16385

Yunchu Chen

September 2023

Q2.1

Show that if you use the line equation $\rho = x\cos\theta + y\sin\theta$, each image point (x, y) results in a sinusoid in (ρ, θ) Hough space. Relate the amplitude and phase of the sinusoid to the point (x, y).

For transformation $\rho = x\cos\theta + y\sin\theta$, fix arbitrary image point(m, n), S.T. $m, n \in Z$
Then $\rho = m\cos\theta + n\sin\theta$

$$
\begin{aligned}
&\rho = m\cos\theta + n\sin\theta \\
\Rightarrow &\rho = \sqrt{m^2+n^2}\left(\frac{m}{\sqrt{m^2+n^2}}\cos\theta + \frac{n}{\sqrt{m^2+n^2}}\sin\theta\right) \\
\Rightarrow &\rho = \sqrt{m^2+n^2}(\sin\alpha\cos\theta + \cos\alpha\sin\theta) \\
\Rightarrow &\rho = \sqrt{m^2+n^2}\sin(\alpha+\theta) \qquad \text{(By the rules of trigonometry)}
\end{aligned}
$$

Since we have transformed the equation to the standard form, we can conclude that the amplitude of the resulting sinusoid is $\sqrt{m^2+n^2}$, while the phase of the resulting sinusoid is $\alpha = \arctan\frac{m}{n}$

Q2.2

Why do we parameterize the line in terms of ρ instead of slope and intercept (m, c)? Express the slope and intercept in terms of ρ and θ.

This is because both m and c could take extreme values $-\infty \leq m, c \leq \infty$, the resulting intersections might be far away, so it would be difficult to define the size of the accumulator. However, if we convert the line in terms of ρ, we could effectively limit the size of the accumulator, which makes computation easier.

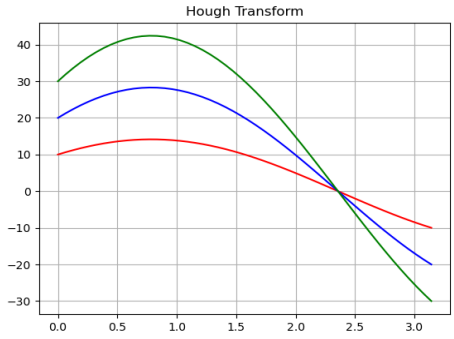
$$\rho = x\cos\theta + y\sin\theta \qquad \text{(Want to convert it in the format } y = mx + c\text{)}$$
$$\Rightarrow y\sin\theta = \rho - x\cos\theta$$
$$\Rightarrow y = \frac{\rho}{\sin\theta} - \frac{\cos\theta}{\sin\theta}x$$
$$\Rightarrow m = -\frac{\cos\theta}{\sin\theta}, c = \frac{\rho}{\sin\theta}$$

Q2.3

According to the formula we derived in Q2.1, the resulting sinusoid is $\rho = \sqrt{x^2 + y^2}\sin(\alpha + \theta)$.
Note that the maximum value for $\sin(\alpha + \theta)$ is 1, and in order to to maximize x and y, x should be the width of the image and y should be the height of the image. So the maximum abosolute value for ρ is $\sqrt{W^2 + H^2}$and the range for θ is $[0, 2\pi]$, as the coefficient before θ is 1 in the sin wave.

Q2.4

For point (10,10) and points (20,20) and (30,30) in the image, plot the corresponding 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.



Solving the intersection, we get

$$\theta \approx 2.3558, \rho \approx 5.4973$$

Plug in the function we derived 2.2, we get

$$m = 1, c = 0$$

Q4.1 Discussion

Use the script included to run your Hough detector on the image set and generate intermediate output images. Include the set of intermediate outputs for one image in your write-up; this means include at least one image from myImageFilter, one image from myEdgeFilter, and one image from myHoughTransform.

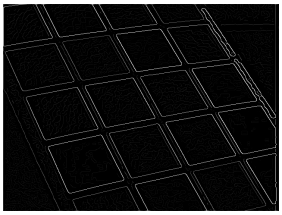


Figure 1: result from myEdgeFilter

Figure 2: result from myHoughTransform

Figure 3: img08

Figure 4: img08 gaussian

Output Demonstration

I used the initial hyperparameters to generate myEdgeFilter and myHoughTransform results for img01. It is evident that myEdgeFilter effectively captures the edges in the grid figure, and the outcome of the Hough transformation aligns with the one provided in the write-up.

Additionally, I included a comparison of original image 08 and the result of applying myImageFilter with a 3x3 Gaussian filter to it. It is noticeable that

the result has a blurring effect on the original figure.
As we can see in the example, increasing threshold would reduce the noise and thus has a better performance.

Hyper-parameter Discussion:

Threshold: In Hough Transformation, pixel lower than the threshold will be ignored, to increasing the threshold will increase the number of pixels with low values to be ignored. In this way, the program will capture less noises in the images. In the example, img07 with a threshold of 0.03 performs not that well for capturing too much noises between woods, but performs better after increase the threshold to 0.07.

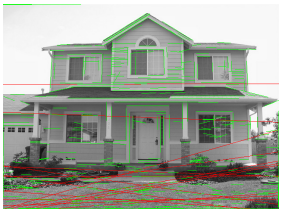


Figure 5: img07 with threshold of 0.03

Figure 6: img07 with threshold of 0.07

Sigma: σ represents the standard deviation of the gaussian distribution, which visually corresponded to the width of the gaussian distribution. If σ gets higher, there would be a wider Gaussian curve, which means the resulting convolution filter would consider more neighbors in calculating the results, thus creating a stronger blurring effect in the results. In this way, the outputs will perform better in capturing less noises in the original graph. Note that noises are significant reduced if increase sigma to 4.

Figure 7: img08 with sigma=2

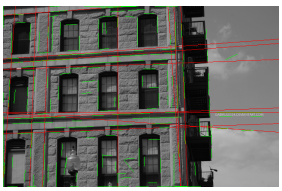


Figure 8: img08 with sigma=4

ThetaRes and RhoRes: Higher resolution allows with subtle differences in terms of degrees to be discovered easier, which means edges that are not obvious would be detected eaiser. However, there might be more noises get detected if resolution grows too high.

nLines: Similar to resolution, increasing nLines would allow more edges to be

detected by the program. However, if nLines grows too large, more noise would be detected.

However, note that there is no universal set of parameters that works well among all of the images, this is because different images require different method of edge detection. For example, for images with lots of edges that are closely connected, we have to increase nLines, resolution, but lowering threshold and sigma in general. Nevertheless, for images such as a road with only a few edges, we need to do the opposite so that the program does not detect too much noise.

Conclusion:

Problem:

The most significant concern regarding time complexity arises in Section 2, where a double loop is used to compare each pixel with its neighbors.

Section3 causes the most problem in terms of debugging as in Cartesian coordinate system, x starts from the bottom left corner while in an np array, the index starts at upper left ([0,0]).

Improvement: Yes, previously, I did not properly vectorized my code in myImageFilter because of using loops to sum up the values. However, after vectorizing my code using numpy function np.sum, my code improves significantly in terms of running speed.