In this project, the goal was to undergo the process of image filtering and Hough transform. The operations undergone to throughout this project were convolving an image with a given convolution filter, edge detection, which takes as input a greyscale image and sigma and then output the edge magnitude image. Then we will need to apply Hough transform to this edge magnitude image to visualize the edges in the Hough parameter space, and then finally finding the lines and fitting line segments for visualization.

**Convolution**

The original image is in jpeg format, so we must convert it to an array format in order to do operations on it. In order to convolve the image, we first need to handle boundary cases on the edges of the image in order to preserve the input images original dimensions. In order to do this, we must pad the image by outputting pixels lying outside the image boundary that have the same intensity values as the nearest pixel that lies inside the image, this is called replication padding. The pad function is used as helper function inside the myImageFilter() function implemented in myImageFilter.py. Because we are undergoing convolution, the filter has be rotated by 180 degrees in order to filter the image, not doing so will just make it undergo correlation instead. Below is an example of how an image array gets replication padded.

Calendar

Description automatically generated with medium confidence

After this is done, we must convolve this array with a filter, in this example, we convolve it with a Prewitt filter. The resulting filtered image is the second output below.

Table

Description automatically generated

Below is the image we are going be using:

**A picture containing shoji, indoor, dark, light

Description automatically generated**

**Edge detection**

Now we must detect edges in the image and output the edge magnitude image. This can be done by taking the image gradients in the x direction and y direction. In order to find the image gradients in the x and y direction, we must filter replicated filtered image by the x-oriented Sobel filter and y-oriented Sobel filter respectively. Examples of the resultant image gradients in the x and y are below.

|  |  |
| --- | --- |
| A picture containing diagram  Description automatically generated | A picture containing diagram  Description automatically generated |
| **Image gradient in the x direction** | **Image gradient in the y direction** |

We also must also smooth out the image with a Gaussian kernel that has a sigma, and is used before edge detection. The implementation of the Gaussian kernel is inspired by Ali\_m’s python implementation of it found on Stack Overflow. [[1]](#footnote-1) After applying the Gaussian smoothing and finding the x gradient and y gradient images, in order to find the edge magnitude images, we must find the magnitude of the x and y gradients, this is done by taking the whole square root of the x gradient squared plus the y gradient squared, and then each pixel of that result must be multiplied by 255 divided the max of the result. The resultant edge magnitude image is below.

A picture containing calendar

Description automatically generated

**The Hough transform**

Next, we had to apply Hough transform to an edge magnitude image. The myHoughTransform.py functions were inspired by Alyssa Queks GitHub repo[[2]](#footnote-2) with a slight alteration changing the thetas from (-, to (0, 2). A visualization of the edge magnitudes image in the Hough parameter space is displayed below.

A picture containing text

Description automatically generated

**Trying out my own image**

I tested the functions on an image of my own liking. It is a birds-eye view of the track colorings of the French Grand Prix for Formula 1. The original input image is below:

A picture containing striped

Description automatically generated

After applying edge detection, the image looks like this, along with the Hough transform

|  |  |
| --- | --- |
| Diagram  Description automatically generated with medium confidence  As you can see, the Hough transform contrasts with the one of the first image in the project. | A picture containing text  Description automatically generated |

**Theory Questions**

**2.** We parametrize the line in terms of (rho, theta) instead of the slope and intercept (m,c) due to the slope and intercept form not being able to account for when x is null and y is any not null, or x is null and y is null which then makes the slope and intercept form undefined, indeterminate.

Expressing the slope and intercept form in terms of and :

\_(1)

\_(2)

Rearranging (1) to make y the subject:

Equating this with (2) we get:

Hence our slope and intercept form expressed in terms of and :

**3.** The maximum absolute value of and the range for are as below:

1. https://stackoverflow.com/questions/17190649/how-to-obtain-a-gaussian-filter-in-python [↑](#footnote-ref-1)
2. https://github.com/alyssaq/hough\_transform/blob/master/hough\_transform.py [↑](#footnote-ref-2)