Detecting Lines and Circles in Images

Computer Vision (CS342) Mini-Project 2

Bhattacharya Brothers
Soham Bhattacharya, Darpan Bhattacharya

February 24, 2025

1 Introduction

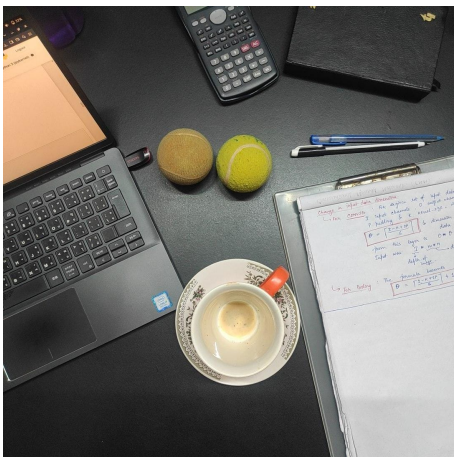
In this mini-project, our goal was to design and implement two methods, `hough_lines_vote_acc`, where we were required to accumulate votes for lines in the hough space, and `hough_circles_vote_acc`, where we were required to accumulate votes for circles in the hough space.

This report is a summary of the work that was done for this mini-project and the results that were obtained.

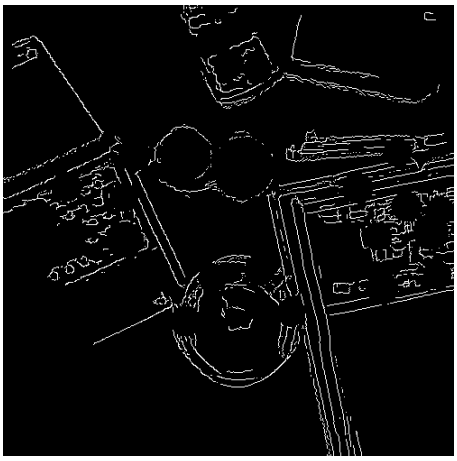
We used the `preprocess` method with the following parameters:

- `image` (=img): Our original image.
- `erodeK` (=3): We used a kernel size of 3 for eroding our image as our image was relatively sharp.
- `blurK` (=11): We used a relatively large size of the gaussian blur kernel because of the same reason as above and because our image had almost no noise.
- `blurSigma` (=3): We used a relatively small value of the standard deviation since we did not have much noise in our original image
- `lowT` and `highT` (=20, 40): Most edges in our image were the edges of the objects from the background, which were pretty moderate, given the background color and almost similar object colors (the calculator, diary, and laptop). Thus we used relatively moderate thresholds for canny edge detector in our image, since it did not have any too sharp or too weak edges.

The following original image was used (this is an image of Soham's study table) for the purpose of this mini-project.



The following image is the "edge image" of the original image, which was used as an input for both the above mentioned functions.



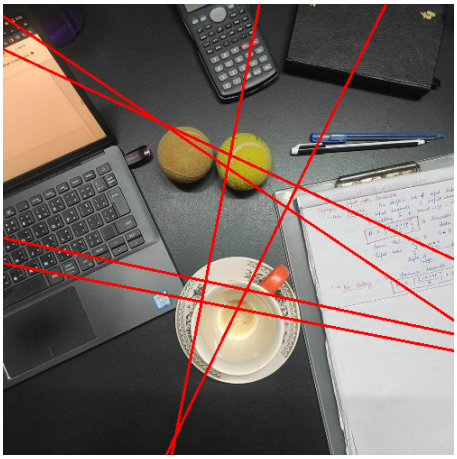
---

2 Hough Lines Vote Accumulator

In the `hough_lines_vote_acc` function, we:

- Received an input image that was binary with the white pixels being the edges in the original image.
- Found out the length of the diagonal of the image from its shape.
- Iterated over all pixels in the image, and for each edge pixel,
  - We enumerated over all values of θ from 0 to 179
  - For each value of θ we converted it into radians, and from there we calculated $\rho = x\cos\theta + y\sin\theta$, and found idx_ρ, the index value of the closest integral ρ value to our originally computed ρ.
  - After we found the value of idx_ρ, we incremented the value of the $\{idx_\rho, \theta\}$-th index in our hough votes accumulator.

The following is the output image obtained after executing the `hough_lines_vote_acc` function.



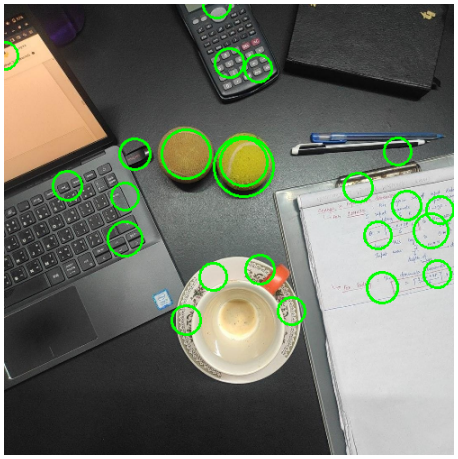
---

3 Hough Circles Vote Accumulator

In the `hough_circles_vote_acc` function, we:

- Received an input image that was binary with the white pixels being the edges in the original image, and the fixed radius r of the circles.
- Iterated over all pixels in the image, and for each edge pixel,
  - We enumerated over all values of θ from 0 to 359
  - For each value of θ we converted it into radians, and then we calculated x_c and y_c, the coordinates of the center of the circle with the current edge pixel in its perimeter in the hough space, using the formulas $x_c = x - r\cos\theta$ and $y_c = y - r\sin\theta$ respectively.
  - After we found the values of x_c and y_c, if the point was within the bounds of the image in the hough space, we incremented the values of the $\{x_c, y_c\}$-th index in our hough votes accumulator.

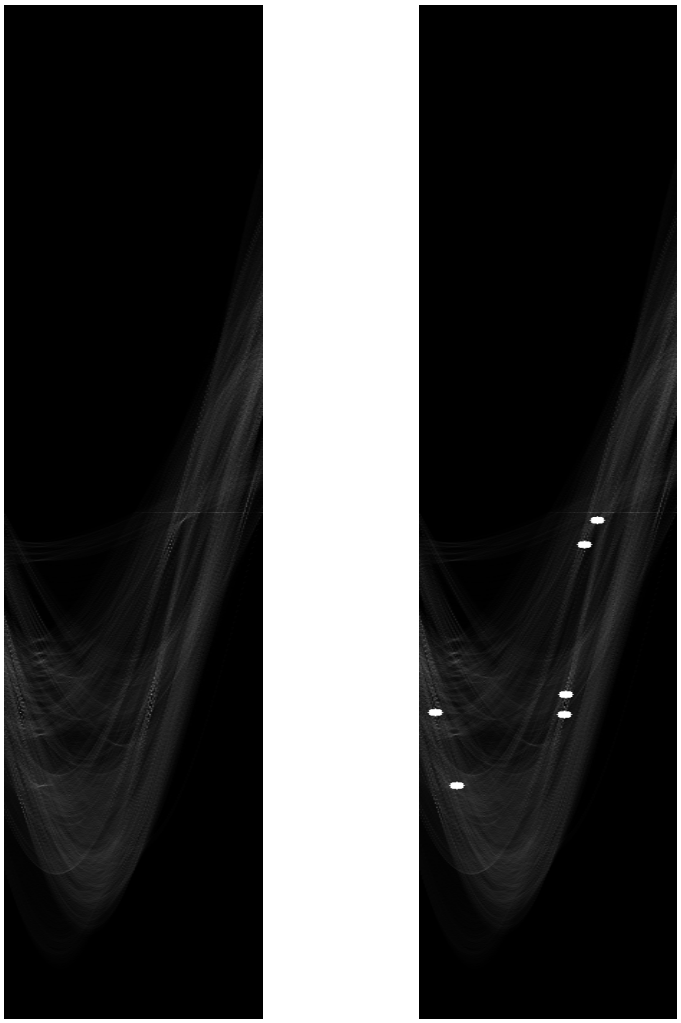
The following is the output image obtained after executing the `hough_circles_vote_acc` function.



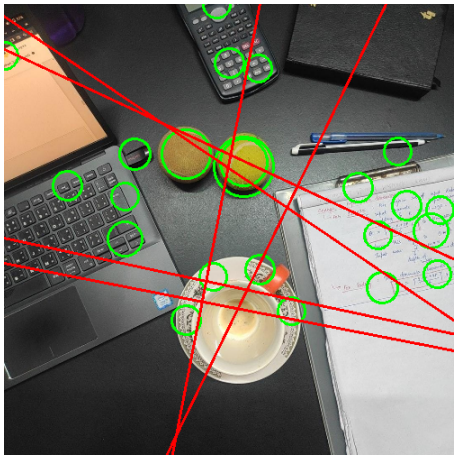
---

4 Some other relevant images

The following images are the images of the hough space and the hough space with peaks highlighted, after executing the `hough_lines_vote_acc` function.



The following is the combined image of both the detected circles and lines.



5 Conclusion

We thank our course instructor, Br. Tamal mj for assigning this project.
Doing this mini-project helped us to gain insights on the hough transform algorithm. Tweaking the parameters helped us to better understand how each parameter affected the output images.
