Assignment 1

Kanta Saito
ICS 483 - Computer Vision

Q2.1 Hough Transform Line Parameterization (20 points)

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).

Answer: The Hough parameterization for a line is given by the equation:

$$\rho = x\cos\theta + y\sin\theta$$

To show this curve is a sinusoid, we can use the harmonic addition theorem to transform the right-hand side of the equation into a single cosine term of the form $A\cos(\theta - \phi)$.

The cosine difference identity is:

$$A\cos(\theta - \phi) = A(\cos\theta\cos\phi + \sin\theta\sin\phi) = (A\cos\phi)\cos\theta + (A\sin\phi)\sin\theta$$

By comparing the coefficients of $\cos\theta$ and $\sin\theta$ between this expanded form and our original expression, we can establish the following relationships:

$$x = A\cos\phi$$
$$y = A\sin\phi$$

From these two equations, we can solve for the amplitude A and the phase shift ϕ in terms of the image point coordinates (x, y).

Finding the Amplitude (A)

To find the amplitude, we square and add the two equations:

$$x^2 + y^2 = (A\cos\phi)^2 + (A\sin\phi)^2$$
$$x^2 + y^2 = A^2\cos^2\phi + A^2\sin^2\phi$$
$$x^2 + y^2 = A^2(\cos^2\phi + \sin^2\phi)$$

Using the trigonometric identity $\cos^2\phi + \sin^2\phi = 1$, we get:

$$x^2 + y^2 = A^2$$
$$\implies A = \sqrt{x^2 + y^2}$$

Finding the Phase (ϕ)
To find the phase, we divide the second equation by the first:

$$\frac{y}{x} = \frac{A\sin\phi}{A\cos\phi} = \tan\phi$$

From this, we can find ϕ using an inverse tangent function. However, the standard $\arctan(y/x)$ function returns values in the range $(-\pi/2, \pi/2)$, which only covers quadrants I and IV. To ensure our phase angle ϕ is correct for all four quadrants, we use the 'atan2(y, x)' function, which takes both x and y as arguments and considers their signs to return an angle in the full range $(-\pi, \pi]$.

$$\phi = \text{atan2}(y, x)$$

By substituting A and ϕ back, we can rewrite the original Hough equation as:

$$\rho = A\cos(\theta - \phi)$$

This gives the final sinusoidal form for a point (x, y):

$$\rho = \sqrt{x^2 + y^2}\cos\left(\theta - \text{atan2}(y, x)\right)$$

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

Answer: The primary reason we prefer the polar parameterization (ρ, θ) over the Cartesian slope-intercept form (m, c) is that the (m, c) form cannot handle vertical lines. The parameter space for (m, c) is unbounded, which makes it unsuitable for the Hough transform.

To see why, we first express m and c in terms of ρ and θ. We start with the polar form:

$$\rho = x\cos\theta + y\sin\theta$$

Rearranging this equation into the form $y = mx + c$:

$$\begin{aligned} y\sin\theta &= \rho - x\cos\theta \\ y &= \left(-\frac{\cos\theta}{\sin\theta}\right)x + \frac{\rho}{\sin\theta} \\ y &= (-\cot\theta)x + (\rho\csc\theta) \end{aligned}$$

By comparing this to $y = mx + c$, we find the relationships:

$$m = -\cot\theta \quad \text{and} \quad c = \rho\csc\theta$$

The problem with this is vertical lines. A vertical line has an angle of $\theta = 0°$ (or $\theta = 180°$). At this angle, the slope $m = -\cot(0)$ approaches infinity. Since both the slope m and the y-intercept c would be infinite, you cannot represent a vertical line in the (m, c) parameter space.

The (ρ, θ) space, however, is bounded. ρ is limited by the image diagonal and θ is limited to a finite range.

3. 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 ρ, and what is the range for θ?

Answer: To find the maximum of ϕ and range θ, we would need to reference the Pythagorean theorem and the Hough accumulator.

Maximum Absolute Value of ρ: The parameter ρ represents the perpendicular distance from the origin to a line.

Assuming the image origin $(0, 0)$ is at one of the corners, the point furthest from the origin is the diagonally opposite corner, (W, H). The distance to this point is found using the Pythagorean theorem:

$$d = \sqrt{W^2 + H^2}$$

Therefore, the maximum absolute value for ρ is the length of the image diagonal.

$$\max(|\rho|) = \sqrt{W^2 + H^2}$$

Range for θ: A line in 2D space is uniquely defined by its normal angle θ and its distance ρ. A line described by (ρ, θ) is identical to the line described by $(-\rho, \theta + \pi)$. To avoid double counting lines in the Hough accumulator, the range of θ is restricted.

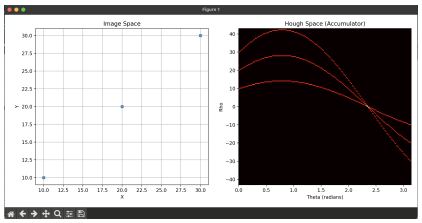
A common approach is to limit the angle to a 180-degree range:

$$\theta \in [0, \pi)$$

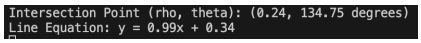
By restricting θ to this range, every possible line is unique. The positive or negative value of ρ then accounts for lines on either side of the origin for the given angle. This makes the representation both unique and effective in the parameter space.

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.

Answer:



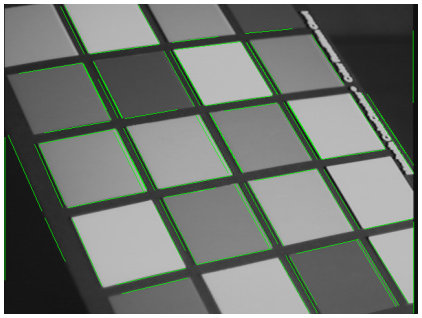
(a) Image Space and Hough Space Graphs



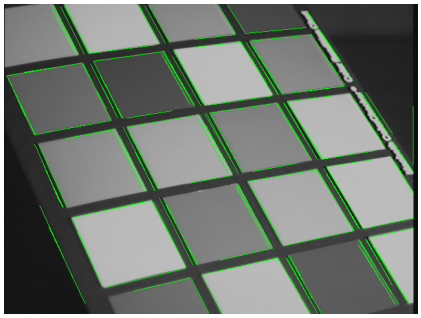
(b) Linear Equation y=mx+c

Results: When plotting the points (10, 10), (20, 20), and (30, 30), I found the slope using my equation $y = -\cot\theta x + \rho \csc\theta$ in Python. I also had to make sure that $\sin\theta$ is not 0, otherwise the result would be undefined. I found that the slope (m) is 0.99 and the y-intercept (c) is 0.34.

Q3.5 Implement HoughLines yourself (Extra Credit 10 points)



(a) CV2 Lines



(b) My Lines

Results: In my custom Hough Line Segmentation function, I hard-coded the maximum gap and minimum length parameters. I attempted to use the same length and gap values for both my function and the 'cv2.HoughLinesP' function. However, the outputs would not match, no matter how many times I adjusted these values. I concluded that this discrepancy was likely due to the 'threshold' parameter in the OpenCV function, which made it difficult to tune its results to match my implementation's output.

After further research and tweaking of my Hough lines implementation, I noticed that I was calculating my own 'thetaScale' and 'rhoScale' instead of using the ones generated by my Hough transform function. After correcting this, I observed a gradual increase in the number of detected lines. However,

the process of finding the lines was very slow because a large number of peaks were being found with the set threshold.

Another factor that helped increase the number of detected lines was the advice from Dr. George. He mentioned that the reason I might not have been getting good results was the size of my Hough transform accumulator. After fixing that, I saw a significant increase in the number of detected lines.

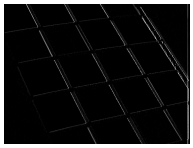
4.1 Write-up (15 points)

My Image Filter: When I used a blur kernel for my filter (h), it performed well on every image with a single set of parameters. However, when I used a Sobel filter, my resulting image appeared flipped compared to the output of cv2.filter2D(). Because of this, I replaced my np.roll code with a simple iteration loop for both the x and y axes.

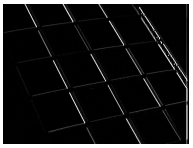
However, I noticed that my iteration code was performing correlation instead of convolution, so I flipped the kernel using np.flip(h) before iterating. Upon further research, I found that the OpenCV documentation states its function performs correlation, meaning the kernel is not flipped during its operations.

When comparing the images from my filter with those from the OpenCV filter, I noticed that the OpenCV image has much brighter lines than mine. This is likely due to differences in how our respective methods padded the image before convolution.

Lastly, I should have created a 5x5 Sobel kernel to compare against my 3x3 Sobel kernel. However, based on experience, I believe the larger kernel would have more difficulty with fine edges and might not show as much detail as the 3x3 kernel.



(a) My Image Filter 1 (Sobel)



(b) CV2 Image Filter 1 (Sobel)

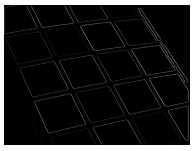
(a) My Image Filter 2 (Sobel)

(b) CV2 Image Filter 2 (Sobel)

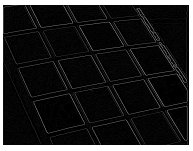
My Edge Filter For my edge filter parameters, I used the provided sigma value of 2. Initially, I tried using the equation ($hsize = 2 \times \lceil 3 \times \sigma \rceil + 1$) to determine the kernel size, but this did not produce the desired edges when compared to the example image.

I then experimented with different hsize parameters for the Gaussian filter and discovered that a size of $hsize = 2 \times \sigma$ yielded the most edges. A visual comparison of the output images clearly showed that my chosen hsize value produced more prominent edges than the result from the originally suggested formula.

I initially struggled with the implementation of non-maximum suppression, as I was unsure how to begin. I used an online resource, specifically Shrestha's blog, to guide my implementation (Shrestha, 2022).



(a) Edge Filter (Given hSize)



(b) Edge Filter (Own hSize)

(a) Edge Filter 2 (Given hSize)

(b) Edge Filter 2 (Own hSize)

My Hough Transform/ Hough Lines In my Hough transform implementation, the biggest initial problem was that I had swapped the rho and theta axes for the accumulator. This caused the accumulator indices to be switched, which rendered the Hough space horizontally instead of vertically. I fixed this by swapping the accumulator indices both when defining the array and during the voting process.

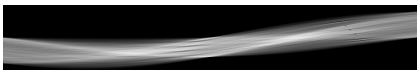


Figure 7: Hough Transform (Axis switched)

After correcting the axes of my Hough transform, another problem arose: an index out of bounds error (e.g., attempting to access index 2000 in an array of size 800). The code took around 5 to 7 minutes to run before producing the error, which led me to suspect something was wrong, as it should have taken a minute at most. I debugged this issue by adding an index check before performing the voting process on the accumulator. After implementing this check, the code ran smoothly and generated all the required images.

Unfortunately, another problem arose: the rho axis was twice as long as the theta axis, as seen in the "Hough Transform (Rho Scale=2, Theta Scale= $\pi/90$)." This implied that my detected Hough lines would not be as optimal as they could be. I contemplated changing either my rho or theta resolution and concluded that adjusting the theta resolution would be better since the rho resolution was already 2. After changing the theta resolution, I noticed more lines in the output compared to the original result.

For my Hough lines, I used the cv2.dilate feature but was initially confused about which kernel to use. After researching online, I found an article on ScienceDirect which mentioned that a 5x5 diagonal kernel allows for better detection when manipulating features (Taslimi et al., 2020). I did not change the nLines parameter because it seemed that a large number of lines were already being generated.

Figure 8: Hough Transform (Rho Res=2, Theta Res= $\pi/90$)

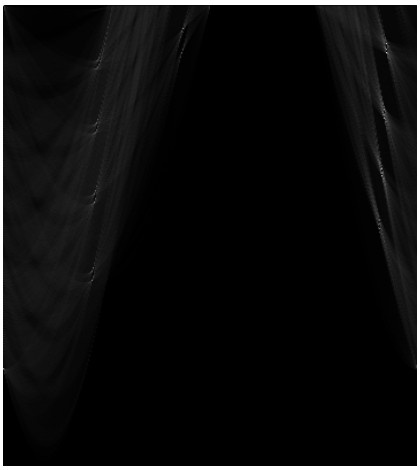
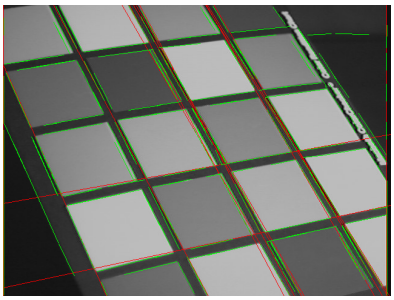
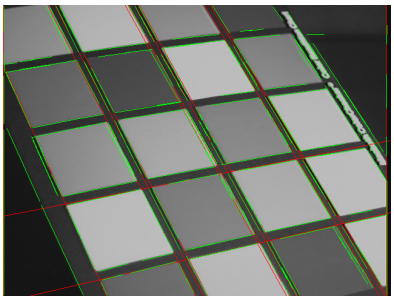


Figure 9: Hough Transform (Rho Res=2, Theta Res= $\pi/180$)



(a) Hough Lines (Rho Scale=2, Theta Scale= $\pi/90$)



(b) Hough Lines (Rho Scale=2, Theta Scale= $\pi/180$)

Discussed/ LLM Usage

Discussed:

Discussed with: Elijah, Jared, Lukas, Wilson, and Andrew, ZWei.

LLM Usage:

Used Gemini:

"How do I format my LaTex so it fits well with pictures?"

"Explain why atan2(y, x) covers all 4 quadrants."

"Why do we need to compare the different quadrants using non maximum suppression?"

"How do I put two images side to side on LaTeX?"

"How do we get Rcos(theta - phase) from xcos(theta) + ysin(theta)?"

"Why is my homemade segment lines different from cv2.HoughLinesP? Is it because of the threshold

"How do we create citations in LaTex?"

"What is the different between a 3x3 Sobel and 5x5 Sobel?"

References

Shrestha, A. K. (2022, September). *Implementation of canny edge detection using python*. Retrieved September 18, 2025, from https://www.anilkshrestha.com.np/blog/computer-vision/2022-09-22-canny-edge-detection

Taslimi, Z., Soroushmehr, S. R., Karimi, N., Samavi, S., & Najarian, K. (2020). Explainable ai for gard-u-net, a u-net based architecture for sarcoidosis granuloma segmentation. *Computers in Biology and Medicine*, *127*, 104084. https://doi.org/10.1016/j.compbiomed.2020.104084