CSC 249 Machine Vision: Homework 3
Bryce Elizabeth Yahn

Problem 1 (10 pts): *Prove the simple, but extremely useful, result that the perpendicular distance from a point* (u,v) *to a line* $ax + by + c = 0$ *is given by* $|au + bv + c|$ *if* $a^2 + b^2 = 1$.

To find the distance from a point (u,v) to a line L1 , you must find a line L2 parallel to L1 that runs through (u,v). Then you must find a third line L3 perpendicular to the first two lines, and find the points (x_1,y_1) and (x_2,y_2) where L3 intersects with L1 and L2 respectively. The distance between these points is the perpendicular distance from the point (u,v) to the line L1.

L1: $ax + by + c = 0$
L2: $a(x-u) + b(y-v) = 0$
The slope of L1 and L2 is $-\frac{a}{b}$, so L3 is $ay - bx = 0$
These can be rewritten as
L1: $y = -\frac{a}{b}x - \frac{c}{b}$ L2: $y = -\frac{a}{b}x + \frac{au}{b} + \frac{bv}{b}$ L3: $y = \frac{b}{a}x$

Find the point (x_1,y_1) at the intersection of L3 and L1 by setting L3 = L1

$\frac{b}{a}x_1 = -\frac{a}{b}x_1 - \frac{c}{b}$
$x_1(\frac{b}{a} + \frac{a}{b}) = -\frac{c}{b}$
$x_1(\frac{a^2+b^2}{ab}) = -\frac{c}{b}$
$x_1(\frac{1}{ab}) = -\frac{c}{b}$
$x_1 = -ac$
$y_1 = \frac{b}{a}x_1$
$y_1 = \frac{b}{a} * -ac$
$y_1 = -bc$

Find the point(x_2,y_2) at the intersection of L3 and L2 by setting L3 = L1

$\frac{b}{a}x_2 = -\frac{a}{b}x_2 + \frac{au}{b} + \frac{bv}{b}$
$x_2(\frac{b}{a} + \frac{a}{b}) = \frac{au}{b} + \frac{bv}{b}$
$x_2(\frac{a^2+b^2}{ab}) = \frac{au}{b} + \frac{bv}{b}$
$x_2(\frac{1}{ab}) = \frac{au}{b} + \frac{bv}{b}$
$x_2 = a(au + bv)$
$y_2 = \frac{b}{a}x_2$
$y_2 = \frac{b}{a} * a(au + bv)$
$y_2 = b(au + bv)$

Find the distance d between (x_1,y_1) and (x_2,y_2) using the equation $d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$

$$d = \sqrt{(-ac - a(au + bv))^2 + (-bc - b(au + bv))^2}$$

$$d = \sqrt{(-a(au + bv + c))^2 + (-b(au + bv + c))^2}$$

$$d = \sqrt{a^2(au + bv + c)^2 + b^2(au + bv + c)^2}$$

$$d = \sqrt{a^2 + b^2 * (au + bv + c)^2}$$

$$d = \sqrt{(au + bv + c)^2}$$

$$d = |au + bv + c|$$

Therefore, if $a^2 + b^2 = 1$, then the perpendicular distance from a point (u, v) to a line $ax + by + c = 0$ is given by $|au + bv + c|$

Problem 2 (30 pts): *The Hough transform was discussed in class as a way to detect lines in images by voting in a transformed parameter space. For the case of lines, this is a 2D parameter space. Now, consider different possible parameterizations of boundaries of interest. In particular, consider a circle or a quadratic curve. These examples may be used in various applications. In this problem, we will use shoe-print images as examples.*

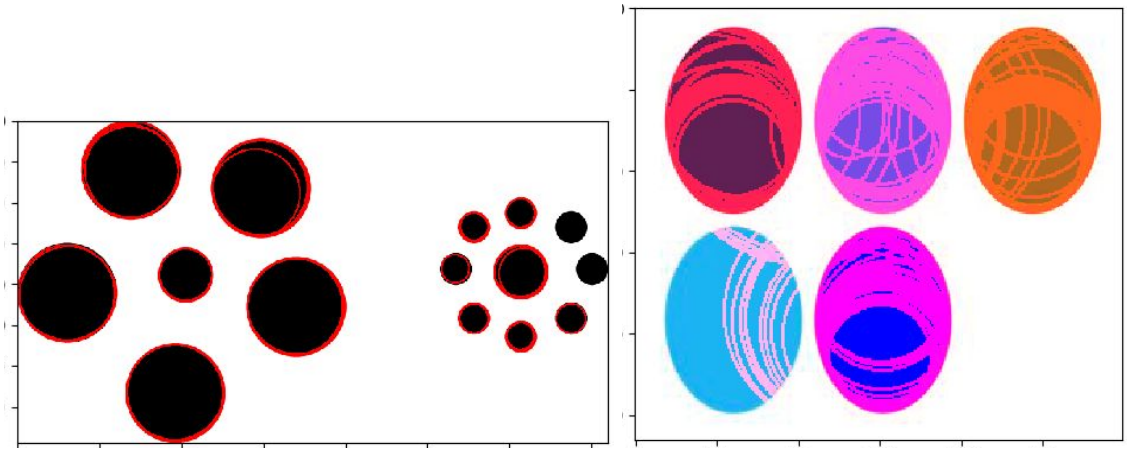
1. *Select either the circle or quadratic curve and derive the Hough transform for that case.*
    a. I selected the circle for which the parameter space is a,b,r where a represents the x position or row of the center of the circle, b represents the y position or column of the center of the circle, and r represents the radius of the circle.
2. *Implement your own Python routine for the boundary shape you chose and apply it to the appropriate image in shoeprint/. You may need to think behind simple Canny edge detection to provide sufficient input to the Hough transform.*

a. 

3. *Are you able to get it to work? Discuss the benefits and drawbacks of the Hough transform in the context of the boundary shape you chose and the image domain you worked with. Supplement the discussion with other images if needed. Max 1 page.*

The image above is the result of drawing all of the circles from the parameter bins which received 85% or more of the maximum number of votes. This does a decent job of identifying the main circles of interest. However, the concentric circles are somewhat difficult, because it seems to match the inner edge of a circular band with its outer edge on the other side, producing a circle that is approximately the correct size, but not centered properly. This also happened in a less noisy image without concentric circles but with a lower threshold. Another issue was the noise in the image. The image is very noisy and has some circular points of noise. In order to identify the important circles, I had to adjust the canny edge detector to make it subdue most of the information in the image.

Using the Hough transform was good because it generates a perfect circle of the radius and center determined, so as long as enough of the circle's perimeter is there to vote for the correct center, it can identify the circle even if the circle boundary itself is broken or noisy. This is particularly useful for noise images or circles that are partially occluded. However, it also means that if the circle is actually an ellipse, it will get a lot of votes for the center of the ellipse, but will likely draw circles with different radii. This is demonstrated below.



The final point I would like to talk about is that I had to choose my parameters according to the result I wanted. I needed to treat noisy images differently. I also had to use different thresholds for images with different numbers of circles. In the first image, there were only 3 circles, so I took the top 15% of detected circles, whereas in the circle image above and to the left, there are many circles, so I took the top 30% of detected circles. As a separate note, I really enjoyed this project. I implemented my own canny edge detector and learned a lot!

Problem 3 (60 pts): *See https://github.com/jshi31/csc249tracking*

Problem 4 (20 pts, mandatory for 449, extra points for 229): *Least square linear fitting is one of the most fundamental fitting models. In class, we have learned 2D line fitting. Now let's extend it to 1) high dimensional points; 2) fitting with L2 weight penalty, which is also known as ridge regression. Now we have*

-Data: $(x_n, y_n),\ n = 1, 2, ..., N,\ x_n \in \mathbb{N}^d,\ y_n \in \mathbb{N}^k$

-Model: $y = Wx$ *where* $W \in \mathbb{N}^{kxd}$ *is the weight of the linear model*

-Objective: find optimal W to minimize $E = ||Y - WX||^2 + \lambda||W||^2$ *where* $||\cdot||$ *is Frobenius Norm which means* $||X|| = \sqrt{Tr(X^T X)}$.

Try to prove that the optimal W is $W = YX^T(XX^T + \lambda I)^{-1}$

For this, I had to look up some of the rules of Traces and Matrix Calculus. Honestly, it would be nice to include some background on Matrix Calculus in the slides or provide resources when explaining the 2D case we learned in class. I've listed below the rules that I used completing this problem. Sources include https://en.wikipedia.org/wiki/Trace_(linear_algebra) https://www.ics.uci.edu/~welling/teaching/KernelsICS273B/MatrixCookBook.pdf

$$||A||^2 = Tr(A^T A)$$
$$Tr(A + B) = Tr(A) + Tr(B)$$
$$Tr(cA) = cTr(A)$$
$$Tr(AB) = Tr(BA)$$
$$Tr(A^T B) = Tr(AB^T) = Tr(B^T A) = Tr(BA^T)$$

$$(AB)^T = B^T A^T$$

$$_aTr(AB) = B^T$$
$$\frac{d}{dX}Tr(AXB) = B^T A^T$$
$$\frac{d}{dX}Tr(X^T X) = 2X$$
$$\frac{d}{dX}Tr(B^T X^T IXB) = I^T XBB^T + IXBB^T$$

Here is my solution using these rules.
I start by rewriting *E* in terms of traces which I can then take derivatives of.

$$E = Tr((Y - WX)^T(Y - WX)) + \lambda Tr(W^T W)$$
$$E = Tr(Y^T Y - Y^T WX - (WX)^T Y + (WX)^T WX) + \lambda Tr(W^T W)$$

Each of the matrices inside the first trace are NxN square matrices

$$E = Tr(Y^T Y) - Tr(Y^T WX) - Tr((WX)^T Y) + Tr((WX)^T WX) + \lambda Tr(W^T W)$$
$$E = Tr(Y^T Y) - 2 * Tr(Y^T WX) + Tr((WX)^T WX) + \lambda Tr(W^T W)$$
$$E = Tr(Y^T Y) - 2 * Tr(Y^T WX) + Tr(X^T W^T WX) + \lambda Tr(W^T W)$$

We now want to find the *W* that minimizes *E*, so we take the derivative of *E* with respect to *W* and set it to 0 and then solve for *W*.

$$
\begin{aligned}
E &= Tr(Y^T Y) - 2 * Tr(Y^T WX) + Tr(X^T W^T WX) + \lambda Tr(W^T W) \\
0 &= \frac{dE}{dW} = 0 - 2YX^T + 2WXX^T + \lambda 2W \\
0 &= 2(-YX^T + WXX^T + \lambda W) \\
0 &= -YX^T + WXX^T + \lambda W \\
YX^T &= WXX^T + \lambda W \\
YX^T &= W(XX^T + \lambda I) \\
W &= YX^T (XX^T + \lambda I)^{-1}
\end{aligned}
$$

Thus, the *W* that optimizes *E* is $W = YX^T (XX^T + \lambda I)^{-1}$