CS 83: Computer Vision
Image Filtering and Hough Transform

Amittai Siavava

01/21/2024

Abstract

This project explores image manipulation with convolutions and the Hough transform.

Contents

Problem 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 the phase of the sinusoid to the point (x, y).

For an arbitrary point (x, y), its distance from the origin is $r = \sqrt{x^2 + y^2}$ and the equation of the line in (ρ, θ) space is $\rho = x\cos\theta + y\sin\theta$. We can derive the equivalent sinusoid:

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

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

$$\rho = r\left(\sin\gamma\cos\theta + \cos\gamma\sin\theta\right) \qquad \text{(since } x\text{, } y\text{, and } r \text{ form a right-angle triangle)}$$

$$\rho = r\sin(\gamma + \theta)$$

Therefore, the equivalent sinusoid has amplitude $r = \sqrt{x^2 + y^2}$ and a phase-shift $\gamma = \arctan\left(\frac{x}{y}\right)$.

Problem 2.

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

Representing the line in terms of (ρ, θ) reduces the space of the parameters. In slope form ($y = mx+c$), the slope m and the intercept c have a range of $[-\infty, \infty]$. The accumulator needed to store the votes for each possible line is therefore unbounded. In (ρ, θ) form, θ is an angle in the range $[0, 360]$ and ρ is bounded by the longest possible distance in the image, $\sqrt{W^2 + H^2}$, allowing us to more feasibly compute the accumulator array and find lines in the image.

(ii) Express the slope and the intercept in terms of (ρ, θ).

Recall that $\rho = x\cos\theta + y\sin\theta$. by re-arranging the equation, we get:

$$x\cos\theta + y\sin\theta = \rho$$
$$y\sin\theta = \rho - x\cos\theta$$
$$y = \frac{\rho}{\sin\theta} - \frac{\cos\theta}{\sin\theta}x$$

$$y = mx + c$$
$$\implies m = -\frac{\cos\theta}{\sin\theta}$$
$$\implies c = \frac{\rho}{\sin\theta}$$

Problem 3.

Assuming that the image points (x, y) are in an image of width W and height H,that is $x \in [1, W]$ and $y \in [1, H]$, what is the maximum absolute value of ρ, and what is the range of θ?

The maximum absolute value of ρ is the longest distance that can fit in the image, $\sqrt{W^2 + H^2}$.

The range of θ is $[0, 360]$.

Problem 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?

Point (x, y)	$r = \sqrt{x^2 + y^2}$	$\gamma = \arctan(x/y)$	Sinusoid
$(10, 10)$	$\sqrt{200}$	45	$\sqrt{200}\,\sin(\theta + 45)$
$(20, 20)$	$\sqrt{800}$	45	$\sqrt{800}\,\sin(\theta + 45)$
$(30, 30)$	$\sqrt{1800}$	45	$\sqrt{1800}\,\sin(\theta + 45)$

TABLE 1. Point and sinusoid values

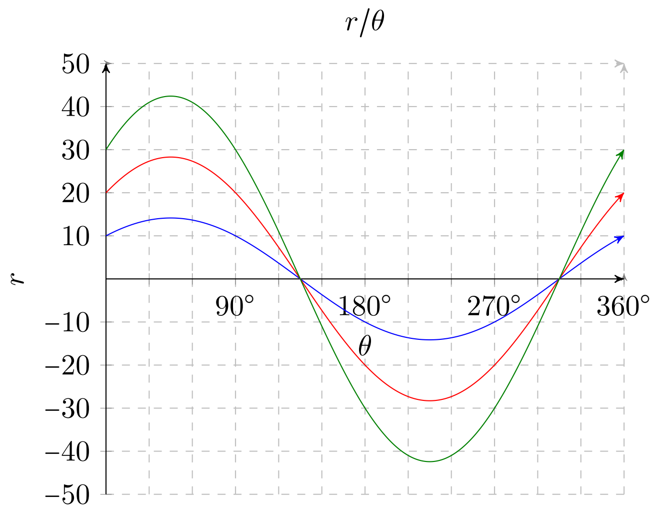

FIGURE 1. Generated sinusoidal waves

The intersection points are at $r = 0$ and $\theta \in \{135, 315\}$.

$$\cos\theta = -\sin\theta$$

$$\rho = x\cos\theta + y\sin\theta = x - y = 0 \qquad \text{(since } x = y\text{)}$$

$$m = -\frac{\cos\theta}{\sin\theta} = 1$$

$$c = \frac{\rho}{\sin\theta} = 0$$

5. Experiments and Results

5.1. **Design Challenges.**

(i) I spent a few hours debugging an issue with my hough transform, only to realize that I was writing *into* my hough accumulator array when I do non-maximum suppression in `myhoughLines.py` because I had not made a copy of the array. This was tough to identify, easy to fix — and I later ran into another student with the same issue.

(ii) Vectorizing code was pretty interesting, especially digging into `numpy` documentation to figure out how to compose certain operations to get a desired result, and later realizing that there's yet another function that combines two or more of those operations. A huge lifesaver here was `np.where`, which I didn't know existed before this assignment. I linked specific documentation links where relevant in the code.

5.2. **Experiments.** I played around with some of the parameters to see how the results are affected.

(i) **Gaussian kernel variance (σ):** Increasing the value of σ improves the image detection up to a point, then flattens out the line once σ goes beyond the optimal value. I found the best results with $\sigma = 2$ to work well for most of my images, although $\sigma = 3$ seemed to work better on one image.

(ii) **Threshold:** Raising the threshold prunes erratic lines in the image, but if the threshold is too high, accurate lines are pruned too (see image). I found the best results with a threshold of 0.3.

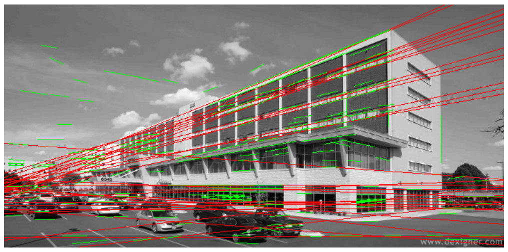
Figure 2. threshold = 0.03

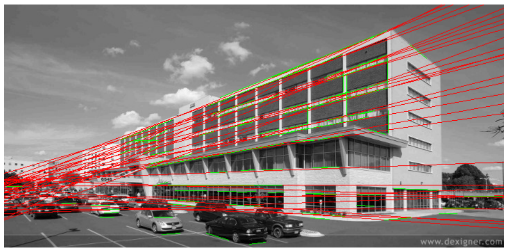
Figure 3. threshold = 0.3

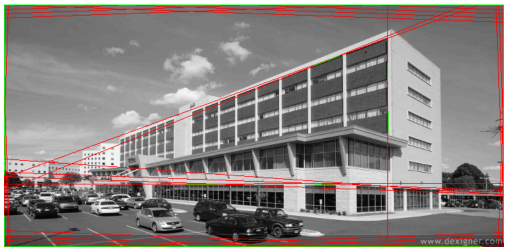
Figure 4. threshold = 0.5

(iii) **Theta Resolution:** Increasing the resolution of θ increases the accuracy of the lines found in the image, but it also increases the computation time. I found the best results with a resolution of $\frac{\pi}{180}$, but I settled on $\frac{\pi}{90}$ because the reduction in accuracy is not too extreme, and it computes much faster.

(iv) **Rho Resolution:** Increasing the ρ-resolution also increases the accuracy, but increases computation time. I thought ρ-resolution of 1 was reasonable enough since the results looked good.

(v) **Number of Lines:** In general, finding more lines allows us to capture most of the lines in the image. However, it also eventually starts to make the image noisy as multiple lines that are close together get captured.

The number of lines also needs to be tuned for each image, as a specific number of lines would work well in a specific image, then result in too much noise in another image.

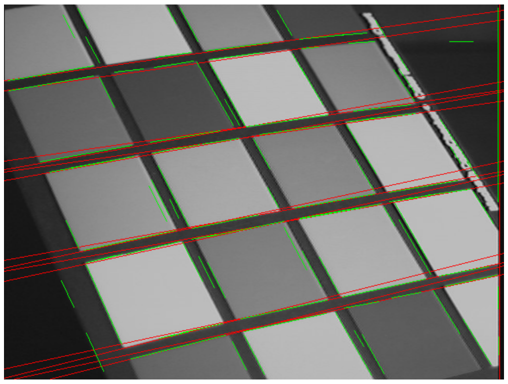
Figure 5. 15 lines (misses all vertical lines)

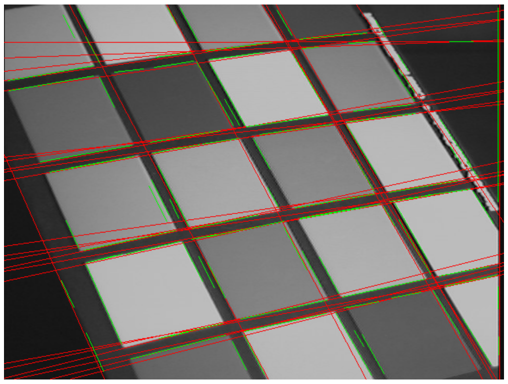
Figure 6. 30 lines (a bit noisy but captures most)

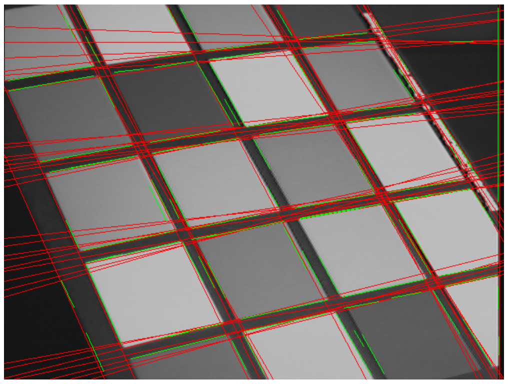
Figure 7. 50 lines (too noisy)

5.3. **Results.** These were the results I got for image 08.

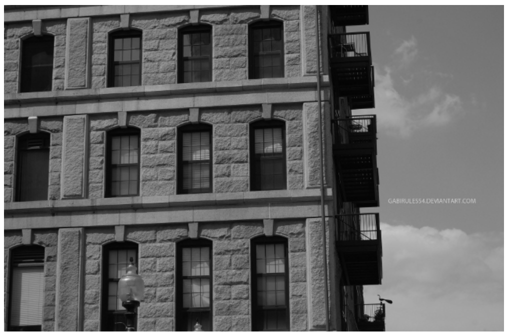

Figure 8. Original Image

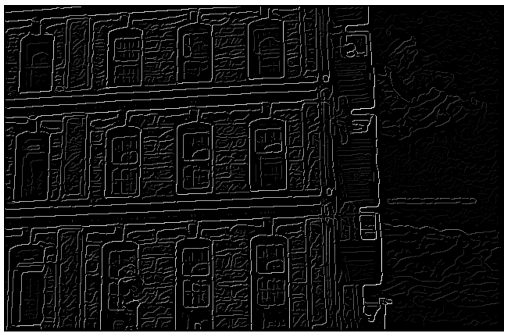

Figure 9. Edges

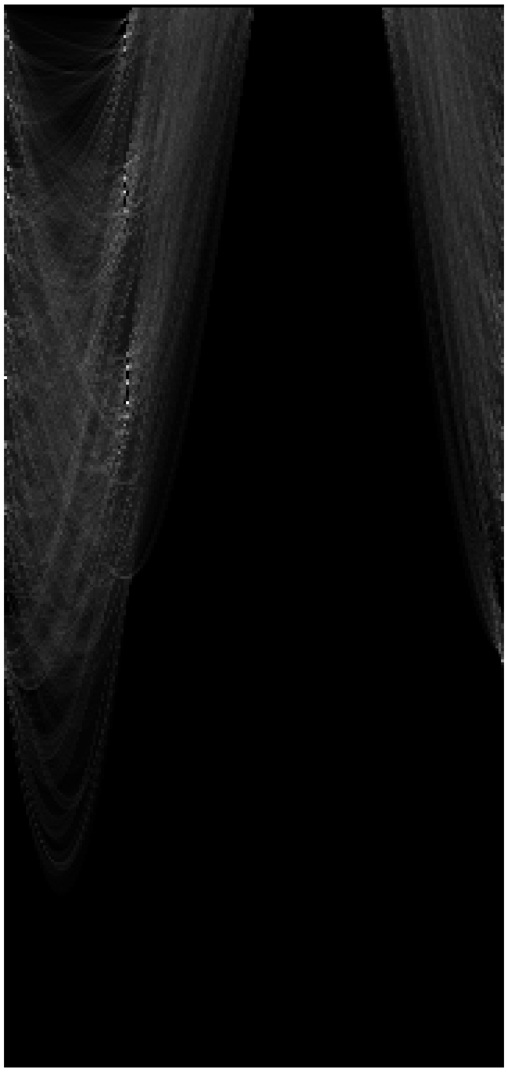

Figure 10. Hough Transform

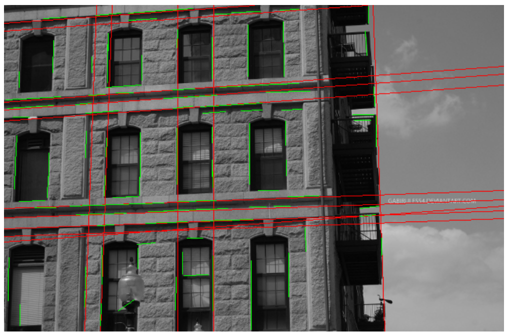

Figure 11. Lines