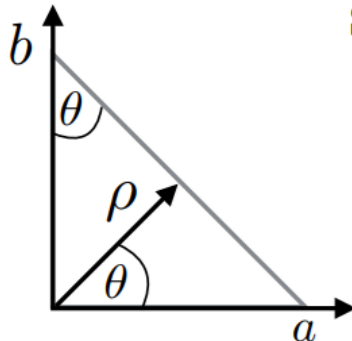


2.1

For any given point (x,y) there are infinite lines that pass through it, each line having a unique ρ and θ . This means each point corresponded to a continuous range of θ (0-360) and associated ρ

Derivation:



$$\cos \theta = \frac{\rho}{a} \rightarrow a = \frac{\rho}{\cos \theta}$$

$$\sin \theta = \frac{\rho}{b} \rightarrow b = \frac{\rho}{\sin \theta}$$

plug into double intercept form: $\frac{x}{a} + \frac{y}{b} = 1$

$$x \cos \theta + y \sin \theta = \rho$$

From the derivation above, one can see that the relationship between ρ and θ is sinusoidal

$$\rho = b \sin \theta$$

Where b is the y intercept. This leads to a sinusoidal curve in parameter space, with each point of the curve corresponding to one line.

Here, $\theta = \tan^{-1} \frac{y}{x}$ and $\rho = \sqrt{x^2 + y^2}$.

2.2

When we parametrize in terms of slope and intercept, the Hough space becomes infinite, as a line of the form $x=a$ has $m=\infty$. This mean that we will not be able to model vertical lines.

Recalling the equation,

$$x \cos \theta + y \sin \theta = \rho$$

Rearranging,

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

Comparing with $y = mx + c$,

$$m = -\cot \theta$$

and,

$$c = \rho / \sin \theta$$

2.3

ρ has maximum absolute value of $\text{ceil}(\sqrt{x^2 + y^2})$

θ has a range of +90 degrees to -180 degrees, considering the origin at top left(1,1 pixel), if negative values of ρ are not allowed. Else, range is -90 to +90 as shown below.

2.4

For this line, $m = -1$ and $c = 0$.

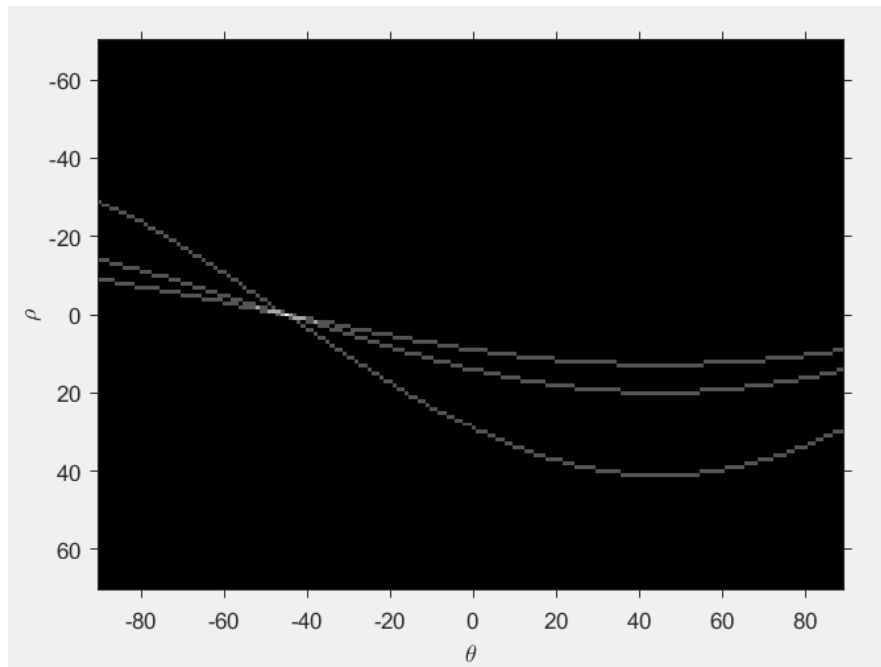


Fig: Sinusoid waves in Hough space

Intersection point is the line $\rho = 0$ and $\theta = -45^\circ$

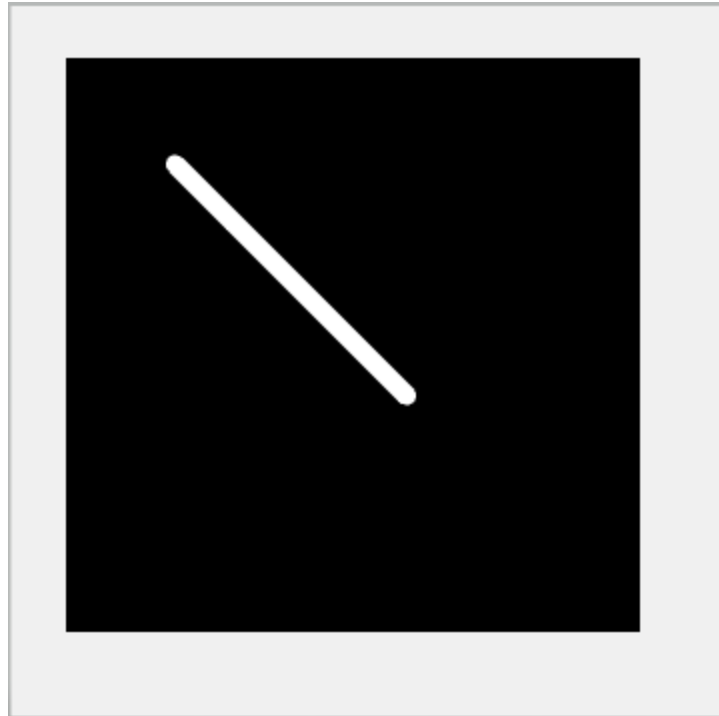


Fig: The detected line visualized

2.5

If the feature space is too large, it takes a long time to compute. Also, the maxima will be distributed over the set of neighbouring bins and may not be detected. Noise in data will also be highly pronounced in high parameter space. On the contrary, if the bins are too large, the points may not be able to place a vote because there is no right line, or too many points vote for the same bucket. In this case, lines detected may will be very approximated as rho and theta are sparse.

Is the parameter space is high, I would condense it but by conserving local information. I would map the high parameter hough space to a lower parameter hough space where each element is a moving average/inverse gaussian blur of corresponding local areas in high parameter space. The problem is that votes get distributed and local maxima may go unnoticed. This can be solved by the above fix.

3.1

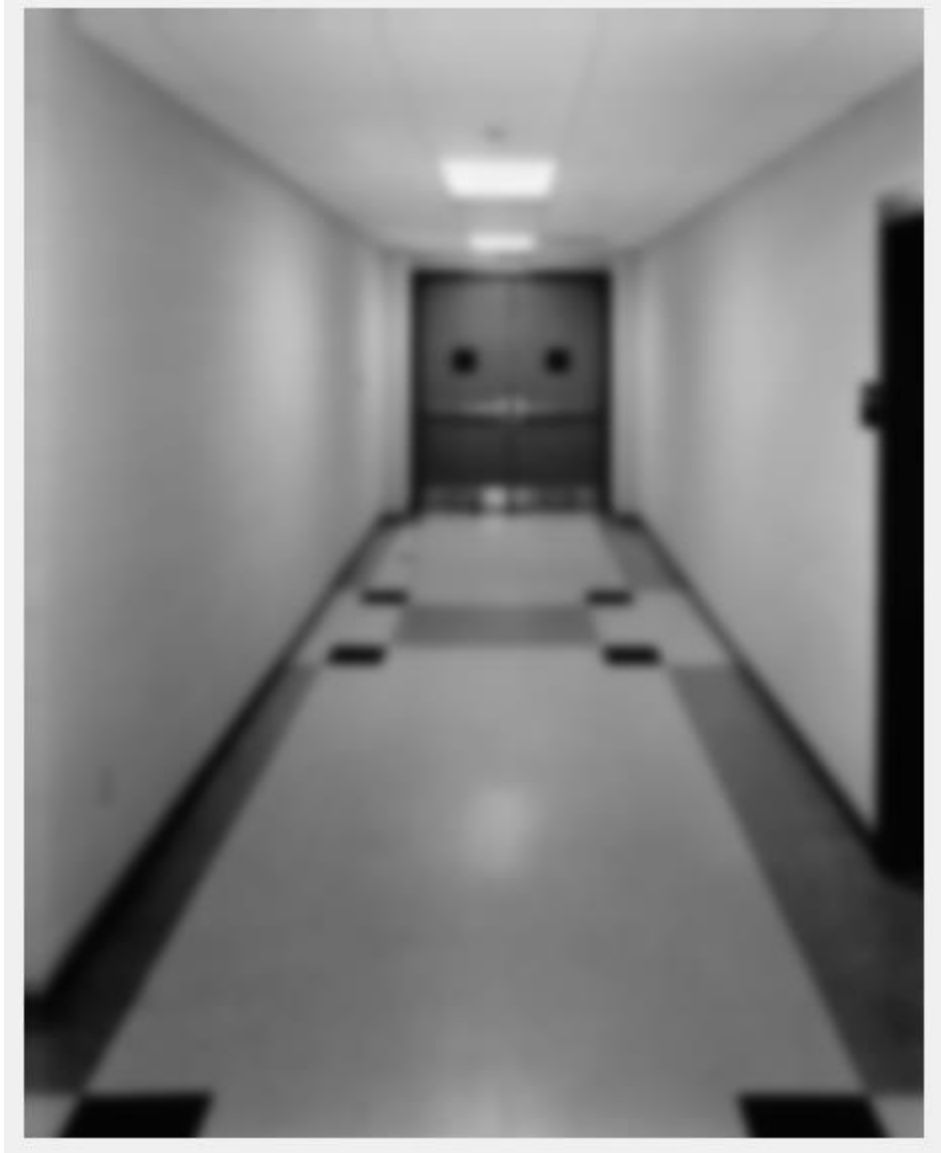


Fig: After convolution

3.2

This is included in kgopalak/ec/

3.3

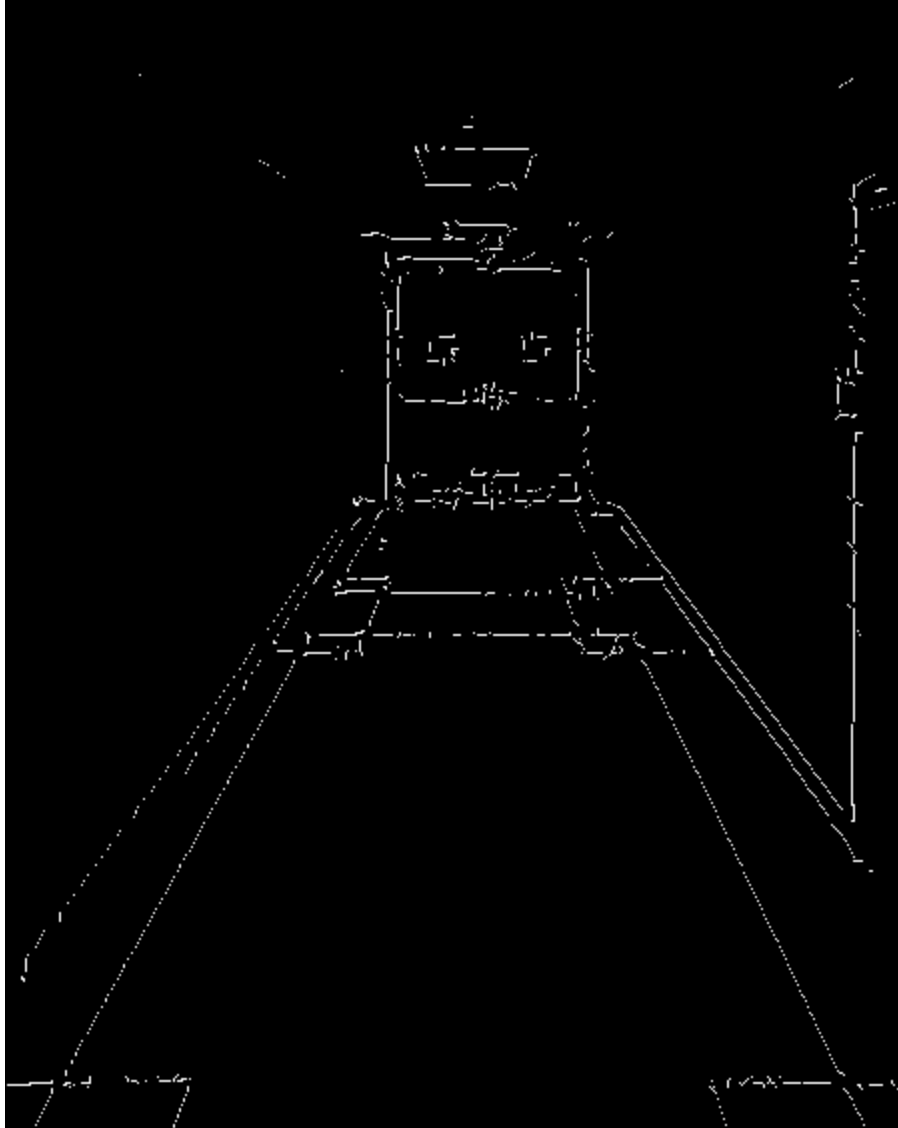


Fig: Edge intensity Image

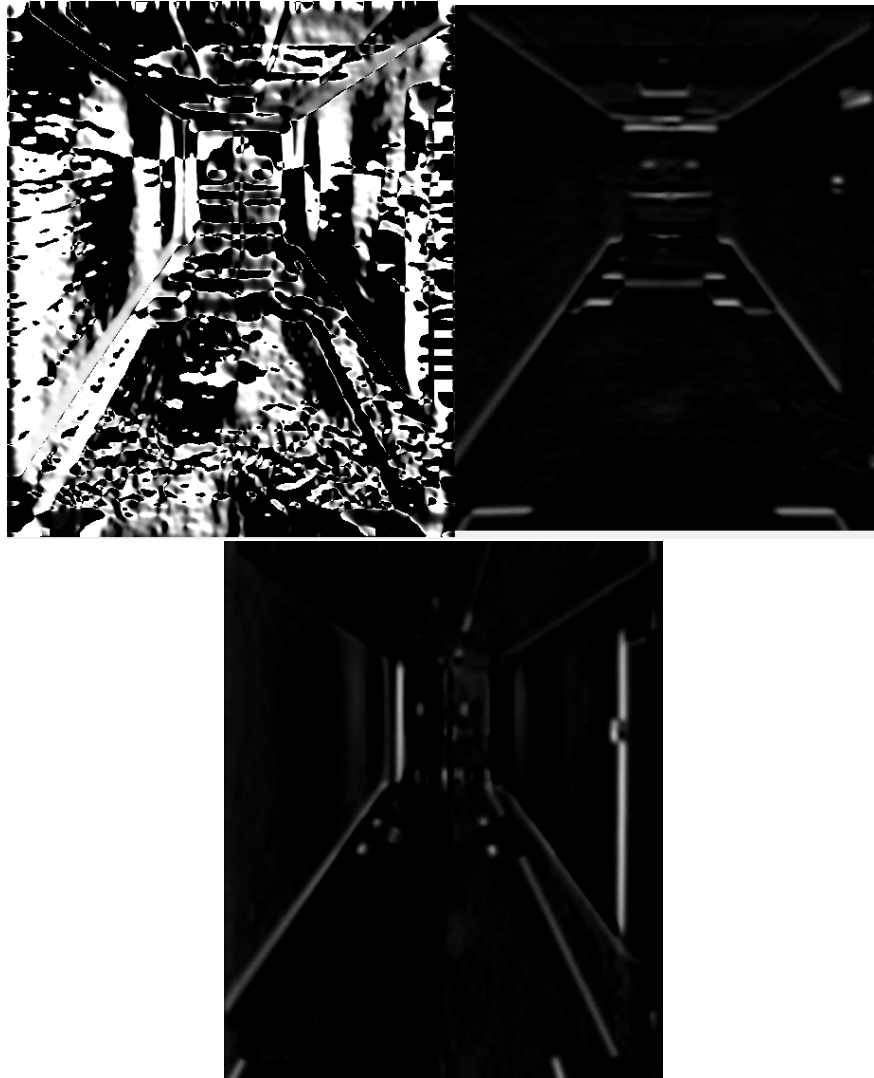


Fig: Edge orientation, I_x and I_y images

3.4



Fig: Hough Space.
Rho on y axis and theta on x axis

3.5 & 3.6



Fig: Edge detection visualized

4. Experiments

Did your code work well on all the image with a single set of parameters? How did the optimal set of parameters vary with images?

No, my code did not work well on all images with a single set of parameters. The optimal set of parameters depended upon the light in the image and the contrast at edges. For images with a lot of light/whiter pixels, increasing the threshold meant less noisy edge detection, however in case of darker images this wiped out good edge detections because less points could now vote. The fineness of optimal parameter space also varied from image to image.

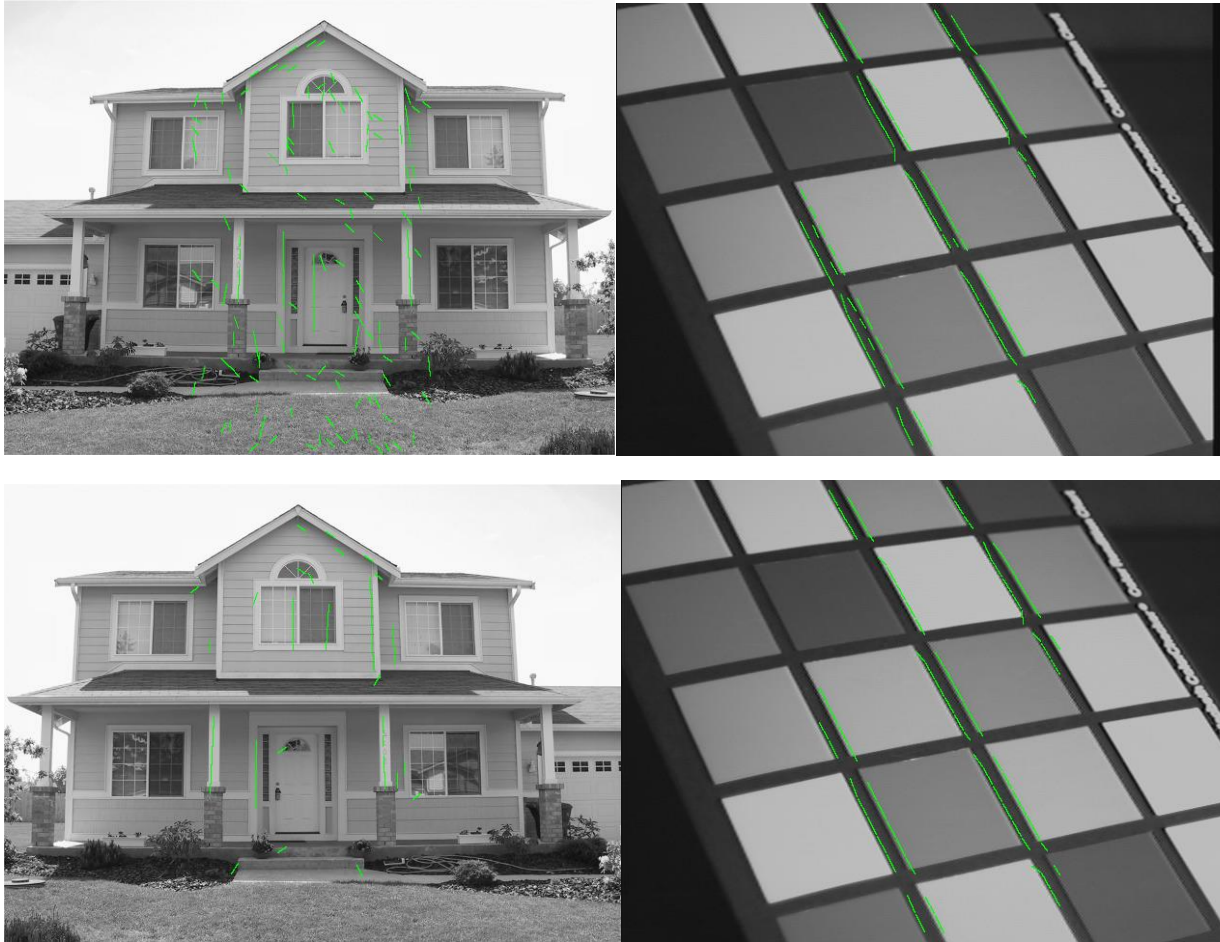


Fig: Performance on changing threshold(first row threshold=0.1, second row threshold = 0.2)

As can be seen from figure above, lowering threshold(first row) leads to lots of noisy detection in the house image above which is brighter, however, performance is somewhat unchanged for the second image.



Fig: Effect of coarse(above) and fine parameter space for two sets of images.

As in figure one can see that the optimal set of parameters is different for different images. While performance generally improves up to a point as one makes the parameter space coarse, the rate of improvement is widely different. Image on left greatly benefits from a coarse parameter space while image on right only moderately improves.

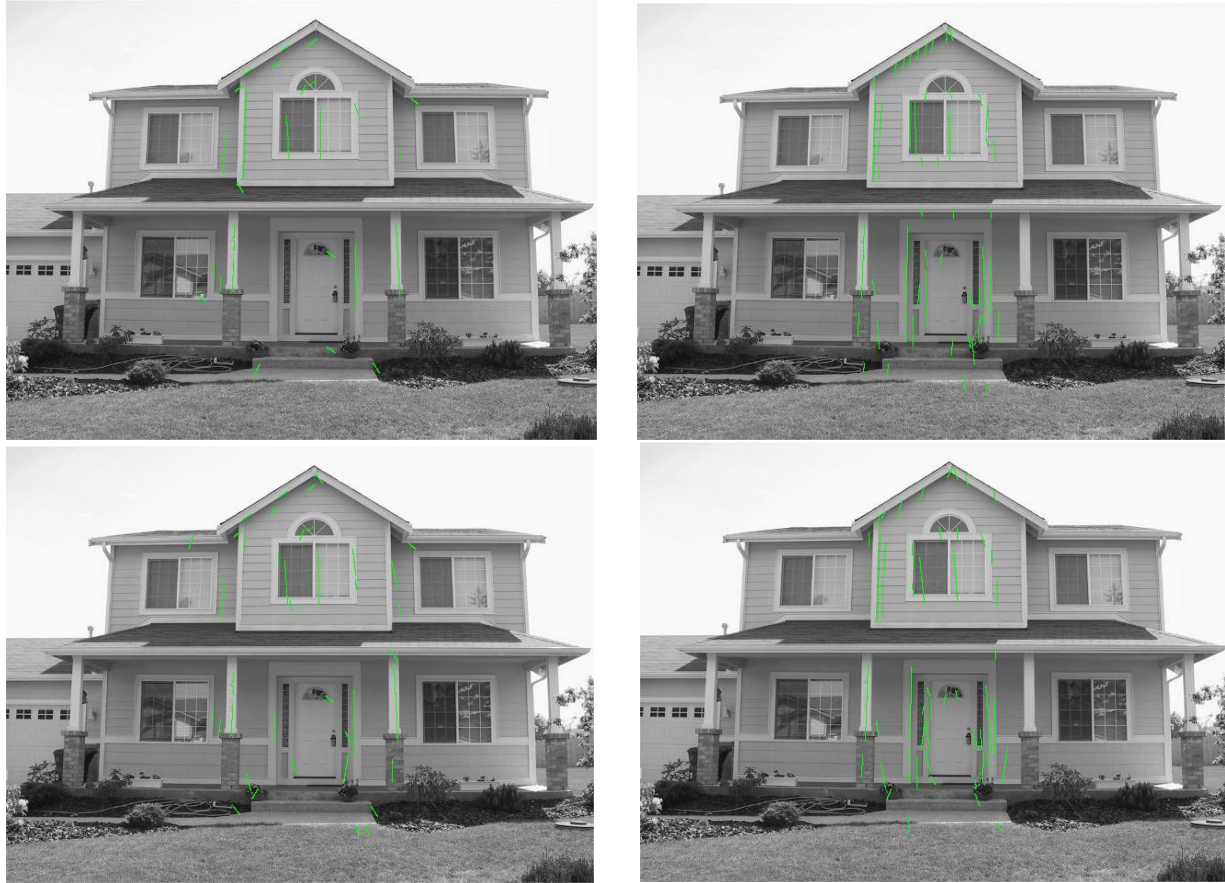


Fig: From clockwise: Performance of same image as parameter space grows coarser

As one makes the parameter space coarser, one hits an optimal point when performance peaks and it falls farther on as ρ_{Res} and θ_{Res} are increased.

Which step of the algorithm causes the most problems?

The Hough lines part of the code gave the most problems. Initially, as parameter space was very fine, hardly any lines were getting detected. The degree of flexibility with respect to ρ_{Res} , θ_{Res} and tolerance made the Hough Transform part tricky.

Did you find any changes you could make to your code or algorithm that improved performance?

Yes. Vectorisation at all stages greatly improved speed. Float comparison while transforming to Hough space works well with a degree of tolerance.