PROBLEM 1:

A close up of text on a white background

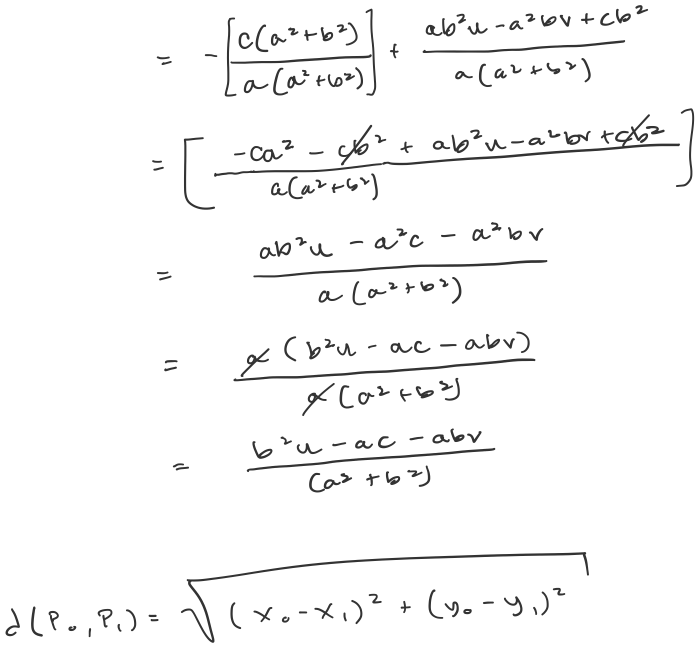
Description automatically generated

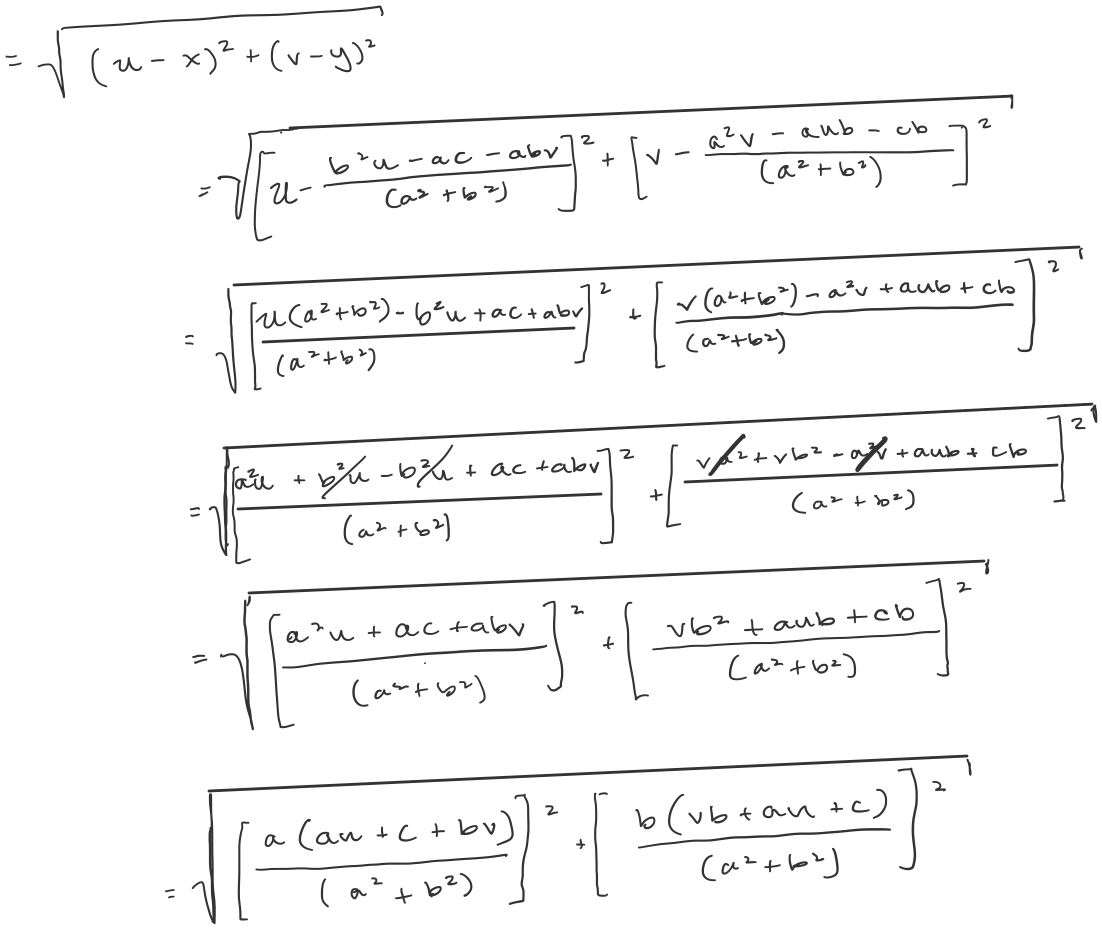
A close up of a map

Description automatically generated  
A close up of text on a white background

Description automatically generated

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | A close up of a logo  Description automatically generated |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |  |





A close up of text on a black background

Description automatically generated

A screenshot of a cell phone

Description automatically generated

PROBLEM 2:

1. I chose circle
2. Done
3. I was able to get it to work, albeit slowly. Please be patient with the code, it does finish and it does give a somewhat accurate Hough transform.

Some of the parameters I implemented in my code that can tweak the accuracy depending on the nature of the input image are:

* 1. I used CV Canny edge detector to detect edge points, after a median blur to circumvent noisy non-circle data points. I default the median blur to 7 for the provided shoe image, but this can be tweaked in the main function.
  2. The size of the bins, aka the granularity, can be accessed in the main function
  3. I used a 3D matrix to store the accumulator votes, (r, a, b) = vote\_count
  4. The percentile cutoff threshold below which any detected local maxima in the accumulator matrix are disposed of is default = 90, but can be tweaked in “findMax()” function)
  5. Size of the max pooling window:

I implemented a window size to search the accumulator matrix and find local maxima which defaults to a neighborhood of 10. The smaller the neighborhood, the finer the granularity of resulting maxima

* 1. Because there are so many a and b calculations, many of them are within proximity of one another and result in a similar number of votes and corresponding indices. Thus, I implemented a loop in “findMax()” that removes some similar indices from the list of accumulator matrix indices. This defaults to disposing of indices with a difference of 25 or less. The higher this value, the less accurate the resulting transform will be, but consequently will also have fewer overall circles.

The primary difference between my implementation and cv2 is that I use the 3-parameter representation, while cv2 uses the Hough gradient method. I explicitly calculate all possible (a, b) values for each r, each (x, y), and each possible theta (in radians). As a consequence, the cv2 implementation is far, far more time efficient. However, a tradeoff of cv2 Hough Circle is that the user must decide between a higher quantity of circles being detected and more accurate circles being detected. When provided images with more sporadic circle shapes, the cv2 method appeared to either find more circular edges, but when detected, were often less off center, or less circular edges, but when detected were more in line with the actual shape of the curve.

PROBLEM 4:

Tuesday, March 17, 2020

7:01 PM

A close up of text on a white background

Description automatically generated

