Line Detection Using Hough Transform

# Page 1: Introduction

Detecting shapes like lines and edges in images is a key part of computer vision. Whether it's for self-driving cars recognizing road lanes or apps scanning documents, identifying lines accurately helps machines understand what they "see."  
  
One of the most reliable techniques for this is the Hough Transform. This method makes it possible to find straight lines in images, even if the lines are broken, noisy, or slightly faded.  
  
Before using the Hough Transform, we usually run an edge detection algorithm (like Canny) to highlight where big changes in brightness occur — these are potential edges. The Hough Transform then steps in to find which of those edges actually form straight lines.  
  
In this document, we'll explore how the Hough Transform works, how to implement it in Python using OpenCV, and where it can be applied in the real world.

# Page 2: How the Hough Transform Works

## Why Not Use the y = mx + c Formula?

When we think about lines, we often use the equation y = mx + c. But this breaks down when a line is vertical — the slope m becomes infinite. That’s why we switch to a different way of representing lines: polar coordinates.

## Using Polar Coordinates

In polar form, any straight line can be written as:  
  
ρ = x cos(θ) + y sin(θ)  
  
Here:  
- ρ is the distance from the origin to the closest point on the line  
- θ is the angle of the line's normal (perpendicular)  
  
This format works for all lines — vertical, horizontal, or diagonal.

## What Is Hough Space?

Imagine every edge point in the image voting for all the lines it could be part of. These “votes” are collected in something called an accumulator (a 2D array for ρ and θ). Where a lot of votes overlap, we likely have a real line.  
  
So in short:  
- Each edge point becomes a curve in the Hough space.  
- Where curves intersect, there's probably a line.  
- We can find these intersections and turn them back into line coordinates.

# Page 3: How to Implement It in Python

Here’s a basic version of how we’d use the Hough Transform in a Python script with OpenCV:

import cv2  
import numpy as np  
import matplotlib.pyplot as plt  
  
# Load image and convert to grayscale  
img = cv2.imread('image.jpg')  
gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  
  
# Detect edges  
edges = cv2.Canny(gray, 50, 150)  
  
# Detect lines using Hough Transform  
lines = cv2.HoughLines(edges, 1, np.pi / 180, 150)  
  
# Draw lines on the original image  
for rho, theta in lines[:, 0]:  
 a = np.cos(theta)  
 b = np.sin(theta)  
 x0 = a \* rho  
 y0 = b \* rho  
 x1 = int(x0 + 1000 \* (-b))  
 y1 = int(y0 + 1000 \* (a))  
 x2 = int(x0 - 1000 \* (-b))  
 y2 = int(y0 - 1000 \* (a))  
 cv2.line(img, (x1, y1), (x2, y2), (0, 0, 255), 2)  
  
# Show the result  
plt.imshow(cv2.cvtColor(img, cv2.COLOR\_BGR2RGB))  
plt.title('Lines Detected')  
plt.axis('off')  
plt.show()

This code loads an image, finds edges, runs the Hough Transform, and draws the detected lines in red.

# Page 4: Real-World Applications

The Hough Transform isn’t just a cool math trick — it’s used all over the place:

🚗 Lane Detection in Self-Driving Cars   
By detecting road lines, the Hough Transform helps autonomous vehicles stay in their lanes, even when the paint is worn out or partially blocked.  
  
📄 Document Scanning   
Apps use it to identify table lines, text boxes, or margins when scanning documents or forms.  
  
🛠️ Industrial Machines   
In manufacturing, cameras use line detection to check if parts are straight, aligned, or cut correctly.  
  
🤖 Robotics   
Robots can use line detection to understand their surroundings — like walls, floors, or paths.

# Page 5: Summary and Reflections

## What Makes the Hough Transform Special?

- It doesn’t care if a line is broken or faded.  
- It works on noisy images.  
- It detects lines based on a voting system — so multiple weak signals can still point to a strong result.

## Limitations to Be Aware Of

- It can be slow on large images.  
- You might get too many lines unless you tune the parameters.  
- Choosing the right values for ρ, θ, and thresholds can take trial and error.

## Final Thoughts

Line detection using the Hough Transform is a powerful tool, especially when paired with edge detection. It opens up many possibilities in machine vision and helps machines interpret the visual world more like humans do.

## 📚 References

- OpenCV Docs: https://docs.opencv.org  
- Gonzalez & Woods, Digital Image Processing  
- Duda & Hart, “Use of the Hough Transformation to Detect Lines and Curves in Pictures” (1972)  
- TutorialsPoint, GeeksforGeeks, and OpenCV-Python Tutorials