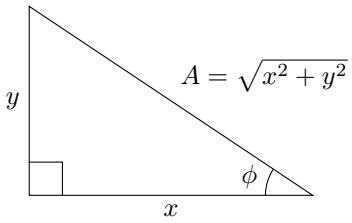
CS 83/183: Assignment 1

January 21, 2024

Jessie Li

2: Theory Questions

Part 1



Let A be the length of the hypotenuse of a right triangle with legs of length x and y, and ϕ be the angle opposite y and adjacent to x.

$$A = \sqrt{x^2 + y^2} \quad \text{and} \quad \phi = \arctan(y/x)$$

Multiplying the right hand side of the equation $\rho = x\cos\theta + y\sin\theta$ by A/A,

$$\begin{aligned} x\cos\theta + y\sin\theta &= A(\frac{x}{A}\cos\theta + \frac{y}{A}\sin\theta) \\ &= A(\cos\phi\cos\theta + \sin\phi\sin\theta) \\ &= A\cos(\theta - \phi) \end{aligned}$$

Part 2

We prefer to parameterize lines in terms (ρ, θ) instead of the slope and intercept (m, c) because the (m, c) space is infinite with $m, c \in (-\infty, \infty)$ whereas θ and ρ can be restricted to $[0, 2\pi)$ and $[0, \rho_{max}]$ respectively.

Since the x-intercept $a = \rho/\cos\theta$ and y-intercept $b = \rho/\sin\theta$, the slope m can be expressed as

$$m = b/a = \cos\theta/\sin\theta = \cot\theta$$

and the intercept c as

$$c = b = \rho/\sin\theta$$

Part 3

The maximum absolute value of ρ is the diagonal length of the image:

$$|\rho| \leq \sqrt{W^2 + H^2}$$

Letting ρ range from 0 to $\sqrt{W^2 + H^2}$, the range for θ would be $[0, 2\pi)$.

Part 4

Under the Hough transform,

$$\begin{aligned}(10,10) \quad &\text{becomes} \quad \rho = 10\sin\theta + 10\cos\theta \\ (20,20) \quad &\text{becomes} \quad \rho = 20\sin\theta + 20\cos\theta \\ (30,30) \quad &\text{becomes} \quad \rho = 30\sin\theta + 30\cos\theta\end{aligned}$$

The three waves intersect at $\rho = 0$ and $\theta = 3\pi/4$ in the Hough space, corresponding with

$$m = \tan(3\pi/4) = 1 \quad \text{and} \quad c = \rho/\sin\theta = 0$$

The corresponding line, $y = x$, passes through all three points.

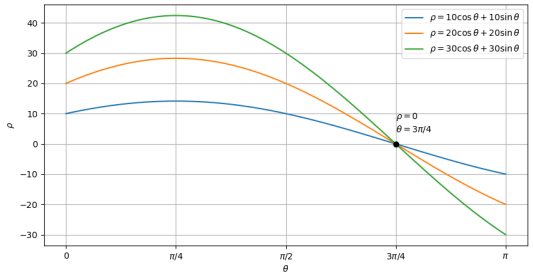


Figure 1: Hough transformations of (10, 10), (20, 20), and (30, 30), and their intersection.

4: Write-up

Default parameters

- `sigma` = 2
- `threshold` = 0.03
- `rhoRes` = 2
- `thetaRes` = $\pi/90$
- `nLines` = 15

Outputs

Below are all outputs for `img1`, including intermediate outputs from `myImageFilter`, `myEdgeFilter`, and `myHoughTransform` with the default parameters above. Figures 6, 7, 8, and 9 are outputs of `houghScript.py`.

myImageFilter

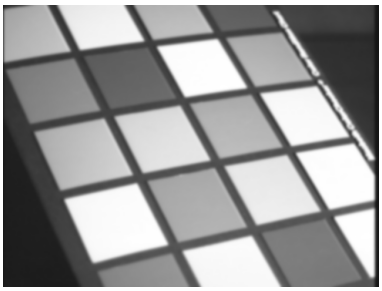


Figure 2: Gaussian blur.

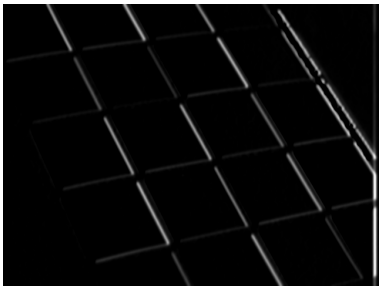


Figure 3: Image gradients in the x direction with a horizontal Sobel filter.

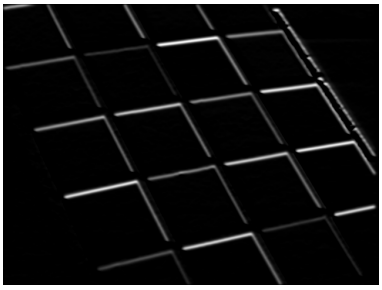


Figure 4: Image gradients in the y direction with a vertical Sobel filter.

myEdgeFilter

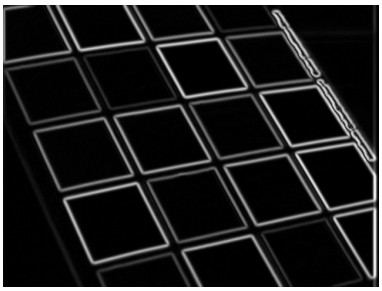


Figure 5: Edges (gradient magnitudes) before non-maximal suppression.

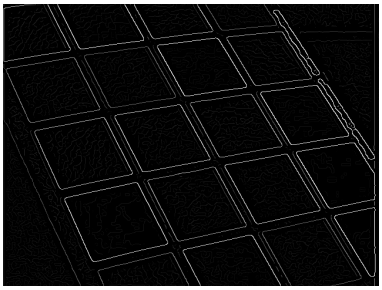


Figure 6: Edges after non-maximal suppression.

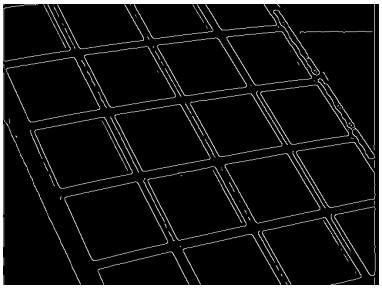


Figure 7: Edges, ignoring pixels with magnitudes less than the threshold.

myHoughTransform

Figure 8: Result of applying Hough transform to the thresholded edge magnitude image.

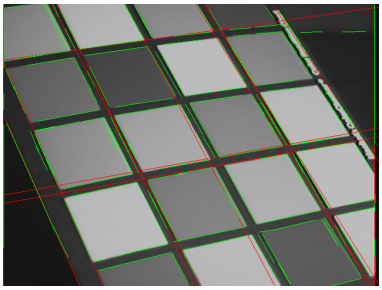
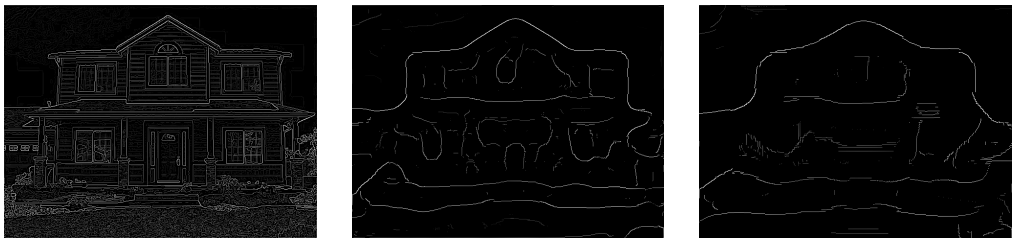


Figure 9: Hough lines (red). Green line segments represent the output of the OpenCV `HoughLinesP` function.

Experimenting with different parameters

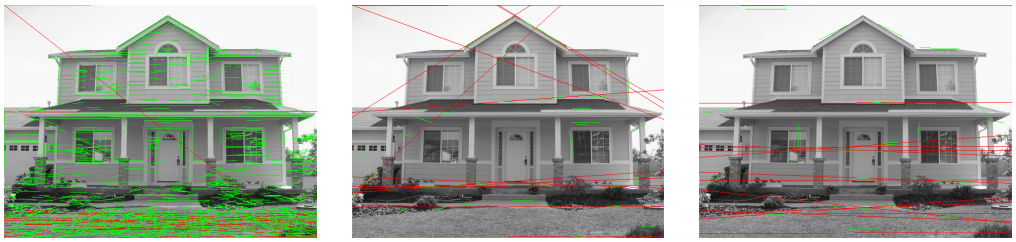
sigma

Increasing the value of `sigma` blurs the images more, which helps reduce small details and noise. This tends to result in more accurate lines for images with high-contrast edges but significant levels of noise, such as `img6`, `img7`, and `img8`. For these images, blurring with large `sigma` values removes distracting pixels. However, when `sigma` is set too high, images start to lose important edges.



(a) sigma = 1 (b) sigma = 16 (c) sigma = 32

Figure 10: Edge magnitude images for `img7` with different values of `sigma`. Increasing `sigma` from 1 to 16 reduces noise and helps focus the prominent edges. At `sigma` = 32, the image starts to lose some of the important lines.



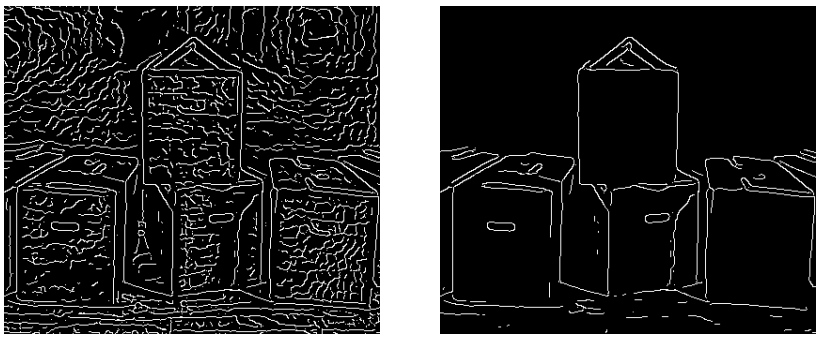
(a) sigma = 1 (b) sigma = 16 (c) sigma = 32

Figure 11: Hough lines for `img7` with `sigma` = 1, 16, and 32.

Images with higher contrast are generally more robust to changes in the value of `sigma`. For example, the lines for `img2`, a picture of a white book against a dark background, are similar for `sigma` = 1 and 32. The dark windows in `img8` also captured well under a range of high `sigma` values.

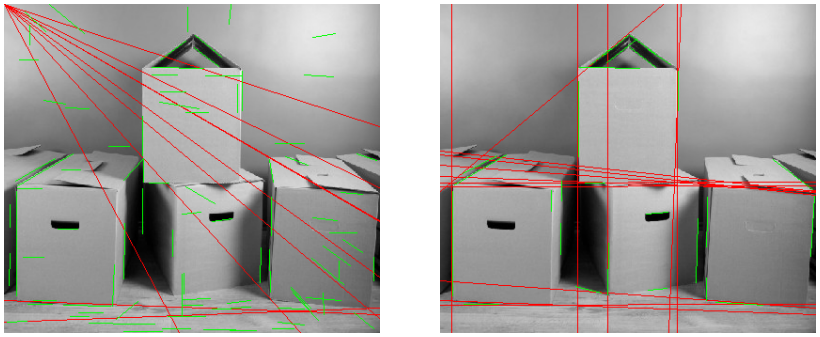
threshold

As with `sigma`, raising the `threshold` reduces noise in the edge image by filtering out weak edges formed by pixels with low gradient magnitudes. Higher thresholds help focus the primary edges, but setting the threshold too high may also result in too few detected edges.



(a) threshold = 0.01 (b) threshold = 0.1

Figure 12: Thresholded edge magnitude images for `img5` with low and high `threshold` values. `threshold` = 0.1 (right) is far less noisy compared to `threshold` = 0.01.

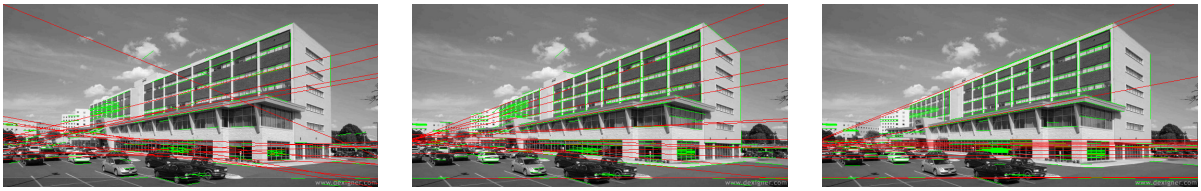


(a) threshold = 0.01 (b) threshold = 0.1

Figure 13: Hough lines for `img5` from low and high thresholds. The higher threshold Hough lines (right) fit better to the edges along the boxes.

thetaRes and rhoRes

Higher resolutions (proportional to `1/thetaRes` and `1/rhoRes`) increases computational complexity, but generally improves the Hough lines. For images with many lines nearly parallel and spatially close (ex. `img9`), dividing the Hough space into smaller pieces allows those edges to be represented more accurately. However, setting the resolution too high (`thetaRes` and `rhoRes` too low) could also spread votes for a single edge across multiple cells, making it difficult to rank the parameters after the transformation.



(a) thetaRes = $\pi/45$, rhoRes = 4 (0.5x default)

(b) thetaRes = $\pi/180$, rhoRes = 1 (2x default)

(c) thetaRes = $\pi/360$, rhoRes = 0.5 (4x default)

Figure 14: Hough lines (red) with different resolutions for `img9`. Increasing the resolution from 0.5x to 2x initially improves the fit. Around 4x, edges start to split into multiple lines.

nLines

`nLines` determines the number of lines to display from the output of the Hough transform. For images with fewer straight edges, like `img4`, decreasing this parameter would help remove some of the extraneous Hough lines, whereas an image with many edges, like `img8`, might be better described with a higher value of `nLines`.

General observations and ideas for improvement

- With the model $y = mx + b$, this Hough transform performs well on images with straight edges, but poorly on images with curves (ex. `img2` and `img10`). For these images, a different model (ex. a circle, $x^2 + y^2 = r^2$) might lead to more accurate results.
- This implementation seems to work better on images with higher contrast, where edges have much larger gradient magnitudes relative to flat areas. Raising the contrast of the images after applying `myEdgeFilter` and increasing the `threshold` before removing pixels below it might help eliminate some of the trivial lines before the Hough transformation is applied.
- Rather than weighing all votes equally, weighing votes proportional to gradient magnitude might reduce the effect of votes cast by pixels on minor edges while emphasizing votes from pixels on major edges.
- Across all images, the Hough transformations consistently pick up lines near the very edges of the images that are less visually significant that those in the middle. Perhaps blurring the edges more or trying a different padding scheme would mitigate this issue.
- Using vectorized operations instead of traditional for loops improves the run time.