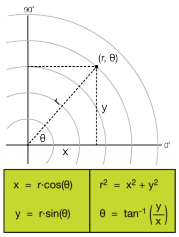
Bryan Zhao
16720-A Fall '18, HW#1
Due: 11:59pm, Friday, 14 September 2018

2.1 Show that if you use the line equation $xcos\theta + ysin\theta - \rho = 0$, 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).

If we use the line equation $xcos\theta + ysin\theta - \rho = 0$, we can rewrite the line equation as $xcos\theta + ysin\theta = \rho$. Recalling the relationship between polar coordinates and Cartesian coordinates, the amplitude of the sinusoid can be expressed as $\sqrt{x^2 + y^2}$. Drawing a triangle with the Cartesian components more clearly illustrates this relationship.



Source: http://xaktly.com/MathPolarCoordinates.html

From this relationship, it can be derived that:

$$xcos\theta + ysin\theta = \sqrt{x^2 + y^2} * \frac{xcos\theta}{\sqrt{x^2 + y^2}} + \sqrt{x^2 + y^2} * \frac{ysin\theta}{\sqrt{x^2 + y^2}}$$

and one can see from the geometric relationships that there exists a $cos\alpha = \frac{x}{\sqrt{x^2+y^2}}$ and $sin\alpha = \frac{y}{\sqrt{x^2+y^2}}$. With this substitution, and knowing that $cos\alpha * cos\theta + sin\alpha * sin\theta = cos(\alpha - \theta)$, we can rewrite the previous equation as the following,

$$\rho = \sqrt{x^2 + y^2} cos(\alpha - \theta)$$

where the amplitude of the sinusoid, typically represented as A, is $\sqrt{x^2 + y^2}$, and the phase of the sinusoid is $\theta = \arctan \frac{y}{x}$. This complements the generic sinusoid form of $A * cos(\alpha + \phi), or A * cos(\alpha + \phi + \frac{\pi}{2})$.

2.2 Why do we parameterize the line in terms of $\rho; \theta$ instead of slope and intercept (m, c)? Express the slope and intercept in terms of ρ and θ.

When we parameterize the line in terms of slope and intercept (m, c), we end up with the equation $\frac{-mx}{c} + \frac{y}{c} = 1$ from the original line equation, $y = mx + c$. With vertical lines, the term c (or the y-intercept) in the parameter space goes to 0, which results in an infinite value and an unbounded parameter space.

On the contrary, when we parameterize the line with (ρ, θ), the line equation becomes $xcos\theta + ysin\theta = \rho$ (as stated in the problem). Simple algebra yields the following:

$$ysin\theta = -xcos\theta + \rho$$

, which becomes:

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

and one can see that in the above equation, if m and c are utilized for the line parameters, $m = \frac{cos\theta}{sin\theta}$ and $c = \frac{\rho}{sin\theta}$, which approach infinity as $sin\theta$ approaches 0 – or as θ approaches 0. In other words, the parameter space for (m, c) can approach infinity, while it is bounded from $[0, \rho_{max}]$ for ρ and $[0, 2\pi]$ for θ.

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

If the image points have width W and height H, the maximum absolute value of ρ is defined by the magnitude between the two components of width and height. In other words,

$$\rho_{max} = \sqrt{W^2 + H^2}$$

while θ has a range of $[0, 2\pi]$.

2.4 For points $(10, 10)$, $(15, 15)$, and $(30, 30)$ in the image, plot the corresponding sinusoid waves in Hough space $(\rho; \theta)$ and visualize how their intersection point defines the line (what is (m, c) for this line?). Please use MATLAB to plot the curves and report the result in your write-up.

After plotting all of the ρ results from $\theta = 0 : \frac{\pi}{90} : 2\pi$, the following MATLAB plot was generated:

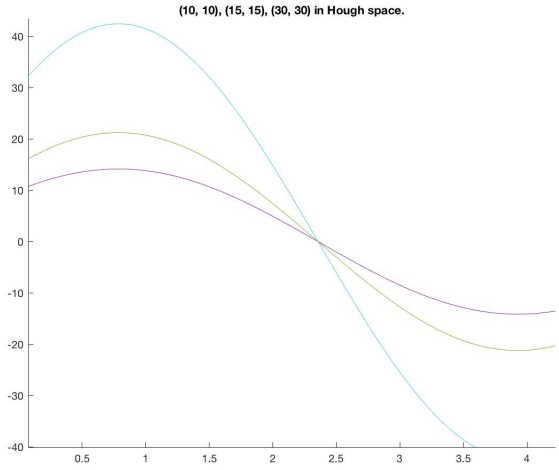


Figure 1: MATLAB results for sinusoid.

and the intersection of the three points lies at $\rho = 0$, $\theta = \frac{3}{4}\pi$. Recalling that $xcos\theta + ysin\theta - \rho = 0$, we can rewrite this equation as $y = \frac{\rho}{sin\theta} - x\frac{cos\theta}{sin\theta}$, and note that $m = -\frac{cos(\frac{3}{4}\pi)}{sin(\frac{3}{4}\pi)} = 1$, while $c = \frac{\rho}{sin\theta} = 0$. This results in a simple line equation of $y = x$.

Section 3, Implementation Results

After completing all of the MATLAB implementation, a good set of parameters for the provided 'img01' were set to the following:

$$\sigma = 1.3$$
$$threshold = 0.3$$
$$rhoRes = 1$$
$$thetaRes = \frac{\pi}{180}$$
$$nLines = 300$$

and these are the resulting images.

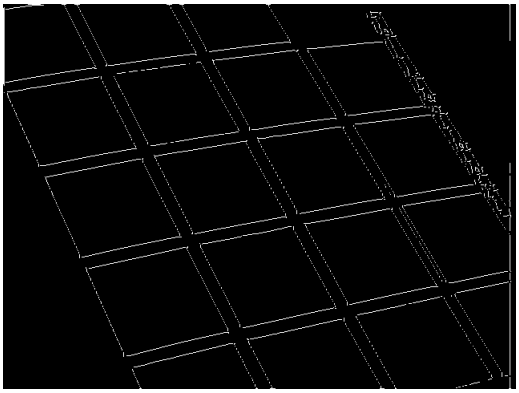


Figure 2: Edge intensity and threshold.

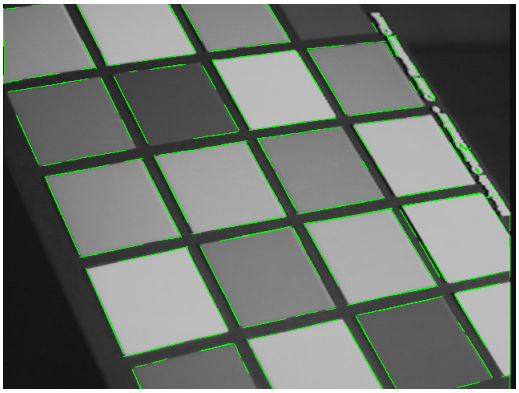


Figure 3: Hough line segment results.

Additional results and experiments are discussed in the next section.

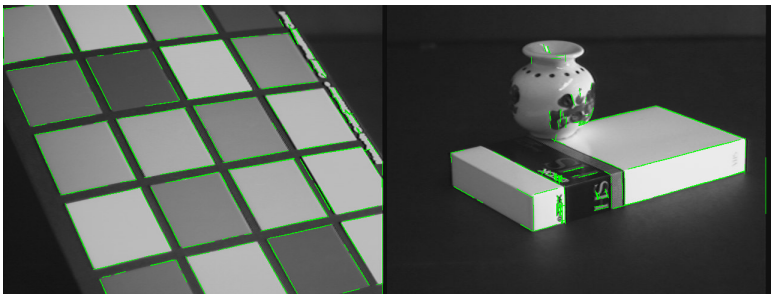
Section 4, Experiments

With the first image, img01, various parameters were tabulated and tested to see what the ideal set of parameters are for detecting lines in that specific image. The effects of varying these parameters on the same image, as well as different images, are discussed in the following.

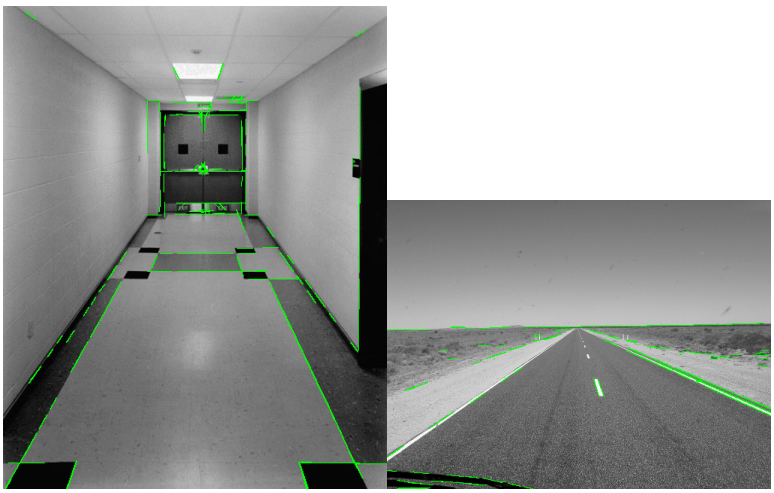
	sigma	thresh	rhoRes	thetaRes	nLines	FillGap	MinLength
Img1	2	0.03	2	Pi/90	50	5	10
Img1	1	0.3	2	Pi/360	75	10	10
Img1	1.5	0.3	1	Pi/180	100	10	20
Img1	1.5	0.15	1	Pi/180	300	5	10
Img1	1.3	0.12	1	Pi/180	400	10	20
Img1	1.3	0.12	1	Pi/180	400	5	10
Img1	1.3	0.12	5	Pi/180	400	5	10

Figure 4: Result of img02 with updated parameters.

After implementing all of the programs, it was clear that the code did not work equally well on all of the images with a single set of parameters. For instance, with the parameters that were shown previously in section 3, these are a few of the resulting images with Hough line segments:



(a) Result for img01. (b) Result for img02.



(c) Result for img03. (d) Result for img04.

Figure 5: Hough line segments with constant parameters.

The optimal set of parameters for the different images were vastly influenced by various factors, which include the object shapes, level of noise, changes in color or intensity, and perspective changes for the image.

For images that had any curvature – in particular img02 with the ceramic vase – the optimal parameters for img01 were insufficient in detecting the object boundaries. It can be observed that the edges in img02 (Figure 4-b) on the left-hand side of the image did not get properly captured by the algorithm. Evidently, this implementation of the Hough transform did not account for other geometric shapes, such as circles or ellipses. There exist almost no green lines that are accurately placed over the spherical rim of the vase, or the curved body of the vase. However, given a much smaller threshold of 0.3, a larger number of lines ($nLines = 500$), an increased resolution in ρ and θ to the values $= 1$ and $= \frac{pi}{360}$, and a smaller $\sigma = 1.1$, the resulting image more accurately traces the boundaries.

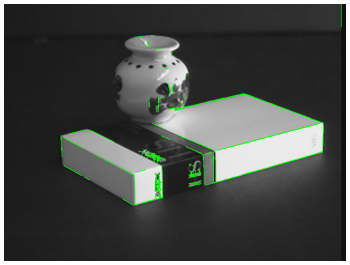


Figure 6: Result of img02 with updated parameters.

In this photo, the left edge of the box was detected, but the algorithm still did not catch any of the major curvature of the vase.

Intuitively, it became more clear after experimenting with various parameters that increasing *rhoRes* and *thetaRes* yielded better results. There were more details and accurately captured edges, but at the cost of run-time. In fact, in the case of img01, doubling the resolution of rhoRes ended up nearly doubling the run-time of the main MATLAB script.

Additionally, lowering the threshold for the various images often introduced more noise into the resulting Hough line segment photos. For most of the images, a threshold of 0.1 to 0.2 worked relatively well. Thresholds below 0.05 often introduced miniature lines that were oriented at the wrong angles, which can be simply attributed to the background noise in images, particularly in the case of img09 with clutter due to cars, trees, and buildings.

Interestingly, decreasing the standard deviation for the Gaussian blur (*sigma*) also had a similar effect of introducing more noise into the resulting photographs. Naturally, a higher standard deviation results in a greater spread, or a greater blur across the pixels of the image, which reduces fine details and noise in the image. Therefore, the noise in the images were highly sensitive to changes in both *sigma* and *threshold*.

Increasing the number of lines to the main MATLAB script also increased the efficacy of the line segment identification. With a greater number of lines, the algorithm which implements non-maximal suppression is able to select a greater number of local maxima from the Hough space. However, this also comes at the cost of longer execution time, since the program has to loop through more indices of the Hough accumulator matrix.

Finally, was also observed that a bottlenecks in the algorithm run-time was the implementation of the myHoughTransform() function. Specifically, the algorithm step in which the accumulator was incremented in which $H(\rho, \theta) = H(\rho, \theta) + 1$ took the longest, depending on the resolution of θ and ρ. Since the function implemented two for loops along the indices of all the edges and along all the theta values, it took a relatively long time to execute – on the order of 20 seconds for img01. A major improvement to this part of the algorithm would be to vectorize the operations for the Hough space accumulator, as MATLAB's vectorized capabilities are far more efficient than running for loops. In fact, if all of the filtering and suppression (e.g. NMS) algorithms can be vectorized, it would significantly decrease the runtime of the main MATLAB script. However, the challenge lies in conceptualizing the problem in a vectorized sense, as it is easier to

comprehend loops and basic iteration along matrices. Future works should utilize vectorized code if writing a Hough transform implementation in MATLAB. For other languages that may be more computationally efficient, such as C++, it is more feasible to get quicker results with finely-tuned parameters with non-vectorized methodology.