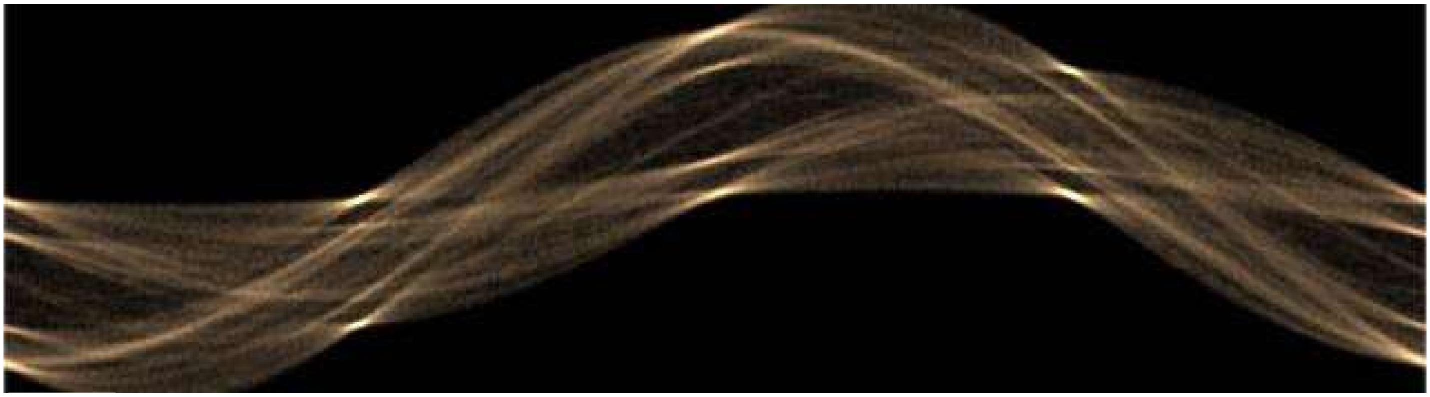


The Hough Transform

The Basics

The Hough transform is an incredible tool that lets you identify lines. Not just lines, but other shapes as well. In this article, I'll talk about the mechanics behind the Hough transform. It will involve a bit of math, but just elementary concepts you learned in school. In this article, we'll work with lines only, though the technique can be easily extended to other shapes.

Why the Hough Transform?

Lets say you take the snapshot of pole. You figure out edge pixels (using the Canny edge detector, the Sobel edge detector, or any other thing). Now you want a geometrical representation of the pole's edge.You want to know its slope, its intercept, etc. But right now the "edge" is just a sequence of pixels.

You can loop through all pixels, and some how figure out the slope and intercept. But that is one difficult task. Images are never perfect.

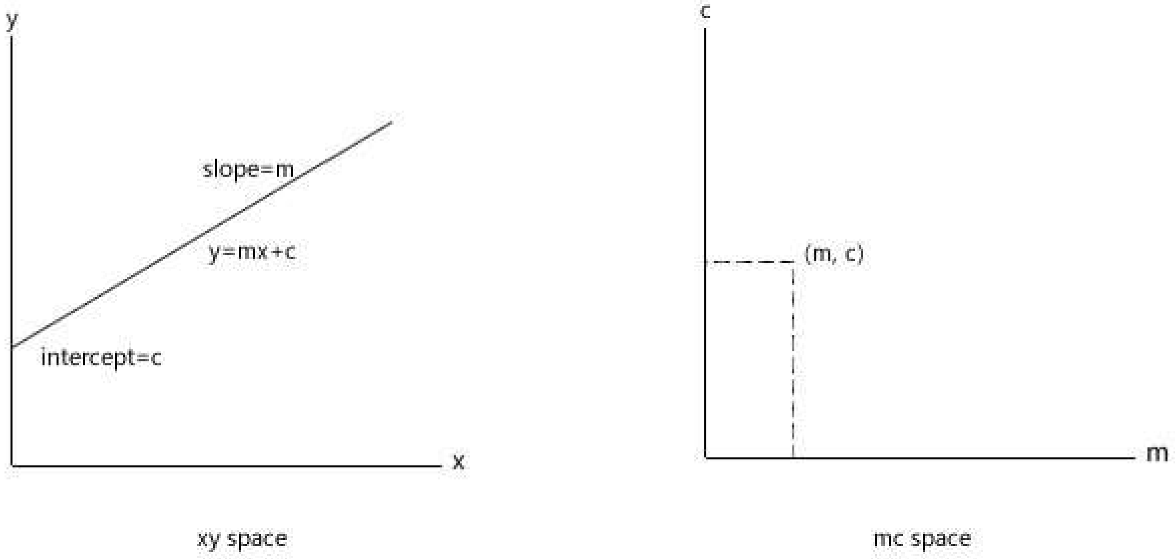
So you want some mechanism that give more weightage to pixels that are already in a line. This is exactly what the Hough Transform does.

It lets each point on the image "vote". And because of the mathematical properties of the transform, this "voting" allows us to figure out prominent lines in the image.

From lines to points

A lines is a collection of points. And managing a collection of points is tougher than managing a single point. Obviously. So the first thing we learn is how to represent a line as a single point, without losing any information about it.

This is done through the **m-c space**.



The m-c space

As shown in the above picture, every line has two quantities associated with it, the slope and the intercept. With these two numbers, you can describe a line completely.

So the parameter space, or the mc space was devised. Every line in the xy space is equivalent to a single point in the mc space. Neat eh?

From points to lines

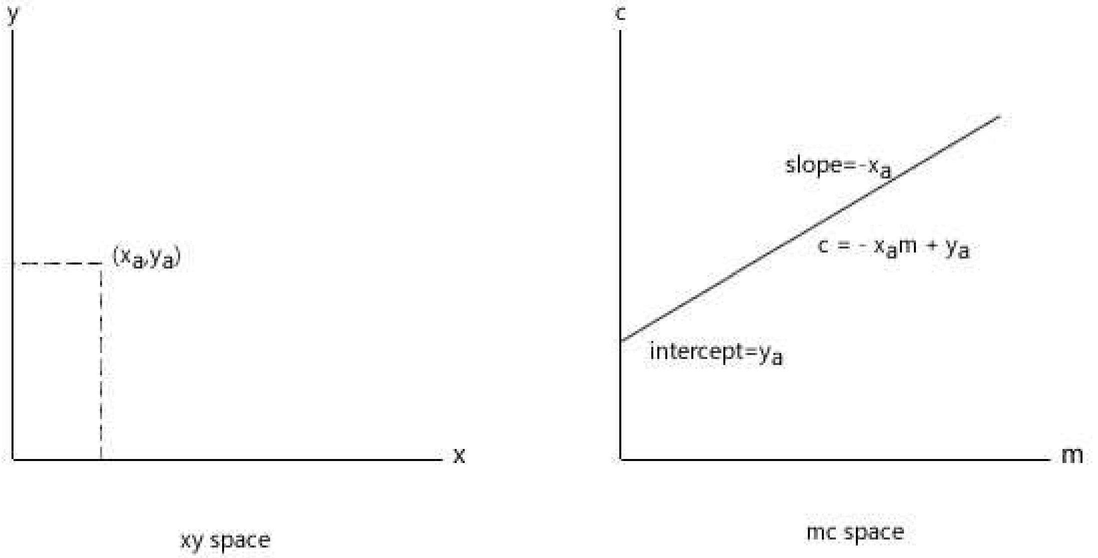
Now onto the next step. Consider a point (say, (x_a, y_a))in the xy space. What would its representation in the mc space be?

For starters, you could guess that infinite lines pass through a point. So, for every line passing through (x_a, y_a), there would be a point in the mc space.

And you're correct there, because of the following:

1. Any line passing through (x_a, y_a): $y_a = mx_a + c$
2. Rearranging: $c = - x_am + y_a$
3. The above is the equation of a line in the mc space.

So, a point in the xy space is equivalent to a line in the mc space.

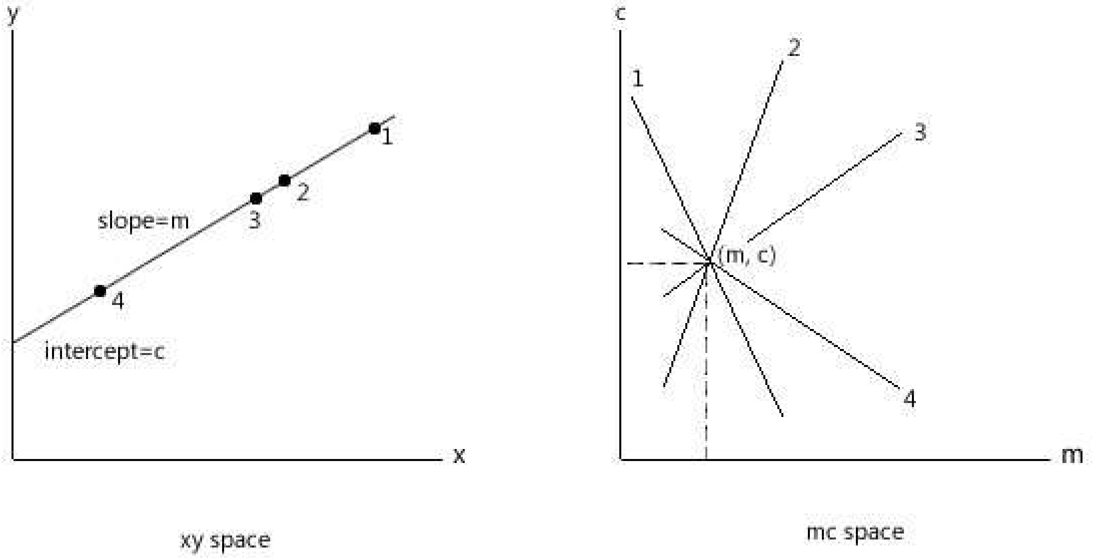


How does this help us?

The Hough transform is all about doing what we just learned: converting points in the xy space to lines in the mc space.

You taken an edge detected image, and for every point that is non black, you draw lines in the mc place. Obviously, some lines will intersect. These intersections mark are the parameters of the line.

The following picture will clarify the idea:



An example of houghlines

The points 1, 2, 3, and 4 are represented as various lines in the mc space. And the intersection of these lines is equivalent to the original line.

Great!

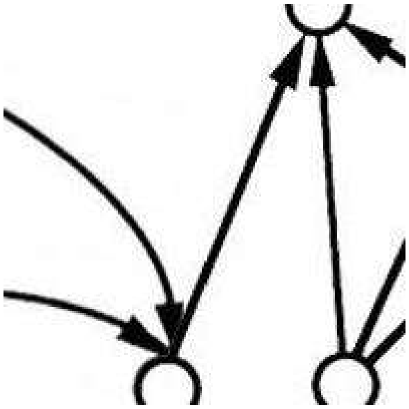
Now you know the basic idea of how the Hough transform works. But there's one big flaw in the We'll discuss how to resolve this issue in the next article.

More in the series

This tutorial is part of a series called **The Hough Transform**:

1. **The Basics**
2. The Normal form (/tutorials/hough-transform-normal/)

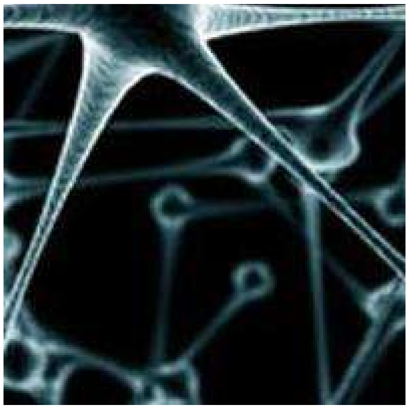
Related posts



(/tutorials/7-unique-neural-network-architectures/)

7 unique neural network architectures (/tutorials/7-unique-neural-network-architectures/)

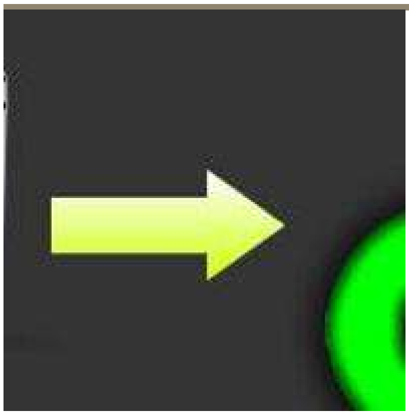
Read more (/tutorials/7-unique-neural-network-architectures/)



(/tutorials/biological-neurons/)

All about biological neurons (/tutorials/biological-neurons/)

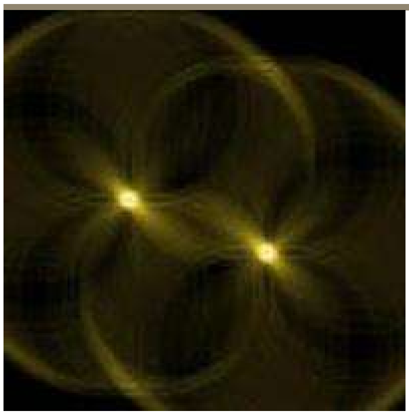
Read more (/tutorials/biological-neurons/)



(/tutorials/capturing-images/)

Capturing images (/tutorials/capturing-images/)

Read more (/tutorials/capturing-images/)



(/tutorials/circle-hough-transform/)

Circle Hough Transform (/tutorials/circle-hough-transform/)

Read more (/tutorials/circle-hough-transform/)

(http://utkarshsinha.com/)

Utkarsh Sinha created AI Shack in 2010 and has since been working on computer vision and related fields. He is currently at Microsoft working on computer vision.

ALSO ON **AI SHACK**

Mathematical Morphology: Basic …

6 years ago · 1 comment

I'm sure you've studied morphology in Biology. Well, that was biological …

First artificial neurons: The McCulloch-Pitts …

6 years ago · 1 comment

The McCulloch-Pitts model was an extremely simple artificial neuron. The …

Why OpenCV?

6 years ago · 1 comment

If you come from the land of Matlab, you might need some convincing to …

Obstacl the Bug-

6 years ag

An obstac algorithm reach fro

4 Comments

1 Login ▾

G

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

♡ 12 Share

Best Newest Oldest

鄭華

4 years ago

Thank you for sharing.
However, it would be better to change the slope in mc-space (fig.2) to minus, if Xa > 0

0 0 Reply • Share ›

Alexis Paques

4 years ago edited

[deleted]

0 1 Reply • Share ›

Utkarsh Sinha Mod → Alexis Paques

4 years ago

I wrote this article back in 2010-2011. I believe the other website should mention the source :)

5 0 Reply • Share ›

Alexis Paques → Utkarsh Sinha

4 years ago

[I could not see the date of the article; Sorry for this article, I apologize for the confusion]

8 0 Reply • Share ›

About AI Shack

Learn about the latest in AI technology with in-depth tutorials on vision and learning!

Follow @utkarshsinha

More... (/about/)

Get started

Get started with OpenCV (/tracks/opencv-basics/)
Track a specific color on video (/tutorials/tracking-colored-objects-opencv/)
Learn basic image processing algorithms (/tracks/image-processing-algorithms-level-1/)
How to build artificial neurons? (/tutorials/single-neuron-dictomizer/)
Look at some source code (http://github.com/aishack/)

AI Shack

Like Page 3.6K likes

Created by Utkarsh Sinha (http://utkarshsinha.com/)