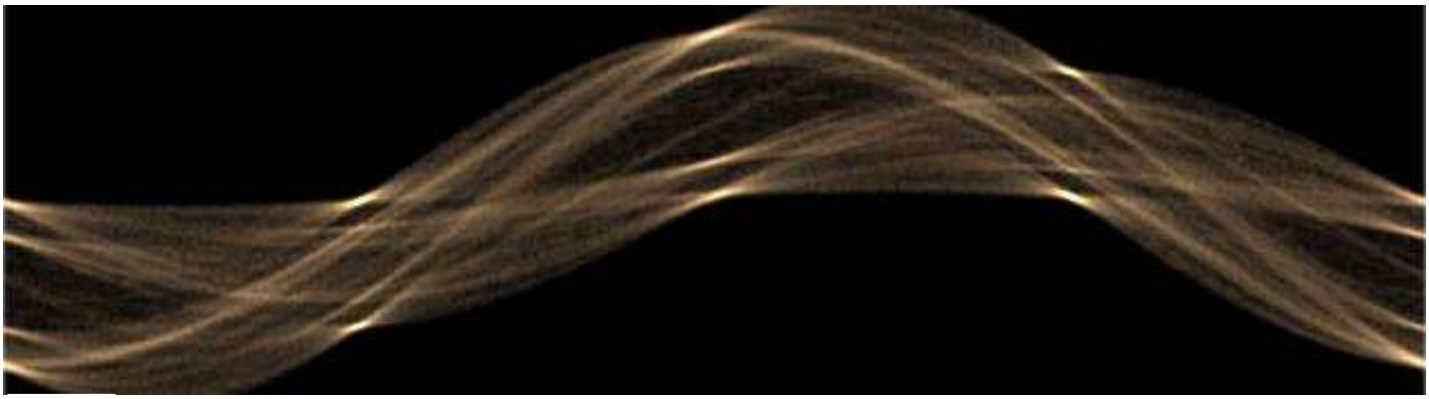


The Hough Transform

The Normal form

The flaw

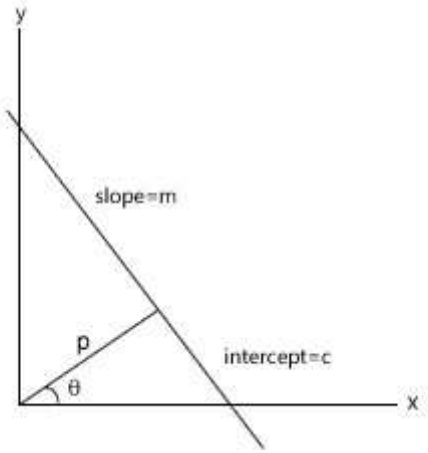
The Hough transform described in the previous article (/tutorials/hough-transform-basics/) has an obvious flaw. The value of m (slope) tends to infinity for vertical lines. So you need infinite memory to be able to store the mc space. Not good.

Series: The Hough Transform:

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

The solution

The problem is resolved by using a different parametrization. instead of the slope-intercept form of lines, we use the normal form.



The normal form

In this representation, a line is formed using two parameters - an angle θ and a distance *p*. p is the length of the normal from the origin (0, 0) onto the line. and θ is the angle this normal makes with the x axis. This solves the flaw perfectly.

The angle θ can be only from -90^{o} to +90^{o} and the length p can range from 0 to length of diagonal of the image. These values are finite, and thus you can "store" them on a computer without much trouble.

In this representation, the equation of the line is:

$$p = x_1 \cos\theta + y_1 \sin\theta$$

where (x_1, y_1) is a point through which the line passes.

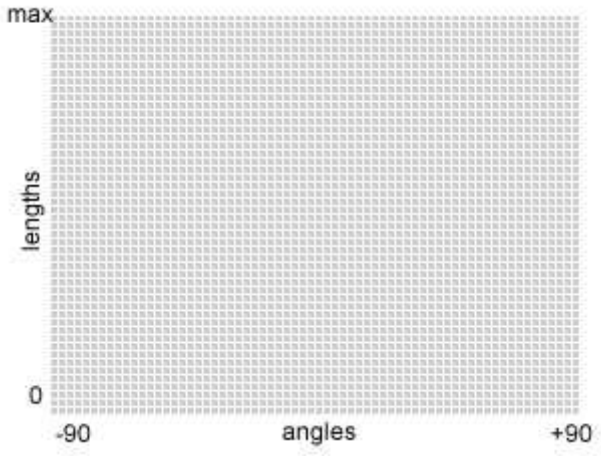
With this new equation, we have a few changes in the end-point to line "transition" from the xy space to the pθ space. A line in the xy space is still equivalent to a point in the pθ space. But a point in the xy space is now equivalent to a sinusoidal curve in the pθ space.

The **image at the top** of this article is an actual pθ space. The sinusoidal curves have been generated while trying to figure out lines an image.

Implementation

With those ideas through, we're ready to implement the Hough transform. The idea is to let each pixel "vote". So an array of accumulator cells is created.

In our case, the accumulator cells form a 2D array. The horizontal axis is for the different θ values and the vertical axis for p values.



The accumulator

Next, you loop through every pixel of the edge detected image (Hough works only on edge detected images... not on normal images).

If a pixel is zero, you ignore it. It's not an edge, so it can't be a line. So move on to the next pixel.

If a pixel is nonzero, you generate its sinusoidal curve (in the pθ space). That is, you take θ = -90 and calculate the corresponding p value. Then you "vote" in the accumulator cell (θ, p). That is, you increase the value of this cell by 1. Then you take the next θ value and calculate the next p value. And so on till θ = +90. And for every such calculation, making sure they "vote".

So for every nonzero pixel, you'll get a sinusoidal curve in the pθ space (the accumulator cells). And you'll end up with an image similar to the one at the top.

Detecting lines

The image at the top has several "bright spots". A lot of points "voted" for this spot. And these are the parameters that describe the lines in the original image. Simple as that.

Accuracy

The accuracy of the Hough transform depends on the number of accumulator cells you have. Say you have only -90^0, -45^0, 0^0, 45^0 and 90^0 as the cells for θ values. The "voting" process would be terribly inaccurate. Similarly for the p axis. The more cells you have along a particular axis, the more accurate the transform would be.

Also, the technique depends on several "votes" being cast into a small region. Only then would you get those bright spots. Otherwise, differentiating between a line and background noise is one really tough job.

Done

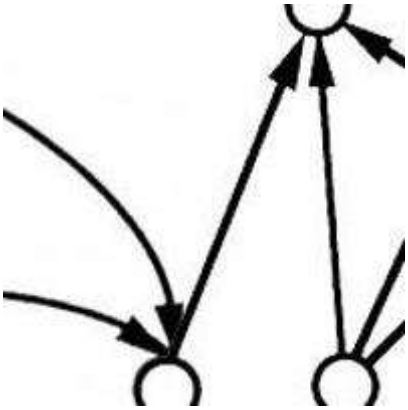
Now you know how the standard Hough transform works! You can try implementing it... should be simple enough.

More in the series

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

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

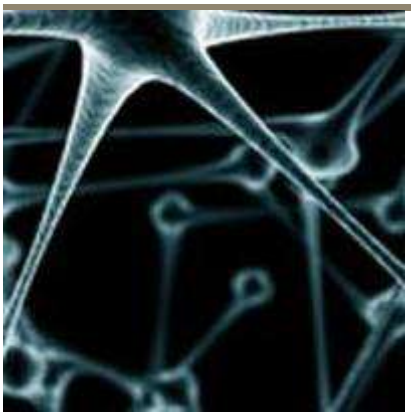
Related posts



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

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

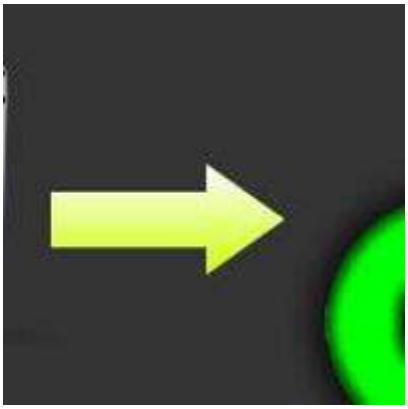
Read more (/tutorials/7-unique-neural-network-architectures/)



(/tutorials/biological-neurons/)

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

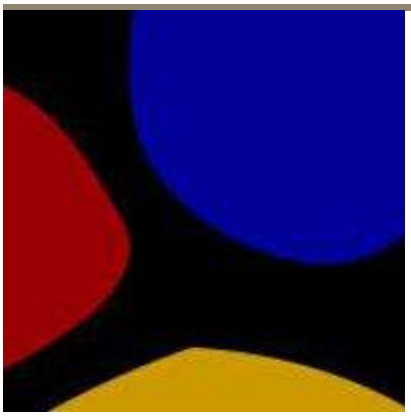
Read more (/tutorials/biological-neurons/)



(/tutorials/capturing-images/)

Capturing images (/tutorials/capturing-images/)

Read more (/tutorials/capturing-images/)



(/tutorials/connected-component-labelling/)

Connected Component Labelling (/tutorials/connected-component-labelling/)

Read more (/tutorials/connected-component-labelling/)

(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**

The Hough Transform: The Basics

6 years ago • 4 comments

The Hough transform is an incredible tool that lets you identify lines. Not just …

Mathematical Morphology: …

6 years ago • 2 comments

Mathematical morphology isn't only about erosion and dilation. Those are just the …

Installing and Getting OpenCV running

6 years ago • 1 comment

Download OpenCV The first thing to do is get OpenCV. You can download it from …

Thresho

6 years ag

Introducti
one of the
technique

4 Comments

Login

G

Join the discussion...

LOG IN WITH OR SIGN UP WITH DISQUS

Name

7 Share Best Newest Oldest

botaybochan botoanthan
5 years ago

love this guy

23 0 Reply • Share ›

A

Ashok Kumar
3 years ago

Very lucid set of articles!
I have just one comment about the parameter range for (theta, p). The given ranges of theta in the interval [-90, +90] and p in [0,+diag] seem to exclude half the lines. Two of the following parameter intervals would include all the lines
(i) theta in [-180, 180] and p in [0, diag]
(ii) theta in [-90, +90] and p in [-diag, +diag]

The sinusoidal plot at the top seems to suggest [-180, 180] interval for theta.

0 0 Reply • Share ›

herb munson
5 years ago

In searching for more on the Normal Form, it might be helpful to know f(r,theta) is also the Hesse Normal Form.

0 0 Reply • Share ›

This comment was deleted.

Utkarsh Sinha Mod Guest
5 years ago

That's an interesting issue. How did you go about solving the problem?

0 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/)