1 Theory Questions

1.1 Q2.1

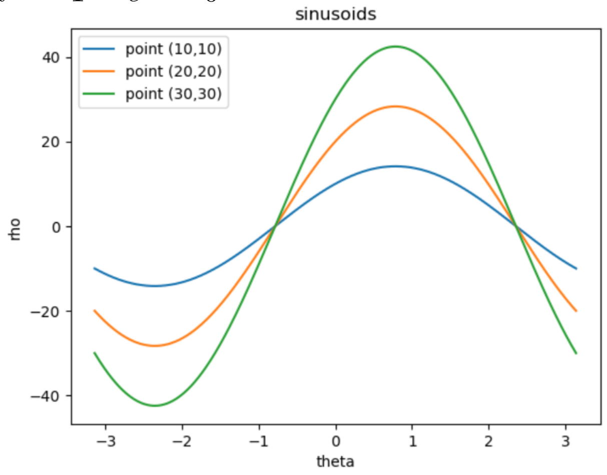
- Show that if you use the line equation $p = x\cos\theta + y\sin\theta$, each image point (x,y) results in a sinusoid in (p, θ) Hough space. Relate the amplitude and phase of the sinusoid to the point (x, y).
  - p = distance from the origin to line
    θ = angle made by the perpendicular line and the x-axis
    x,y = point
  - When in the image space, different values of θ generate different values of p.
  - When these (p, θ) points are plotted they show a sinusoidal curve
  - The amplitude of the sinusoid relates to the distance of the point from the origin. The phase changes depending on the angle of the line connecting with the point to the origin, line detection works with every point in the image space pairing with a sinusoid in the Hough space.
- Why do we parametrize the line in terms (p, θ) instead of the slope and intercept (m, c)? Express the slope and intercept in terms of (p, θ).
  - This happens because if we parameterized the line in terms of the slope and intercept, the accumulator for line fitting would become too large, and it does not handle vertical line.
  - The slope in term of (p, θ):

$$y = \left(-\frac{\cos(\theta)}{\sin(\theta)}\right) x + \frac{p}{\sin(\theta)}$$

$$m = -\frac{\cos(\theta)}{\sin(\theta)} \qquad c = \frac{p}{\sin(\theta)}$$

- Assuming that the image points (x,y) are in an image of width W and height H, that is, $x \in [1,W]$, $y \in [1,H]$, what is the maximum absolute value of p, and what is the range for θ?
  - p maximum $= \sqrt{W^2 + H^2}$
  - θ range $= 0^\circ$ to 180°
- For point (10,10) and points (20,20) and (30,30) in the image, plot the cor- responding sinusoid waves in Hough space, and visualize how their intersection point defines the line. What is (m, c) for this line? Please use Python to plot the curves and report the result in your write-up.

- $y = mx + c$
  $p = x \cos\theta + y \sin\theta$
- $m = 1 \quad c = 0$



2 Implementation

2.1 Q3.1

Convolution

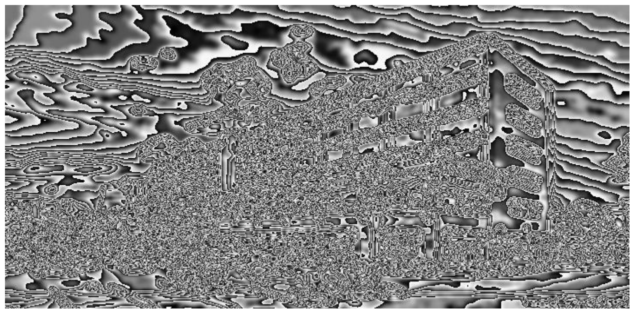


Figure 1: Image after image filtering

2.2 Q3.2

Edge Detection

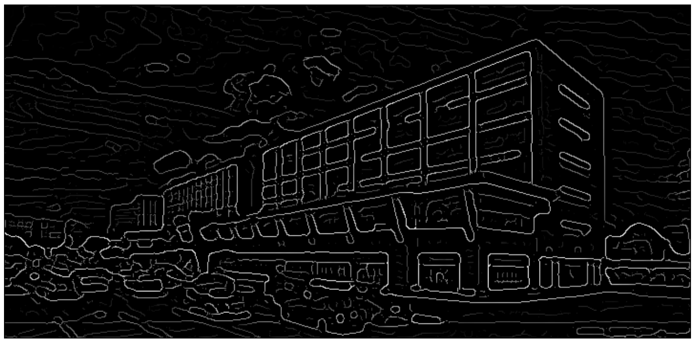


Figure 2: Image after edge filtering

2.3 Q3.3

Hough Transform

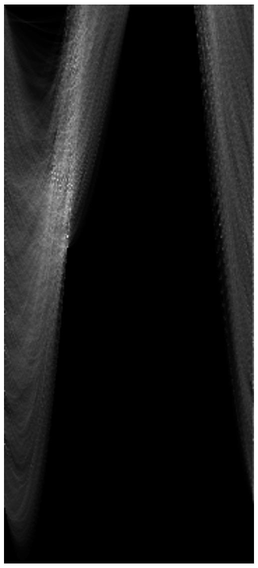


Figure 3: Hough result

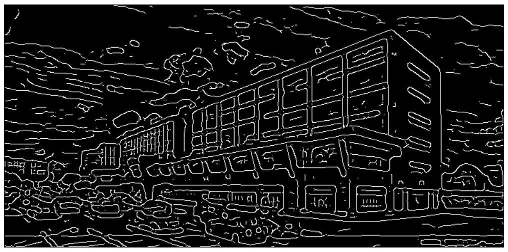


Figure 4: Threshold result

2.4 Q3.4

Finding Lines

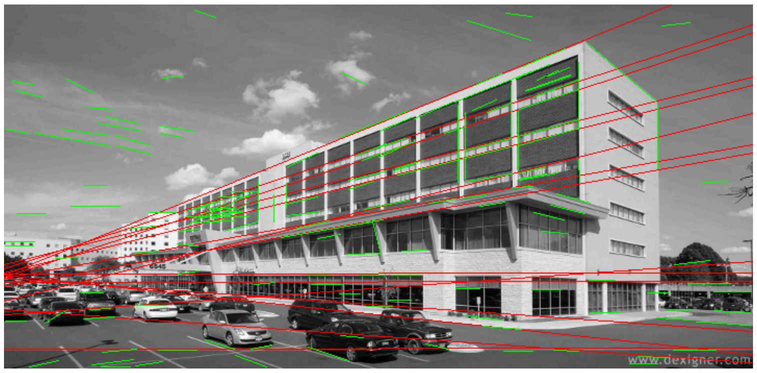


Figure 5: Hough lines result

3 Write Up

3.1 Q4.1

- Did your code work well on all the image with a single set of parameters?
  - It seemed as though different images require different parameters. For example, items with more edges/lines in the original image required a higher threshold to get a good looking result, compared to images with minimal lines, because a higher threshold meant the code was more selective when drawing out lines. I especially had a difficult time with the images that meet at a point, like the hallway and the road. Increasing sigma resulted in a more smoothed image. The nLines parameter affected how many lines were ultimately drawn out/detected. The thetaRes and rhoRes affected the resolution of the resolution of the Hough accummulator angularly and in the rho dimension respectively.

- How did the optimal set of parameters vary with images? Which step of the algorithm causes the most problems?
  - The threshold seemed to be the most important value. It drastically changes the results you get. For my results I set the the threshold to 0.975 and my sigma value to 3. Both the hough transform and the edge filter sections caused the most problems.
- Did you find any changes you could make to your code or algorithm that improved performance?
  - I initially created my Hough Transform function using a small nested for loop. This seemed to not quite do the trick and it also took a longer time to work, so i instead switched to the method i ended up submitted, which didn't use for loops, and the performance significantly improved.