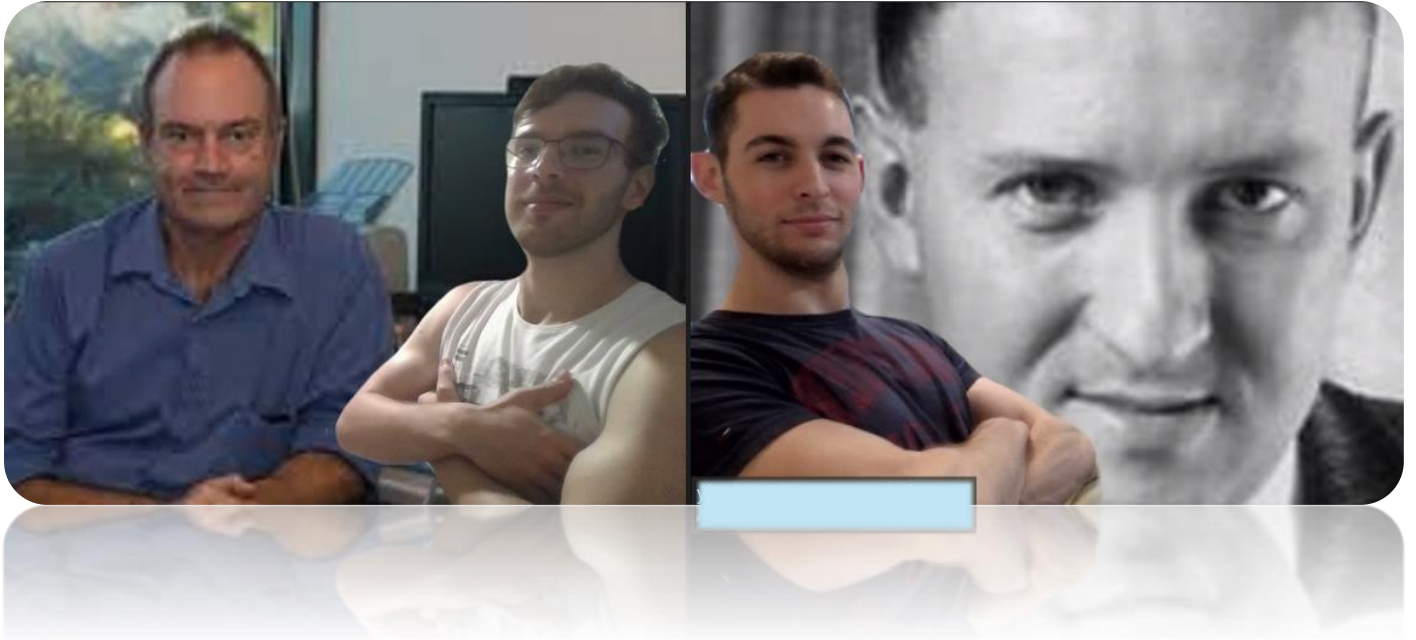


Hough Transform Research report



Authors:

Yitzchak Grunbaum 318837317

Or Bauberg 207477738

Introduction

The Hough transform uniquely enables the dual representation of lines and points within two spaces: the image space and the parameter space.

This report covers the fundamental concept of this duality, as discussed in class. We show our implementation of the generic accumulator matrix method introduced by Hough in 1962. We discuss the correctness of the algorithm and present our two claims, that we can find a granularity of the discretization to satisfy completeness and soundness which we define later. We then go on to talk about the transition from a standard image to a set of edge points on which we can use the previous algorithm presented. This transition is being made by using the algorithm of John F. Canny, which we also implemented.

Part 1: Introduction to Hough transform

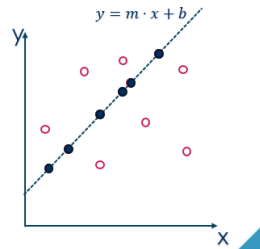
Paul Hough was a British mathematician and computer scientist who is best known for inventing the Hough transform, an important technique in image analysis and computer vision.

The Hough transform was first introduced by Paul Hough in 1959 while he was working at the U.S. Atomic Energy Commission. It was originally patented as a method for detecting complex patterns of points in binary image data.

Given: set of points $\{(x_i, y_i)\}_{i=1}^n$

Task: Detect line:

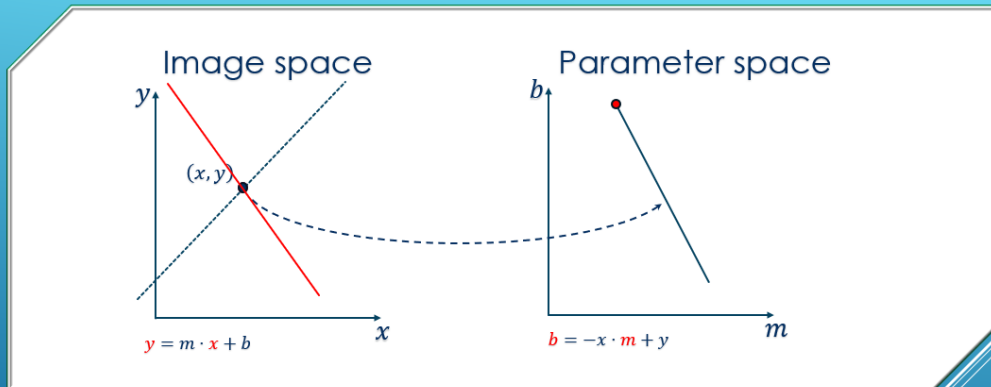
$$y = m \cdot x + b$$



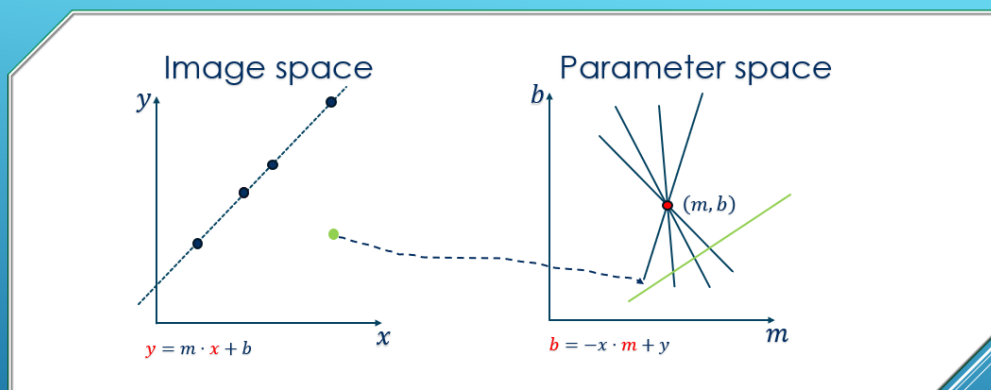
STARTING OFF SIMPLE...



ENTER HOUGH



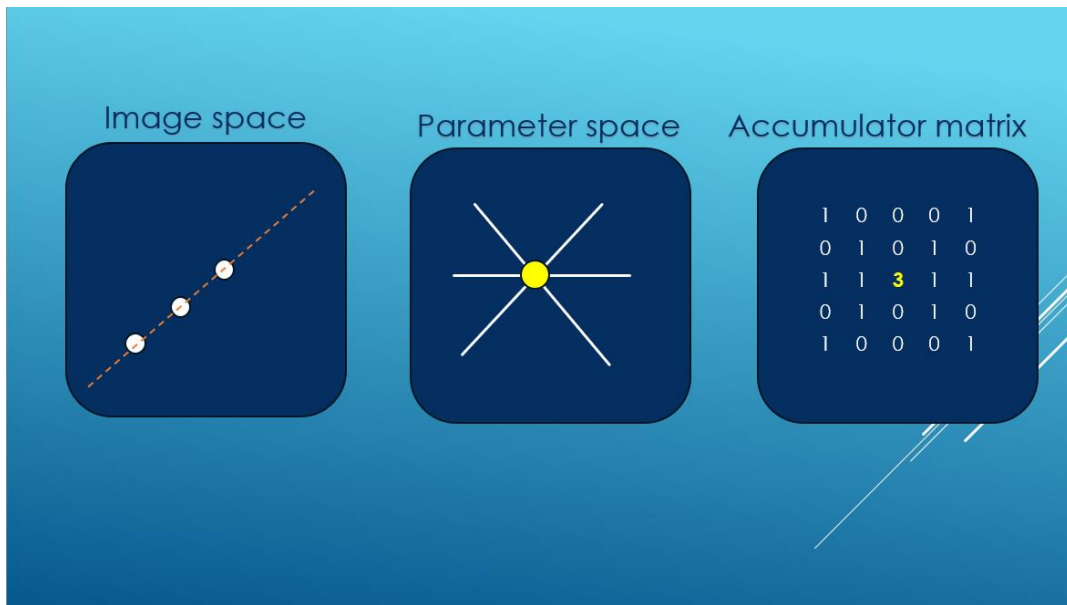
REMINDER: THE DUALITY



As thought in class, looking at the parameter space we can see that a line that passes through some points will be manifested as an intersection point between the lines corresponding to these points in the parameter space.

1. Discretize parameter space
2. Create accumulator matrix A
3. For each point (x_i, y_i) :
 For all entries (m, b) in A :
 If (m, b) satisfies: $b = -x_i \cdot m + y_i$
 $A(m, b) += 1$
4. Locate local maxima in A

ALGORITHM



An illustration of the way that the matrix should be filled.

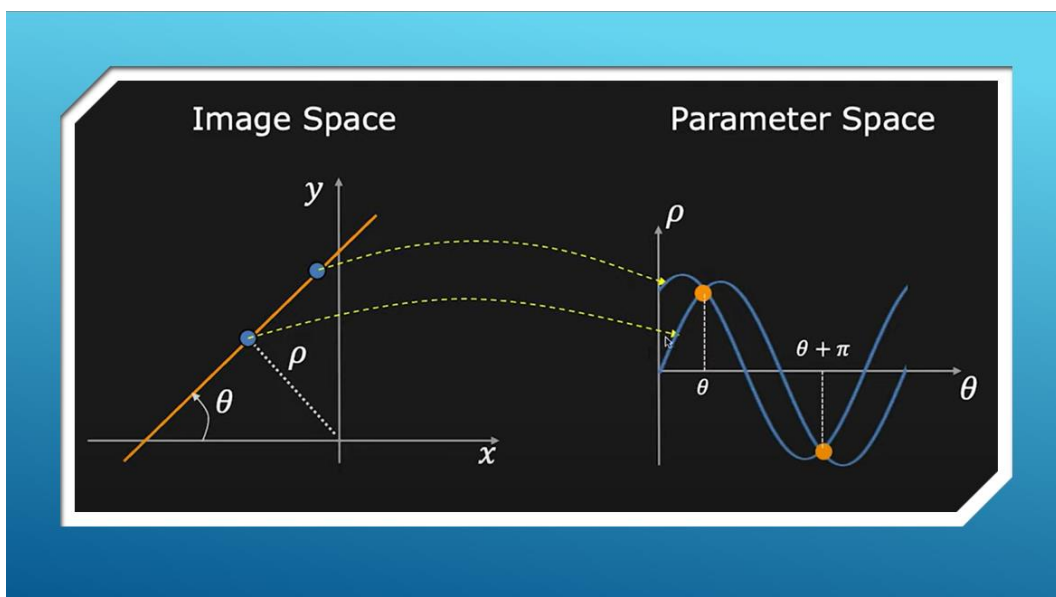
Issue: Slope of the line $-\infty \leq m \leq \infty$

- Large Accumulator
- More Memory and Computation

Solution: Use $x \sin \theta - y \cos \theta + \rho = 0$

- Orientation θ is finite: $0 \leq \theta < \pi$
- Distance ρ is finite

A problem arises when devising an implementation for this idea, since the parameter m for the slope is not bounded, making it impossible to be represented fully in a discrete finite matrix.

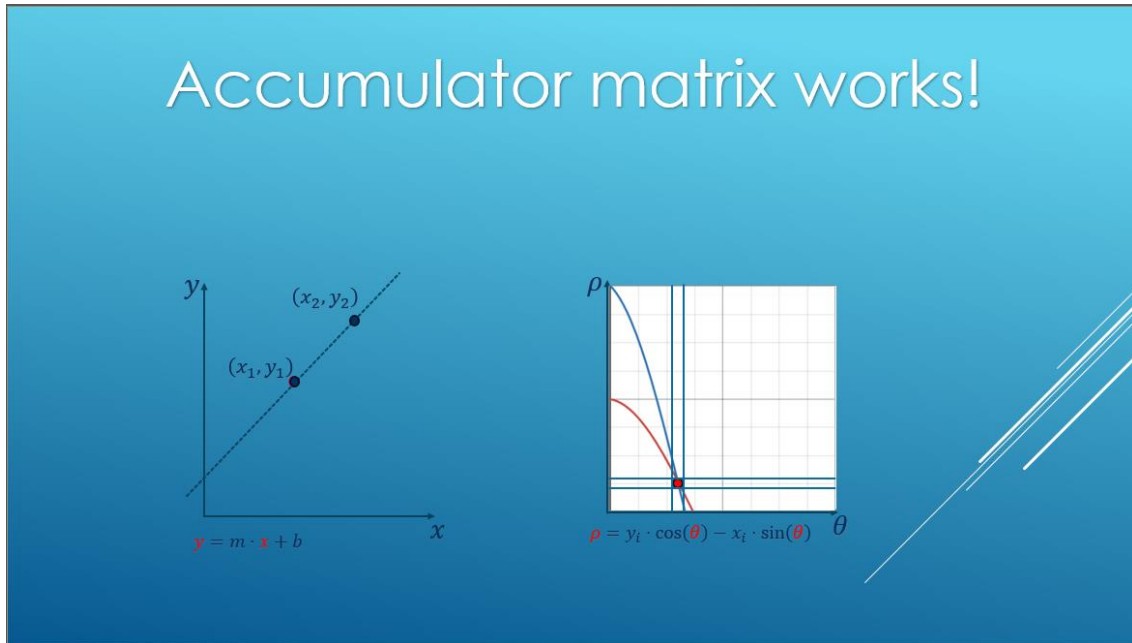


The representation with ρ, θ as the offset from $(0,0)$ and the angle between the line and x' 's axis represents a unique line now with bounded parameters. With $\theta \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ and $\rho \in [-diag, diag]$ where $diag = \left\lceil \sqrt{(x_{max} - x_{min})^2 + (y_{max} - y_{min})^2} \right\rceil$.

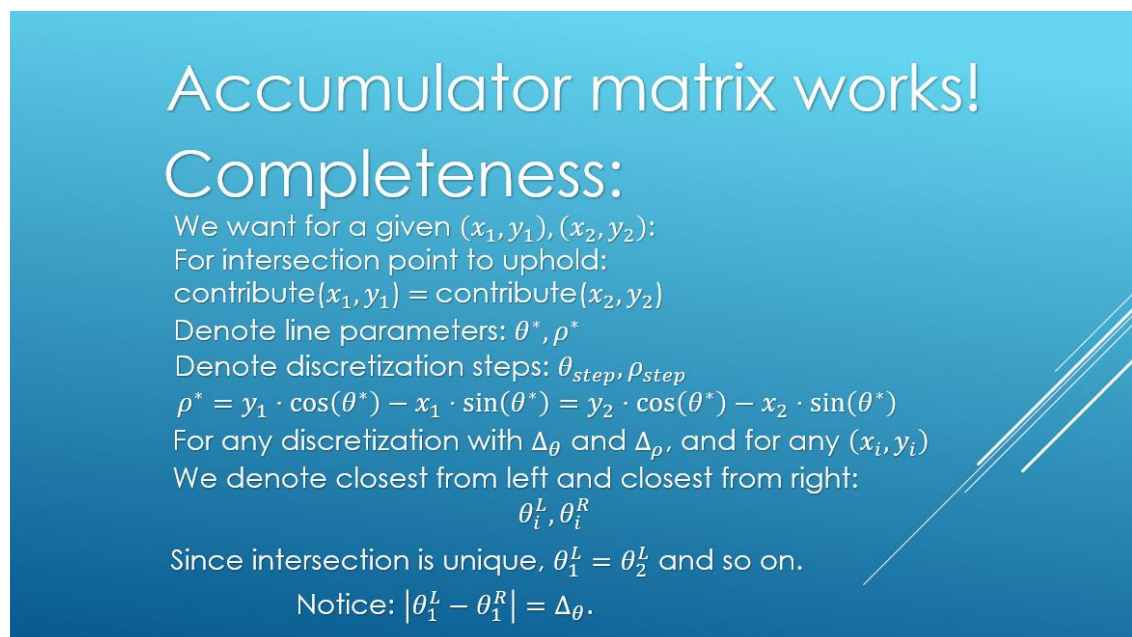
Part 2: completeness and soundness

We discretize our space, what guarantees that we catch every intersection? Secondly What guarantees that irrelevant points do not vote in cells of the accumulator inadvertently?

Completeness:



We want to show that we can make a requirement that will necessarily make the intersection point be caught, meaning that when different points on the same line will vote in matrix cells, the ρ values that will be chosen won't be too far apart and scattered, rather they will be closed in two adjacent cells in the matrix.



Accumulator matrix works!

Completeness

in the algorithm, in iteration of θ_1^L :

For point 1:

$$\rho_1 = y_1 \cdot \cos(\theta_1^L) - x_1 \cdot \sin(\theta_1^L)$$

For point 2:

$$\rho_2 = y_2 \cdot \cos(\theta_1^L) - x_2 \cdot \sin(\theta_1^L)$$

And each will vote in the closest bin of the accumulator

We want ρ_1 and ρ_2 to be rounded to the same bin

Denote $\rho_{1_{idx}}, \rho_{2_{idx}}$ as the indices that will be chosen for θ_i^L .

$|\rho_{1_{idx}} - \rho_{2_{idx}}| < 1 \Rightarrow \rho_1$ and ρ_2 are being rounded to the same bin.

Accumulator matrix works!

Completeness

$|\rho_{1_{idx}} - \rho_{2_{idx}}| < 1 \Rightarrow \rho_1$ and ρ_2 are being rounded to the same or adjacent bin.

That is what we want to achieve!

Accumulator matrix works!

Completeness

$$\begin{aligned}\rho_1 &= y_1 \cdot \cos(\theta_1^L) - x_1 \cdot \sin(\theta_1^L) \\ \rho_2 &= y_2 \cdot \cos(\theta_1^L) - x_2 \cdot \sin(\theta_1^L)\end{aligned}$$

If $|\theta^* - \theta_i^L|$ is small,

Using a linear approximation:

$$\cos(\theta_1^L) \approx \cos(\theta^*) - \sin(\theta^*) \cdot (\theta_i^L - \theta^*)$$

$$\sin(\theta_1^L) \approx \sin(\theta^*) + \cos(\theta^*) \cdot (\theta_i^L - \theta^*)$$

$$\rho_1 = y_1 \cdot (\cos(\theta^*) - \sin(\theta^*) \cdot (\theta_i^L - \theta^*)) - x_1 \cdot (\sin(\theta^*) + \cos(\theta^*) \cdot (\theta_i^L - \theta^*))$$

Accumulator matrix works!

Completeness

$$\rho_1 = y_1 \cdot (\cos(\theta^*) - \sin(\theta^*) \cdot (\theta_i^L - \theta^*)) - x_1 \cdot (\sin(\theta^*) + \cos(\theta^*) \cdot (\theta_i^L - \theta^*))$$

$$\begin{aligned} \rho_1 &= y_1 \cdot \cos(\theta^*) - x_1 \cdot \sin(\theta^*) - y_1 \cdot \sin(\theta^*) \cdot (\theta_i^L - \theta^*) - x_1 \cdot \cos(\theta^*) \cdot (\theta_i^L - \theta^*) \\ &= \rho^* + (\theta^* - \theta_i^L) \cdot (x_1 \cdot \cos(\theta^*) + y_1 \cdot \sin(\theta^*)) \end{aligned}$$

$$\rho_1 - \rho^* = (\theta^* - \theta_i^L) \cdot (x_1 \cdot \cos(\theta^*) + y_1 \cdot \sin(\theta^*))$$

$$\begin{aligned} & (x_1 \cdot \cos(\theta^*) + y_1 \cdot \sin(\theta^*)) \\ \frac{d}{d\theta^*} (x_1 \cdot \cos(\theta^*) + y_1 \cdot \sin(\theta^*)) &= -x_1 \cdot \sin(\theta^*) + y_1 \cdot \cos(\theta^*) = 0 \\ x_1 \cdot \sin(\theta^*) &= y_1 \cdot \cos(\theta^*) \\ \frac{y_1}{x_1} &= \tan(\theta^*) \\ \theta_{max}^1 &= \arctan\left(\frac{y_1}{x_1}\right) \end{aligned}$$

Accumulator matrix works!

Completeness

So we can say:

$$\begin{aligned} |\rho_1 - \rho^*| &= |(\theta^* - \theta_i^L) \cdot (x_1 \cdot \cos(\theta^*) + y_1 \cdot \sin(\theta^*))| \\ &= \Delta_\theta \cdot |x_1 \cdot \cos(\theta^*) + y_1 \cdot \sin(\theta^*)| \leq \Delta_\theta \cdot \rho_{max} \end{aligned}$$

Denote $\rho_{max} = \max_{i \in [n]} \left\{ x_i \cdot \cos\left(\arctan\left(\frac{y_i}{x_i}\right)\right) + y_i \cdot \sin\left(\arctan\left(\frac{y_i}{x_i}\right)\right) \right\}$

Hence:

$$|\rho_1 - \rho_2| \leq |\rho_1 - \rho^*| + |\rho_2 - \rho^*| \leq 2 \cdot \Delta_\theta \cdot \rho_{max}$$

Accumulator matrix works!

Completeness

$$|\rho_1 - \rho_2| \leq |\rho_1 - \rho^*| + |\rho_2 - \rho^*| \leq \Delta_\theta \cdot \rho_{max}$$

Using this result we can say:
Rounding is executed like this:

$$\left\lfloor \frac{\rho_i + \#_\rho}{\Delta_\rho} \right\rfloor$$

So if we require $\left| \frac{\rho_1 + \#_\rho}{\Delta_\rho} - \frac{\rho_2 + \#_\rho}{\Delta_\rho} \right| < 1$

Completeness follows!

Accumulator matrix works!

Completeness

$$|\rho_1 - \rho_2| \leq |\rho_1 - \rho^*| + |\rho_2 - \rho^*| \leq 2 \cdot \Delta_\theta \cdot \rho_{max}$$

$$\left| \frac{\rho_1 + \#_\rho}{\Delta_\rho} - \frac{\rho_2 + \#_\rho}{\Delta_\rho} \right| < 1$$

$$\frac{1}{\Delta_\rho} |\rho_1 - \rho_2| \leq \frac{1}{\Delta_\rho} 2 \cdot \Delta_\theta \cdot \rho_{max}$$

$$\frac{1}{\Delta_\rho} 2 \cdot \Delta_\theta \cdot \rho_{max} < 1$$

We completed Completeness!

$$\Delta_\theta < \frac{1}{2} \cdot \frac{\Delta_\rho}{\rho_{max}}$$

What we showed is that given a step in ρ : Δ_ρ , satisfying $\Delta_\theta < \frac{1}{2} \cdot \frac{\Delta_\rho}{\rho_{max}}$, for any two points that are on the same line, for the closest θ in the discretization to the real θ^* of the intersection point in the parameter space, for both points the ρ index that will be chosen in the algorithm will be at most 1 cell apart in the accumulator matrix. Satisfying this for any two points on a line necessitates that for any k points in the same line the claim holds for any pair, meaning that for all k points, there are at most 2 adjacent cells for ρ that they vote to when the algorithm select θ_L or θ_R (symmetric cases).

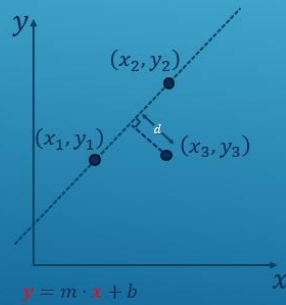
That way guarantees that the intersection point will indeed be 'caught' by a cell or two in the matrix.

Soundness:

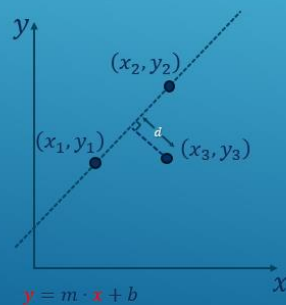
The next thing we want to show is that we can make a requirement that makes sure that unrelated points will not contribute to a cell they do not belong to, i.e., a cell of parameters of a line it does not reside in.

For that we define α as the minimal distance that makes a point far from a line ℓ by α being considered as not belong to ℓ .

Accumulator matrix works! Soundness



Accumulator matrix works! Soundness



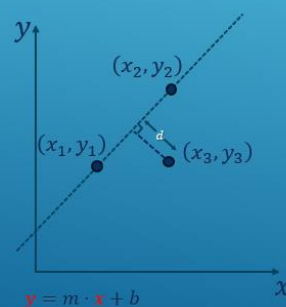
Define:
 $\alpha := \text{minimal distance threshold}$

We want:

$$d > \alpha$$

(x_3, y_3) does not contribute to the same or adjacent cells of the accumulator for θ_L and θ_R .

Accumulator matrix works! Soundness



WLOG we'll talk about left side:
 $\rho_3^L = y_3 \cdot \cos(\theta_L) - x_3 \cdot \sin(\theta_L)$

And again WLOG we'll look at the difference between (x_1, y_1) 's contribution and
We want to demand:

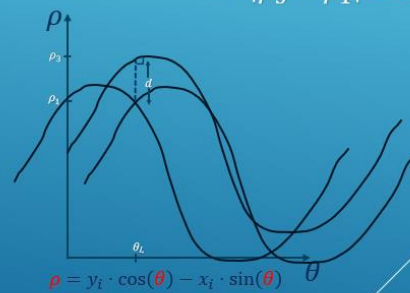
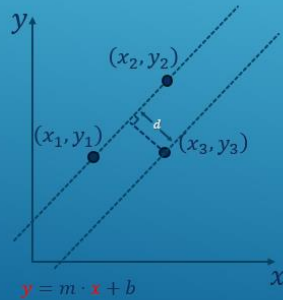
$$\left| \frac{\rho_1 + \#_p}{\Delta_p} - \frac{\rho_3 + \#_p}{\Delta_p} \right| > 2$$

$$\frac{1}{\Delta_p} |\rho_1 - \rho_3| > 2$$

Accumulator matrix works!

Soundness

Notice that:
For the same θ , the difference
in the offsets is exactly d !
 $\Rightarrow |\rho_3 - \rho_1| = d$



Accumulator matrix works!

Soundness

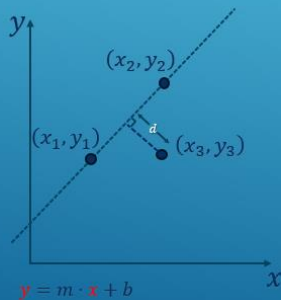
$$\frac{1}{\Delta_\rho} |\rho_1 - \rho_3| = \frac{1}{\Delta_\rho} \cdot d > \frac{1}{\Delta_\rho} \cdot \alpha$$

Therefore, we can simply demand:

$$\frac{1}{\Delta_\rho} \cdot \alpha > 2$$

Sounds like soundness is safe and sound!

$$\Delta_\rho < \frac{\alpha}{2}$$



What we showed is that if choosing $\Delta_\rho < \frac{\alpha}{2}$, a point distant more than α from a line ℓ will not contribute to a cell adjacent to where any two points residing on ℓ will contribute to.

Accumulator matrix works!

$$\Delta_\rho < \frac{\alpha}{2} \quad \Delta_\theta < \frac{1}{2} \cdot \frac{\Delta_\rho}{\rho_{max}}$$

$$\rho_{max} = \max_{i \in [n]} \left\{ x_i \cdot \cos \left(\arctan \left(\frac{y_i}{x_i} \right) \right) + y_i \cdot \sin \left(\arctan \left(\frac{y_i}{x_i} \right) \right) \right\}$$

Part 3: Comparing to a relevant paper

After our presentation, we understood that we unadvertantly overlooked some of the material while reading papers about Hough. We read the original (1959) and some recent papers (2022-2024) talking about Hough's algorithm and it's uses for medical purposes, face recognition, and robotics. But, on our exploration regarding the discrization of the space to an accumulator matrix and why does this works, we came up empty handed.

Thankfully, afterwards Freddie asked us to review his paper "Antialisaing the Hough Transform" from 1990, which deals with the same problem in a different way.

Part 3.1 Explaining the paper

The paper tries to resolve the aliasing issue. Aliasing is a term that describes undersampling a high frequency wave, thus getting a low frequency result. In our world of context, images. This means losing out on the fine details of the image because the accumulator matrix subsamples the image, e.g. too coarse for the image.

A bandlimited signal, is a signal with zero energy outside of a defined frequency range, meaning that it's Furier transform is non-zero only at finite amout of locations. Without delving too much into bandlimited and non-bandlimited signals. Sinusodials are bandlimited, and using Nyquist-Shannon sampling theorem we can solve the aliasing problem for those functions. Because we use bounded sinusodials, we are dealing with functions that the Nyquist-Shannon sampling theorem can't be applied to. And in this paper Freddie presented the "extended Hough transform" with carefully designed "influence functions" to make the transform effectively bandlimited before sampling.

In the standard Hough transform, each input point contributes equally to all lines that pass through it. The extended Hough transform, however, allows each point to contribute with varying weights to different lines based on their distance from the point using Influence functions. Influence functions are the key component that enables the extended Hough transform.

The paper discusses how to select appropriate influence functions to balance two competing objectives:

- a) Good localization in the image plane: This requires influence functions with small spatial support.
- b) Low sampling requirements in the parameter plane: This requires influence functions with small bandwidth.

The extended Hough transform with carefully chosen influence functions, creates an effectively bandlimited function before sampling, thus reducing aliasing effects. This allows for a more robust and accurate line detection, especially in the presence of noise or when dealing with complex images containing multiple lines or curves.

Part 3.2: comparing similarities

Our work and the paper are trying to reach a common goal, antialisaing the hough transform. In our work we tried to set some guidelines and limits, our big discovery was that with the right parametrization, the discrization works well, e.g. we will get gaussians of hot spots in relevant positions. This paper tries to achive the same goal, the whole idea is stemmed from antialiasing creating a similar but bandlimited function. In that sense we had the same thing we wanted to prove or achive.

Part 3.3: noting differences

There are a lot of differences between the two lines of thought. The main difference is the starting point, in our journey, we didn't know about bandlimited function, or in general how the problem we tackle is related to signal processing, we came up with our claims and proofs using the geometry of the problem, and we found it fascinating that this problem can be solved using claims that are so far from what we initially started with.

Tackling this problem from a geometrical perspective gave us the power to set some parameter thresholds and by that to indicate that we our sampling is "good enough". While the other perspective gave an intriguing original way of turning a non-bandlimited function to a bandlimited one, and showing the improvements it brings.

Part 3.4: final thoughts

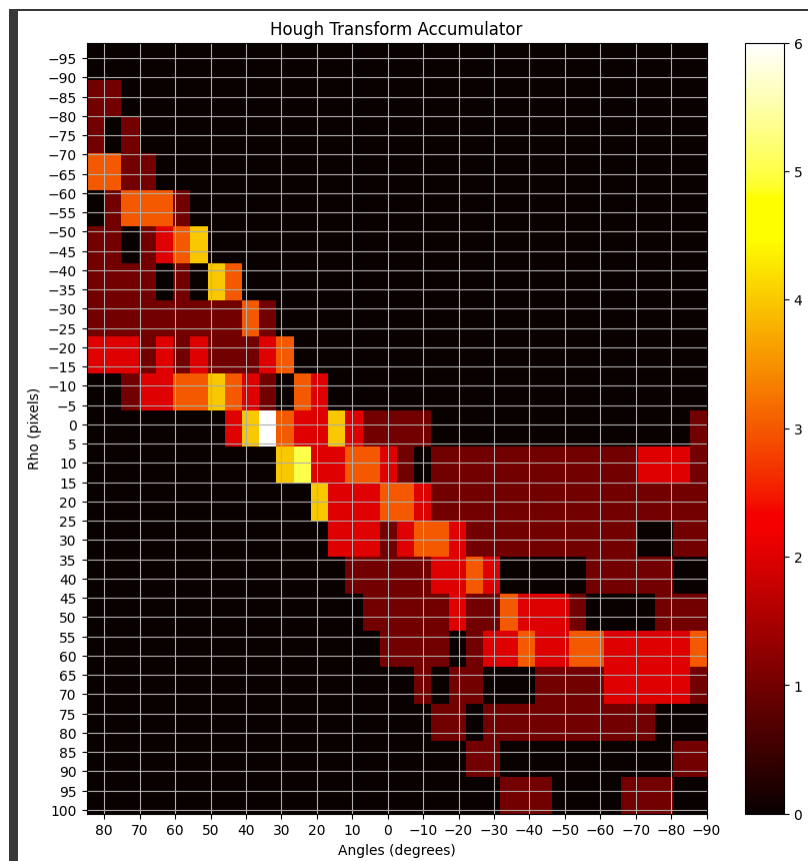
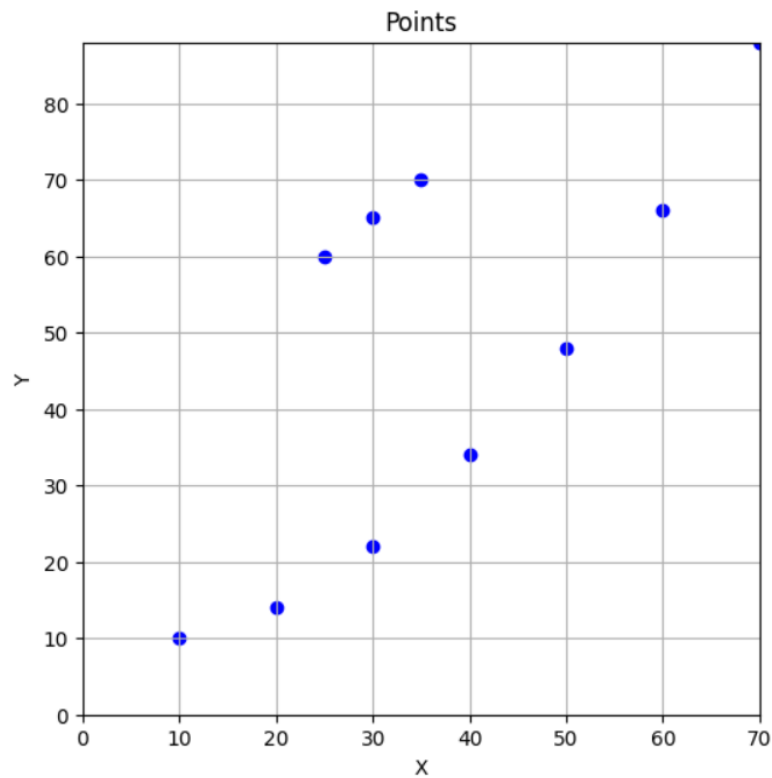
We think that during this project, we could have done better academic research, and even if we wouldn't find out about your paper, we might have came across aliasing problems, or even thought of another way to look at this problem from a different perspective. In other words, there are multiple ways to look at a certain problem, and multiple ways to solve or mitigate it.

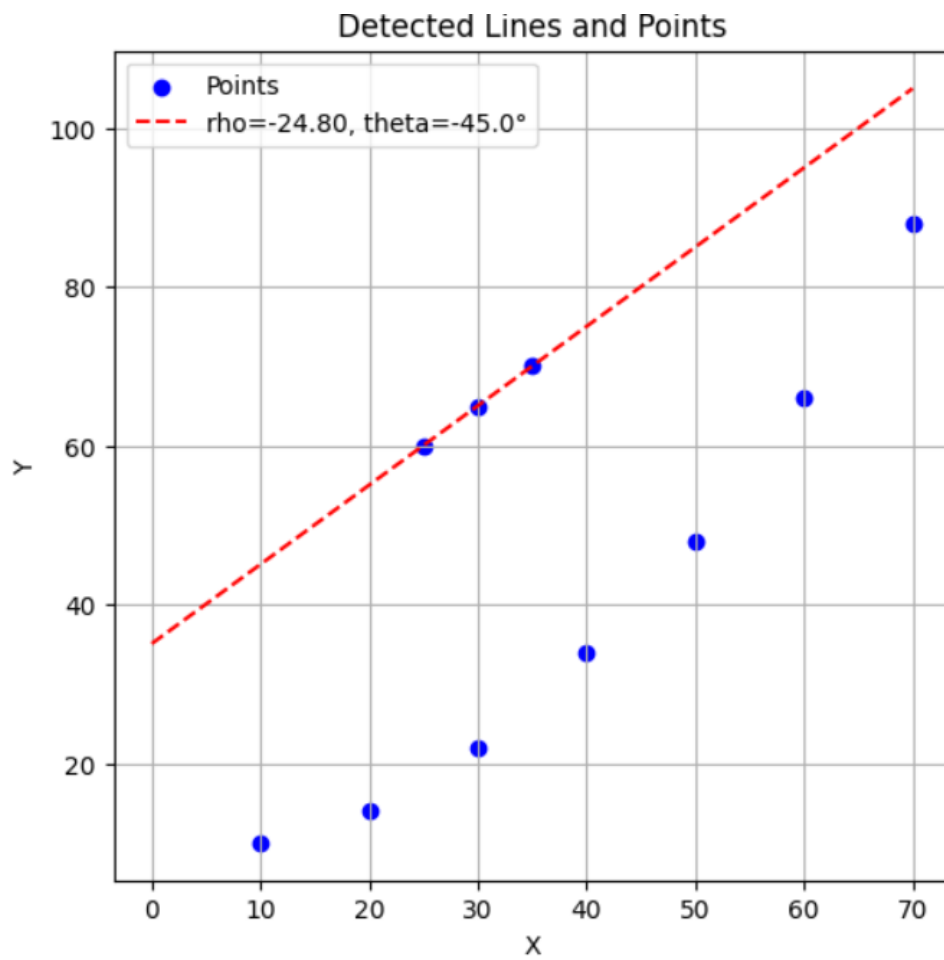
To sum up, this section has been all about two ways to solve the same problem, we found it very fun and insightful, even if we tried to solve a solved problem. We will conclude this section with an open question, are there any other perspectives that can show great result in solving problems related to ours? And more generally, what will be the next big unsolved problem that a new perspective will solve?

Part 4: First Hough demo

In this part we show the demo of the code we implemented. The code itself is in the file *"demo.ipynb"*.

We use a sample of points to demonstrate the algorithm. we chose $\alpha = 0.1$ tuned $\Delta\rho, \Delta\theta$ accordingly.



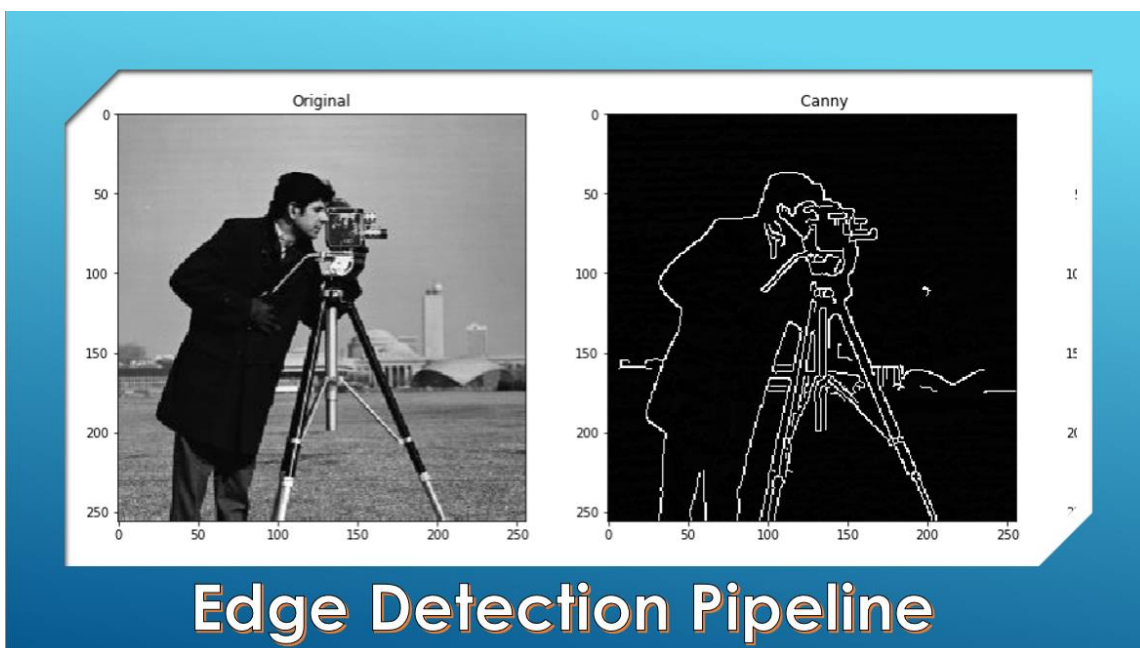


It can be seen that even though there is a sub-set of more than 3 points that construct a curve that can be interpreted as a line, due to our parameters tuning there is a precedence to the straighter line with less points.

Part 5: Introduction to Canny



John Canny is an Australian computer scientist, while pursuing his doctorate, he developed an edge detection algorithm. The input of his pipeline is an image (RGB / grayscale), and the output is a mask of the picture, indicating for each pixel in the image whether it is on the edge or not.



We will now quickly go over the main steps of the algorithm:

Step 0: Gray scaling

Not much to explain, the original algorithm works with grayscale images, there are improvements that work with RGB images with different convolution operators but we will not delve into them.

Step 1: Gaussian smoothing

Although we chose for our demo a generated image, meaning that this image does not suffer from camera artifacts, images taken from cameras may exhibit some or even a lot of noise, to set our algorithm up for success, we will need to remove those noises. We are achieving this goal by applying a gaussian filter on the image. This operation smooths out the image, making our existing edges more spread out (will be taken care of in step 3), but almost completely removing the noise out of the image.




$$\frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$


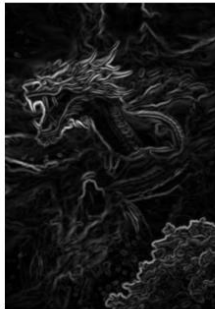
STEP 1: GAUSSIAN SMOOTHING

Step 2: Gradient calculations

In this step, we calculate for each pixel the size and direction of the gradient of the pixel using the Sobel operator (many alternatives to this operator, we chose it due to it's simplicity)

We convolve the image with a horizontal and diagonal gradient evaluation 3×3 matrices, thus getting for each pixel it's X and Y gradient, we move on with this information (The size and direction of the gradient of each pixel) throwing out the grayscale values.

$$\theta = \arctan\left(\frac{G_y}{G_x}\right)$$

$$G = \sqrt{G_x^2 + G_y^2}$$

STEP 2: Sobel operator

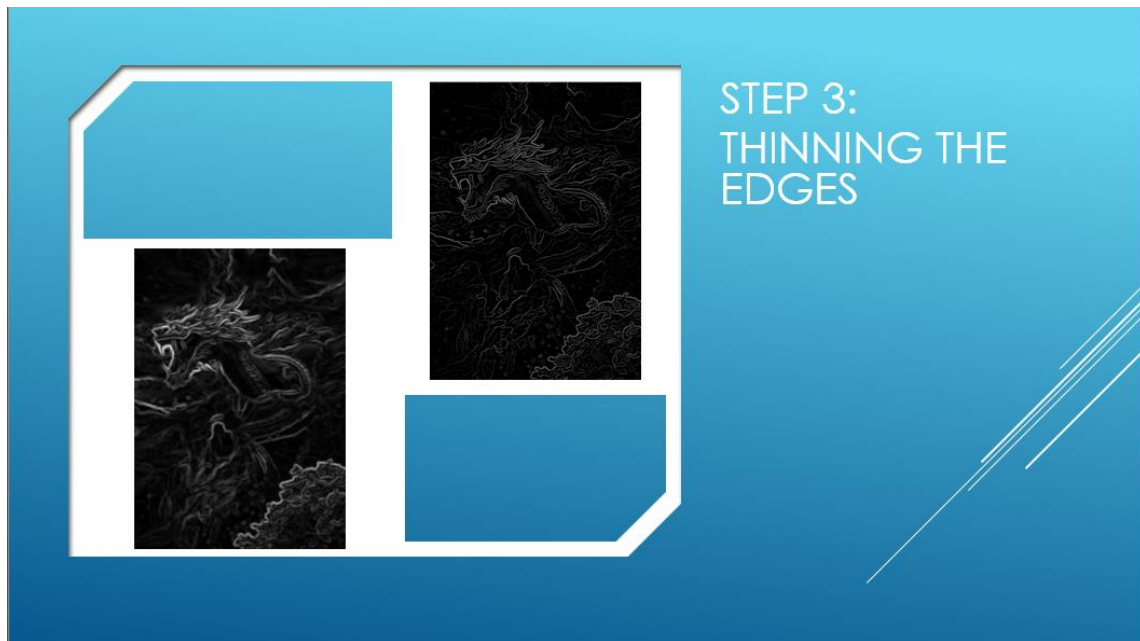
$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Part 3: Edge refinement

As we can see from the result of step 2, we get a lot of gradient clusters. This is bad, because normally we would expect outer edges to be as thin as possible, so the next step is all about thinning these edges. For each pixel, we check the size of it's gradient and direction. Based on the direction we check out the two closest pixels to it in the direction on the gradient (4 options based on the direction, pixel above and below, or on the right and left, or on the diagonals).

We compare the pixel's gradient to it's neighbors'. If its value is larger, the gradient stays. Otherwise, we set the gradient to be 0. This thins out thick edges, by only keeping lines with the highest gradients.



Step 4: double thresholding

By this point, we have a gradient size and value for each pixel, but our goal is a binary edge mask. To converge to this goal, we first want to classify each pixel as non-edge, weak edge or strong edge. We heuristically set low and high thresholds, and go over each pixel in the image to check:

If the gradient size is below the lower threshold, this pixel is classified as a non-edge.

If the gradient is higher than the upper threshold, the pixel is classified as a strong edge.

Otherwise it will be classified as a weak edge.

This will almost be the result as we desire. We only make a last tweak in the final step:



Step 5: hystersis

This final step is the one that gets us the binary pixel map we are looking for. The strong edges immediately enter the binary mask and the non-edges don't. The weak edges are going through one final check, if a weak edge connects two strong edges, it stays. Otherwise, we drop it.



Voila, we have the binary mask that we wanted.

One last thing to be asked, is why do we need this algorithm?

The answer is very simple. Our project mainly focuses on Hough transform. We wanted to be able to show it's image processing capabilities. The missing part was the transition from the picture to the relevant edge points.

When we look at relevant image processing papers using Hough, we can see a lot of them using Canny's algorithm or something similar on an image in order to set the stage for Hough to shine.

Part 6: Second demo – Hough and Canny together

Here we show our demo using a 'real' picture.

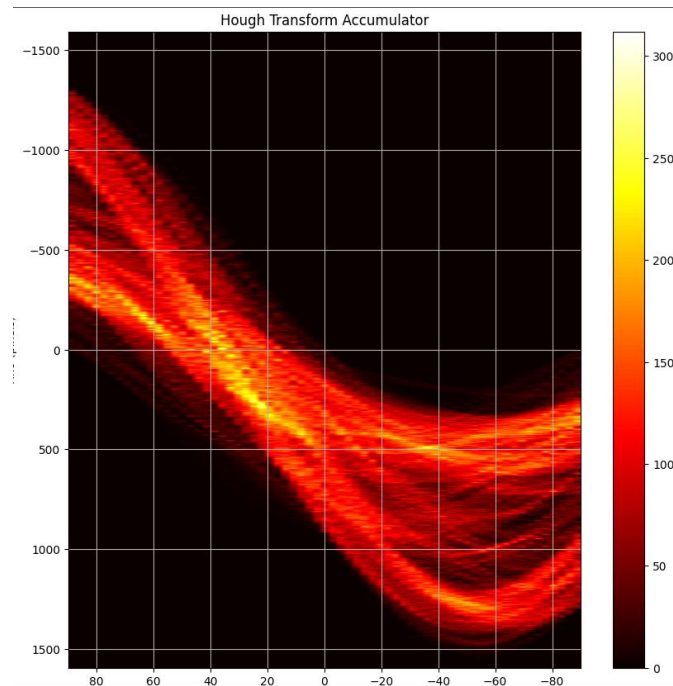
First and foremost, we want to use Hough, so we have to convert the picture into it's edges using Canny's algorithm we described earlier.

The pictures attached to the algorithm explanation are the pictures from the demo, after performing each step, so we will go on from there.



This is the final result of our implementation of Canny's algorithm , side by side with the original picture, and we can see that Canny did quite a good job, capturing the body of the dragon, the tree, and the mountatins.

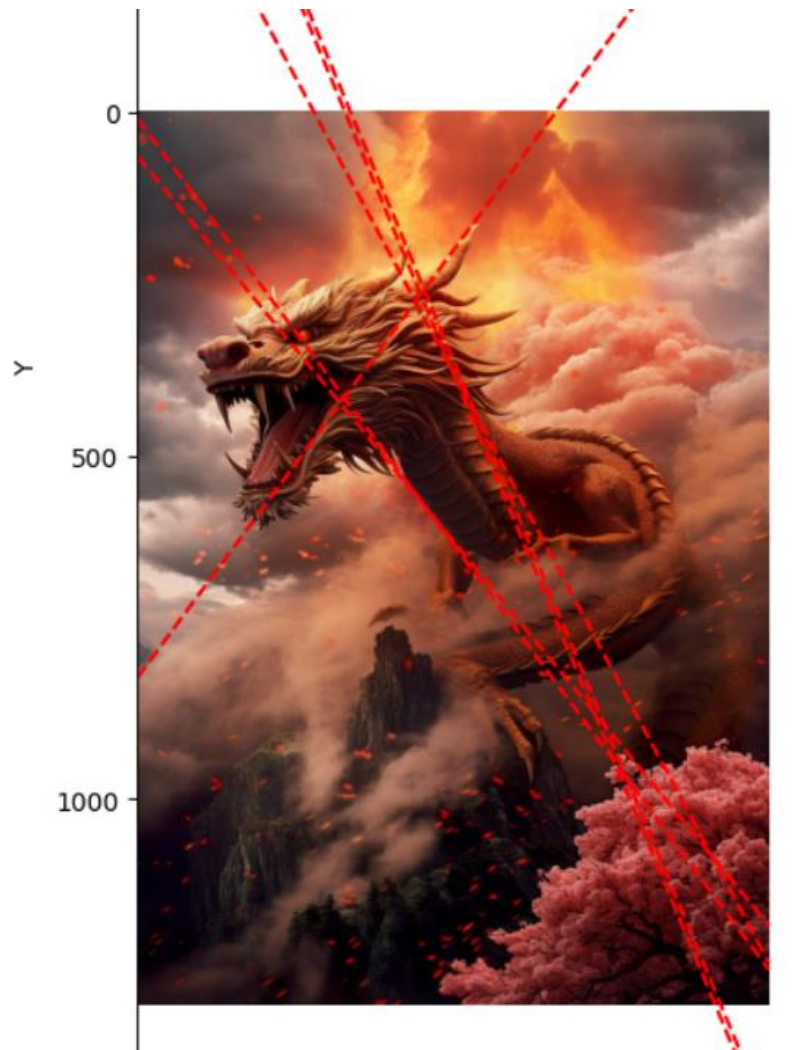
Now, we applied Hough's line detection algorithm to the mask, and got this accumulator matrix:



As we can all see, rather than getting hit points, we got hit gaussians.

One more thing to notice is that the grid of the graph does not match the grid of the accumulator matrix, because we set the grid to be very dense in order to find accurate lines in image with many pixels (unlike our first demo where we matched the grid of the graph with the grid of the matrix)

Meaning that if we choose the best 6 lines we get:



These lines are very good, they got the whole body of the dragon and the jaw line, but we can see that there are duplicate lines, or at least very similar lines tied together. This happens because two lines from the same gaussian cluster were selected.

To mitigate this effect, we implemented a final step, after finding a maxima of the matrix, we nullified all of its adjacent cells (within a predetermined radius).

The better way to do the same job is to treat the image as a graph, each pixel is a node, and each two adjacent pixels share an edge. After that run a BFS, to identify clusters and reduce them to a single line, for example taking the median values of the cluster.

The new final result is:



We can see that we got a much better result with more distinct lines. Each line describes a different key line in the original image.