4.1 Calibration with GOPR1 …GOPR11

Calibration parameters after initialization:

```
Focal Length:         fc = [ 572.29794   572.29794 ]

Principal point:      cc = [ 639.50000   479.50000 ]

Skew:            alpha_c = [ 0.00000 ]   => angle of pixel = 90.00000 degrees

Distortion:           kc = [ 0.00000   0.00000   0.00000   0.00000   0.00000 ]


Main calibration optimization procedure - Number of images: 11

Gradient descent iterations:
1...2...3...4...5...6...7...8...9...10...11...12...13...14...15...16...17...18...19...20...21...22...23...24...25...do
ne

Estimation of uncertainties...done


Calibration results after optimization (with uncertainties):


Focal Length:         fc = [ 558.54978   559.14975 ] +/- [ 7.83296   8.00240 ]

Principal point:      cc = [ 646.44130   503.79943 ] +/- [ 8.07018   6.95204 ]

Skew:            alpha_c = [ 0.00000 ] +/- [ 0.00000  ]   => angle of pixel axes = 90.00000 +/- 0.00000
degrees

Distortion:           kc = [ -0.25605   0.03413   0.00271   0.00048  0.00000 ] +/- [ 0.01046   0.00758
0.00221   0.00132  0.00000 ]

Pixel error:         err = [ 1.31476   0.96458 ]
```

Self calibrated K Matrix = [558.55   0   646.44

0   559.15   503.80

0   0   1   ]

Go Pro 3 K Matrix = [719.30   0   334.63

0   718.40   256.17

0   0   1   ]

Using the images of GOPR1…GOPR11 provided, the focal length of the camera is 558.55 which indicates that it has a lower magnification value (The camera is less zoomed in) compared to that of the go pro 3 K matrix (~720).

Computing Vanishing Point

Using matlab houghlines function, I was able to compute the intersection of lines with the following function.

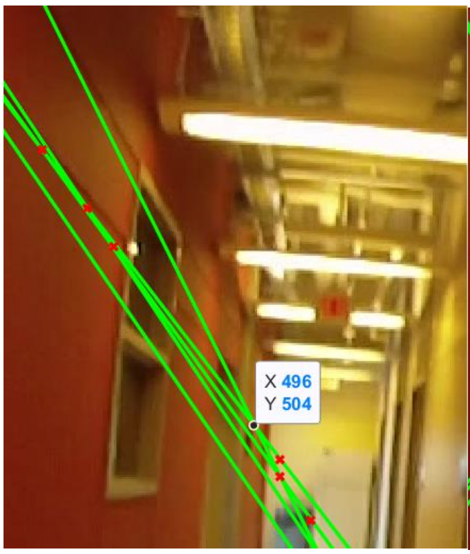
```
function point = linelineintersect(lines)

%https://en.wikipedia.org/wiki/Line%E2%80%93line_intersection

  x = lines(:,1);

  y = lines(:,2);

  denominator = (x(1)-x(2))*(y(3)-y(4))-(y(1)-y(2))*(x(3)-x(4));

  point = [((x(1)*y(2)-y(1)*x(2))*(x(3)-x(4))-(x(1)-x(2))*(x(3)*y(4)-y(3)*x(4)))/denominator,((x(1)*y(2)-y(1)*x(2))*(y(3)-y(4))-(y(1)-y(2))*(x(3)*y(4)-y(3)*x(4)))/denominator];

end
```

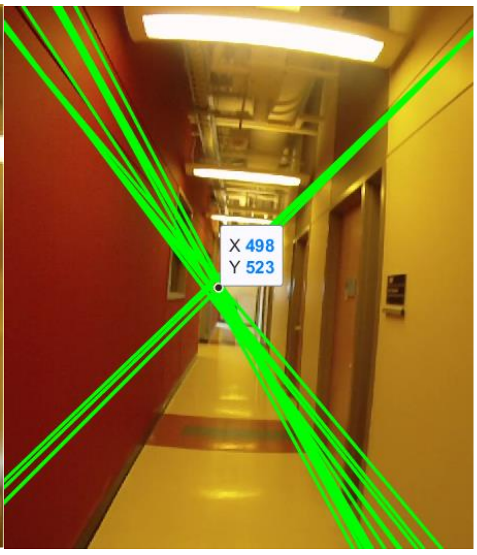
Reading and getting the points of the lines with matlab hough,houghpeaks,houghlines.

```
corridor1 = imread('Corridor1.jpg');

corridor2 = imread('Corridor2.jpg');

g1 = rgb2gray(corridor1);

g2 = rgb2gray(corridor2);

canny1 = edge(g1,'canny');

canny2 = edge(g2,'canny');

%imshowpair(canny1,canny2,'montage')

[H1,theta1,rho1] = hough(canny1);

[H2,theta2,rho2] = hough(canny2);

P1 = houghpeaks(H1,15,'threshold',ceil(0.2*max(H1(:))));

P2 = houghpeaks(H2,15,'threshold',ceil(0.2*max(H2(:))));


lines1 = houghlines(canny1,theta1,rho1,P1,'FillGap',5,'MinLength',7);

lines2 = houghlines(canny2,theta2,rho2,P2,'FillGap',5,'MinLength',7);
```

After plotting the lines and getting the intersection points, I use a voting scheme in which I detect the number of lines passing through the same point. I then use a thresholding to determine the point that has the most lines passing through. I have to tweak the houghpeaks and the thresholding a lot to have the cleaner picture to showcase here. The reason I choose to compute vanishing point this way is because there are too many lines presented in the image which despite many efforts from me I was not able to retain only the parallel lines. The downside of this approach is that since there are too many 'noises' as indicated by the lines, you can see that there are regions where there are multiple points that fulfil the same condition. The problem could lie in the radial distortion of the GoPro lens. The matlab code can be found below.

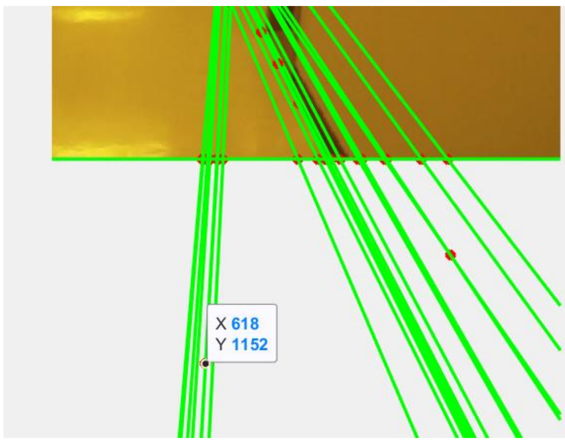
Centre Vanishing point for image 1 (left) and central vanishing point for image 2 (right)

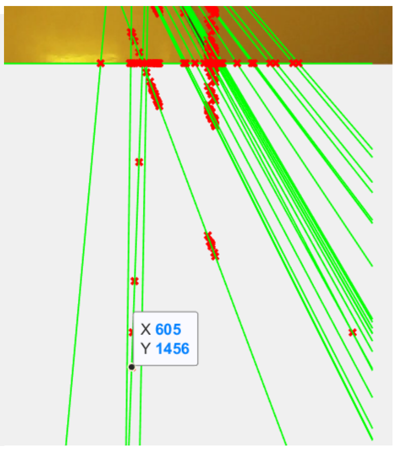




I plot a line from x= 1 to x = 999 to

Vertical Vanishing point for image1 (left) and vertical vanishing point for image 2 (right)





Central vanishing point = (496, 504) (498,523)

Vertical Direction = (618,1152) and (605,1456)

$$\theta = \arcsin(\frac{x-u}{f_x})$$
$$\phi = \arcsin(\frac{y-v}{-f_y\cos(\theta)}).$$

```
heading1 = asind((496 - 664.903569991381570)/585.850107917267790);

pitch1 = asind((504 - 498.409524449186850)/-586.003198722303180*cosd(heading1));

heading2 = asind((498 - 664.903569991381570)/585.850107917267790);

pitch2 = asind((523 - 498.409524449186850)/-586.003198722303180*cosd(heading2));
```

I have computed the rotation with the formula above using my central vanishing point[Source: Chapter 4 of Ruotsalainen, L. (2013). Vision-Aided Pedestrian Navigation for Challenging GNSS Environments. (Suomen geodeettisen laitoksen julkaisuja - Publications of the Finnish Geodetic Institute;151). Suomen geodeettinen laitos. The priori assumption is that the camera's skew is zero.

Results:

```
heading1 = -16.7565

heading2 =  -16.5523

pitch1 =  -0.5234

pitch2 = -2.3053
```

There is a trivial change in the heading angle while the pitch angle has rotated a little bit.

MATLAB code

```
[rows, columns] = size(canny1);
imshow(corridor1)
hold on
pointsx = []
pointsy = []
votes = []
for i = 1:length(lines2)-1
  for j = i+1:length(lines2)
    xy1 = [lines2(i).point1; lines2(i).point2];
    xy2 = [lines2(j).point1; lines2(j).point2];
    x11 = xy1(1,1);
    y11 = xy1(1,2);
    x12 = xy1(2,1);
    y12 = xy1(2,2);
    x21 = xy2(1,1);
    y21 = xy2(1,2);
    x22 = xy2(2,1);
    y22 = xy2(2,2);
    m1 = (y12-y11)/(x12-x11);
    point = linelineintersect([x11 y11; x12 y12; x21 y21; x22 y22]);
    found = false;
    [s1, s2] = size(pointsx);
    for ix=1:s2
      if floor(pointsx(ix)) == floor(point(1)) && floor(pointsy(ix)) == floor(point(2))
        %Straight line passing the same point
        votes(ix) = votes(ix) + 1;
        found = true;
      end
    end
    if found == false
```

```
        pointsx(end+1) = point(1);
        pointsy(end+1) = point(2);
        votes(end+1) = 0;
      end
      if abs(floor(point(1))) > size(canny1,1) || abs(floor(point(2))) > size(canny1,2)
        disp(floor(point(1)));
        disp(floor(point(2)));
      end
      %Drawing the line
      xLeft = 1;
      yLeft = m1 * (xLeft - x11) + y11;
      xRight = 999;
      yRight = m1 * (xRight - x11) + y11;
      plot([xLeft, xRight], [yLeft, yRight], 'LineWidth',1,'Color','green');
      plot(floor(point(1)), floor(point(2)), 'x','LineWidth',2,'Color','red');
    end
  end
[s1, s2] = size(votes);
for i=1:s2
  if votes(i) > 0
        plot(floor(pointsx(i)), floor(pointsy(i)), 'x','LineWidth',2,'Color','red');
  end
end
```