A(:,6);
```

- Read in the x, y, z data from the text file

```
53       % Noise filtering
54 -     yam = mode(ya);
55 -     for i = 1:size(ya)
56 -       if (ya(i) > yam + var || ya(i) < yam - var)
57 -           ya(i) = NaN;
58 -       end
59 -     end
```

- Filter out noise using a mode-variance model

3D POINT CLOUD GENERATION

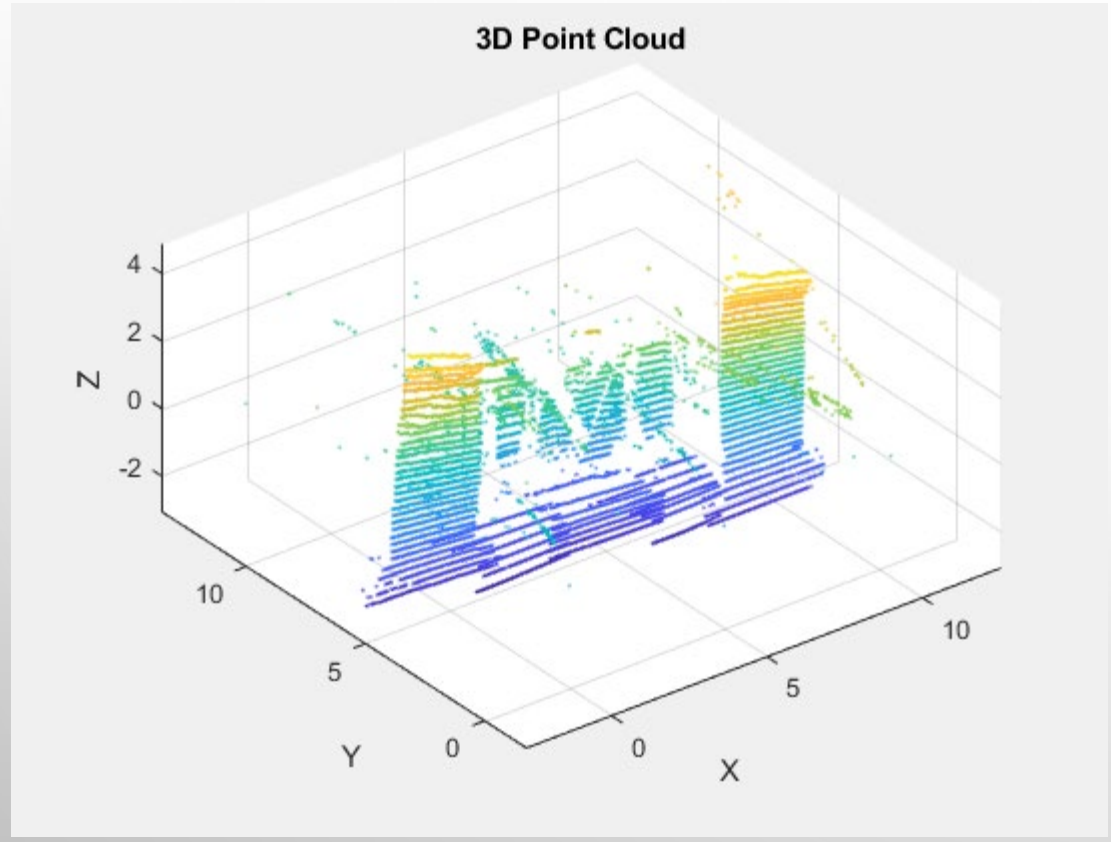
```
92 -    x = [xa; (xb+5); (xc+9); (xd+2.5); (xe+7.0)];
93 -    y = [ya; yb; (yc-1.8); (yd+1.2); (ye+1.1)];
94 -    z = [za; zb; (zc+1.5); (zd+0.7); (ze+0.7)];
```

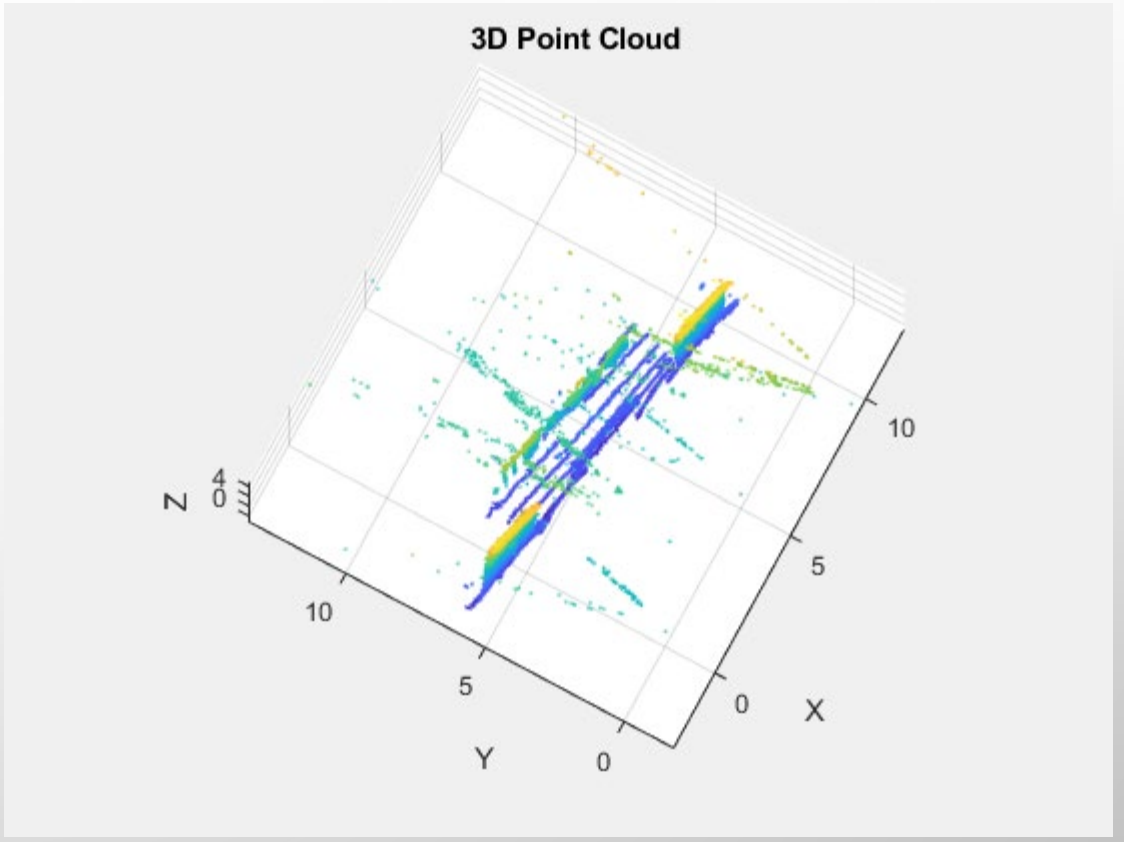
- Adjust the x, y, z position of each data set for the point cloud

```
104     % Plot point cloud using pcshow()
105 -   figure(1)
106 -   pcshow([x ,y, z]);
107 -   title('3D Point Cloud');
108 -   xlabel('X');
109 -   ylabel('Y');
110 -   zlabel('Z');
111 -   axis equal
```

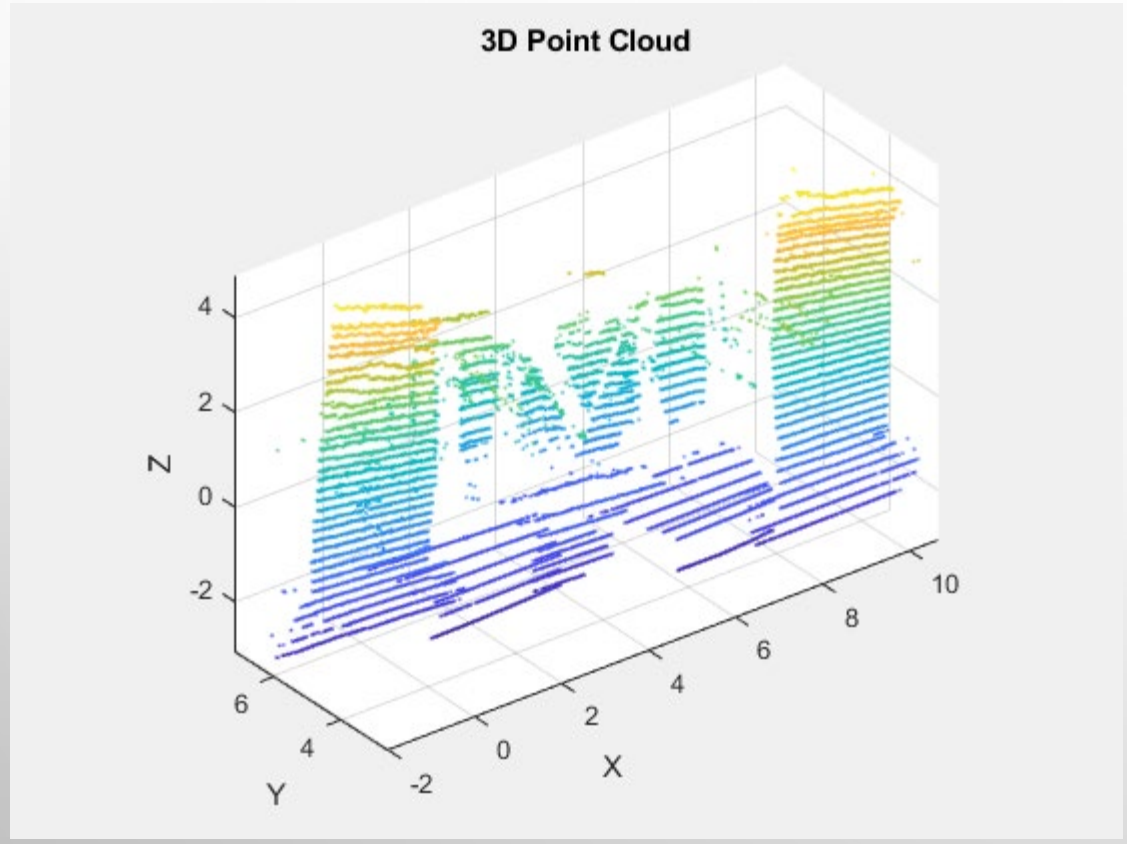
- Combine the x, y, z coordinates into a single plot
- Compute using MATLAB

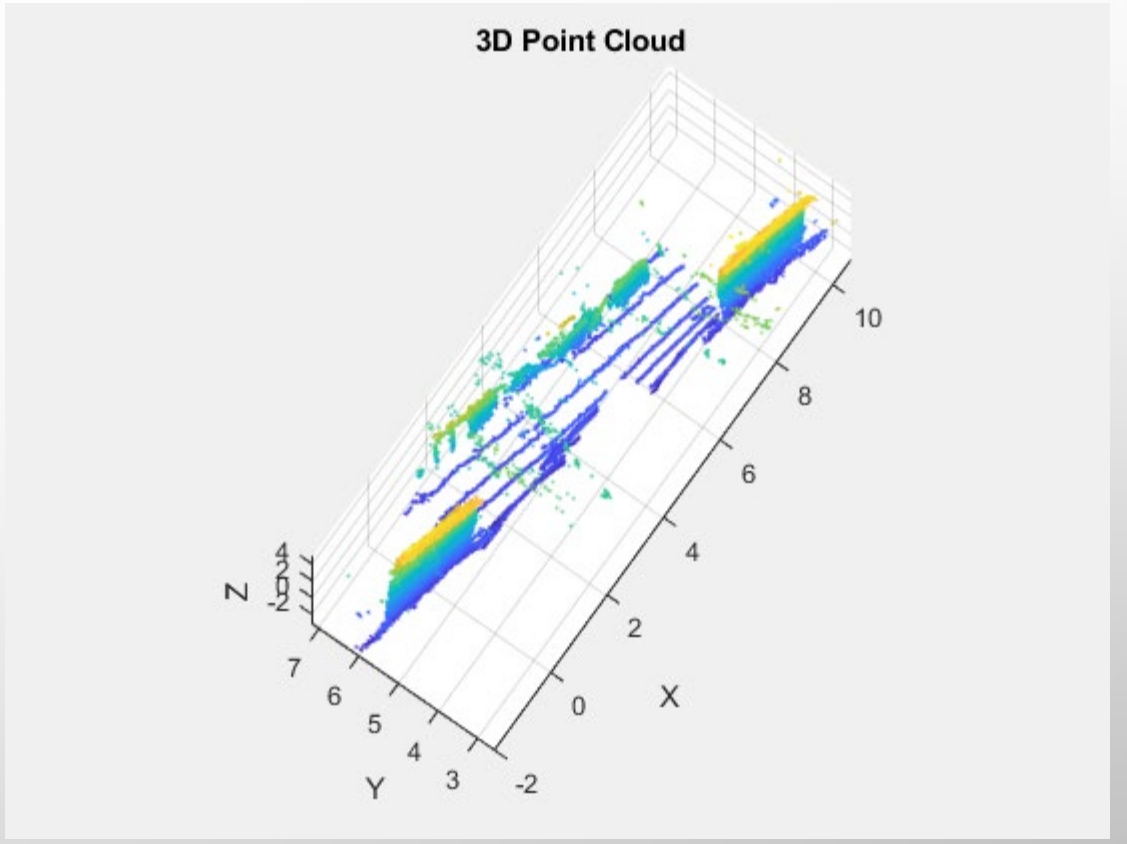
BEFORE FILTERING





AFTER FILTERING





3D POINT CLOUD GENERATION