Computer Vision Lab #2
Hough Transform

Prof. Ricardo Fabbri, Ph.D.*

Polytechnic Institute at the Rio de Janeiro State University
http://wiki.nosdigitais.teia.org.br/CV

November 23, 2013

Abstract

- Detect lines and circles in an image using the Hough transform
- Implement your own version and compare to SIP and OpenCV

The images you will need for the lab can be downloaded from the course website. All extra work will be considered for bonus grade points.

1 Implement the Hough Transform for Multiple Line Detection

In this problem, you will implement the Hough Transform to detect lines, see Figure 1. We will use the normal form of a line, Equation 1, as explained in class

$$\rho = x \cos\theta + y \sin\theta \tag{1}$$

The pseudocode for the Hough transform algorithm is as follows

*Based on Image Understanding 2011 lab material from Ben Kimia, Brown University

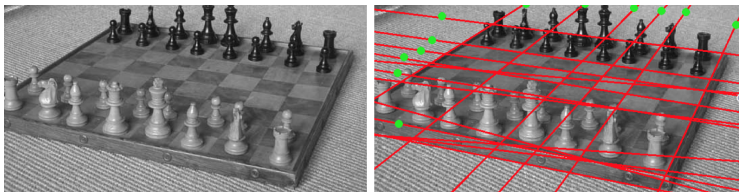

Figure 1: (left) Checkerboard image (right) Sample output of line Hough transform

1: I, Binary Edge Map size [M x N]
2: $\rho_d[r] \in [0, \sqrt{N^2 + M^2}], \theta_d[t] \in [0, pi]$, Arrays containing the discretized intervals of the parameter space
3: A be the accumulator, matrix initialized to all zeros, size [R x T] where R is the length of the array of $\rho_d[r]$ and T is the length of the array of $\theta_d[t]$

4: **for** $i = 1 \rightarrow M$ **do**
5: **for** $j = 1 \rightarrow N$ **do**
6: **if** $I(i, j) == 1$ **then**
7: **for** $t = 1 \rightarrow T$ **do**
8: $\rho = i \cos(\theta_d[t]) + j \sin(\theta_d[t])$
9: Find index r so that $\rho_d[r]$ is closest to ρ
10: $A[r, t] \leftarrow A[r, t] + 1$
11: **end for**
12: **end if**
13: **end for**
14: **end for**
15: Find all local maxima (r_k, t_k) such that $A[r_k, t_k] > \tau$, where τ is user defined threshold
16: The output is a set of lines described by $(\rho_d[r_k], \theta_d[t_k])$

Some comments on lines of the algorithm:

Line 1: This is the output of a gradient-based edge detector. You can use SIP's `edge` function with the Sobel operator. Does the algorithm work better if you use `thin` on top of the edge image? Show the edge images for all results.

Line 2: When you define these arrays you are discretizing the parameter space of ρ and θ using steps of $\Delta\rho$ and $\Delta\theta$. Choose appropriate step sizes that yield acceptable and manageable resolution of the parameter space.

Line 9: The value computed for ρ will not exactly be in your parameter space, so you have to choose either to round, ceil, or floor the value.

Line 15: When defining your threshold, think about how many edge points would have to vote for a line.

Requirement for your report:

1. Display your accumulator space A as an image. This is also called a sinogram.

2. Plot all lines found on top of 3 different images of your choice (buildings, mechanical pieces), such as those in Figure 2. You can use SIP's `imshow` followed by a plot command, as long as you set the plot to be persistent.

Figure 2: You should run your line detection algorithm on these images or very similar ones.

3. What does an edge point (x, y) in the image correspond to in the $(\rho, theta)$ parameter space?

4. Comment on the effect of discretization of the parameter space, and the threshold you used

5. If you were to implement this multiple line detection with RANSAC, what type of shortcomings can you predict? How would you use RANSAC to detect multiple lines?

6. Compare your implementation to SIP's `hough` and `ihough` functions. What are some flaws of SIP's code that you can see?

2 Implement Hough Transform for Multiple Circle Detection

The Hough transform is a very general approach that we can use to detect any shape that has (short) a parametric form! We sill use the same aforementioned approach but now to detect circles, see Figure 3. A circle with radius r and center (a, b) can be described with parametric equations, Eq. 2.

$$\begin{cases} x = a + r\cos\theta \\ y = b + r\cos\theta \end{cases} \tag{2}$$

If an image contains many points (x, y), some of which fall on the perimeters of circles, then the job of the search program is to find parameter triplets (a, b, r) to describe each circle. We will use the same approach as in Algorithm 2 for lines. Some comments on lines of the Hough algorithm:

Line 2: Here you will have three arrays that define the discretization of a_d, b_d, r_d.

Line 3: Your accumulator will be a 3D matrix of all zeros, again, the size of it will be $\text{length}(a_d) \times \text{length}(b_d) \times \text{length}(r_d)$, the 3D case will be more expensive in terms of both time and memory, so be judicious in your discretization of the parameter space

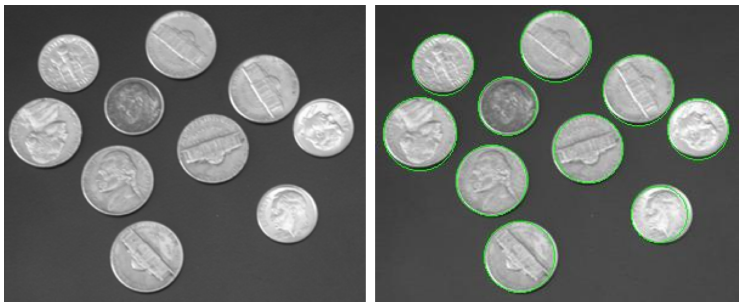

Figure 3: (left) Input of coins, (right) Sample output of circle Hough transform.

Line 7: To evaluate Equation 2 you will have two loops instead of one, one loop will be over r_d and the other will be over a discretization of θ, where θ ranges from 0 to 2π.

Line 9: After evaluating Equation 2 you will have two parameters a, b and you will again have to find the closest bin in the 3D accumulator $A[i, j, k]$, keeping in mind you are looping over r_d so you already know the index of one parameter.

Line 15: When defining your threshold, think about how many edges would have to vote for a circle. Also keep in mind you have to look around 3 dimensions when finding the local maxima

Requirement for your report:

1. Plot all found circles on top of the input images of Figure 4. There are many options to plot a circle on top of an image, you can use `plot` on top of `imshow` or even use SIP's `mogrify` to draw a circle directly in the image (as in the mogrify demo).

2. What does an edge point (x, y) in the image correspond to in the (a, b, r) parameter space?

3. Comment on the effect of discretization of the parameter space, and the threshold you used

3 Comparison with OpenCV

1. Install OpenCV from Git following instructions at `http://wiki.nosdigitais.teia.org.br/OpenCV`

2. Learn how to run OpenCV hough transform functions from the commandline, for both circles and lines

3. Compare the results and performance with your implementation. Report running times and mainly on the quality of the output.

4. Bonus: open the source code for the hough transform in OpenCV. What can you identify? Describe two optimization tricks being used.

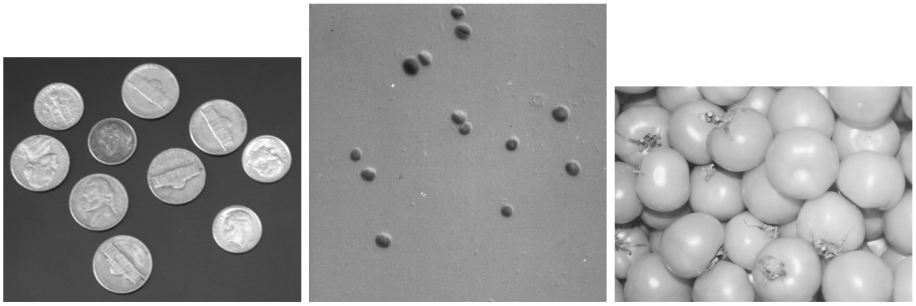

Figure 4: You should run your circle detection algorithm on these images or very similar ones.

Bonus Challenge – Accelerated Hough Transform – 3 extra points One way of reducing the computation required to perform the Hough transform is to make use of gradient information (magnitude and/or direction) which is often available as output from an edge detector. Explain in detail how you would use the gradient to accelerate the Hough transform, and provide an implementation.