Assignment-3

Hough Transform

Summer 2022 CS 59000 VT- Topic Computer Sci-XB9 Cross list

By- Rudraksh Sugandhi

Report;

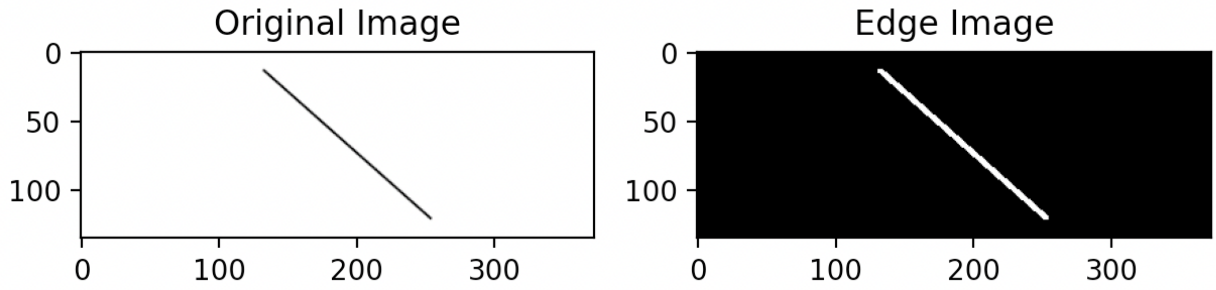
Code:

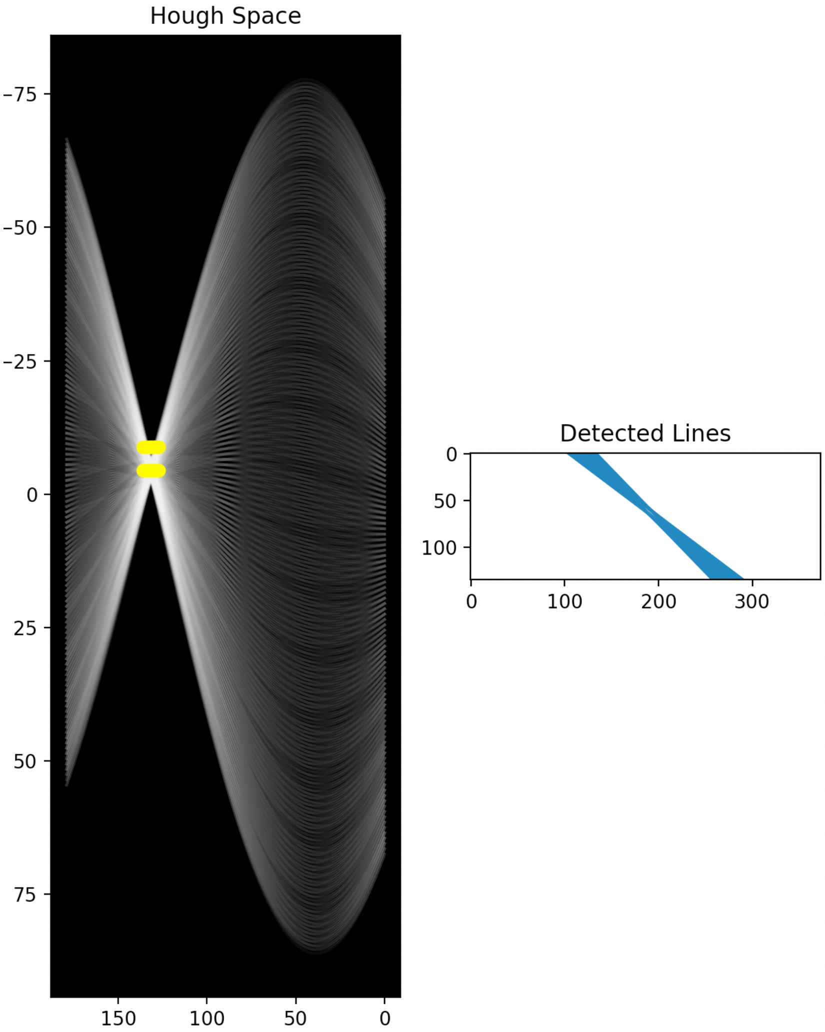
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.lines as mlines

# line detection method in vectorized format
def line_detection_vectorized(image, edge_image, num_rhos=180, num_thetas=180,
t_count=220):
  edge_height, edge_width = edge_image.shape[:2]
  edge_height_half, edge_width_half = edge_height / 2, edge_width / 2
  #
  d = np.sqrt(np.square(edge_height) + np.square(edge_width))
  dtheta = 180 / num_thetas
  drho = (2 * d) / num_rhos
  #
  thetas = np.arange(0, 180, step=dtheta)
  rhos = np.arange(-d, d, step=drho)
  #
  cos_thetas = np.cos(np.deg2rad(thetas))
  sin_thetas = np.sin(np.deg2rad(thetas))
  #
  accumulator = np.zeros((len(rhos), len(rhos)))
  #
  figure = plt.figure(figsize=(12, 12))
  subplot1 = figure.add_subplot(1, 4, 1)
  subplot1.imshow(image)
  subplot2 = figure.add_subplot(1, 4, 2)
  subplot2.imshow(edge_image, cmap="gray")
  subplot3 = figure.add_subplot(1, 4, 3)
  subplot3.set_facecolor((0, 0, 0))
  subplot4 = figure.add_subplot(1, 4, 4)
```

```
subplot4.imshow(image)
#
edge_points = np.argwhere(edge_image != 0)
edge_points = edge_points - np.array([[edge_height_half, edge_width_half]])
#
rho_values = np.matmul(edge_points, np.array([sin_thetas, cos_thetas]))
#
accumulator, theta_vals, rho_vals = np.histogram2d(
    np.tile(thetas, rho_values.shape[0]),
    rho_values.ravel(),
    bins=[thetas, rhos]
)
accumulator = np.transpose(accumulator)
lines = np.argwhere(accumulator > t_count)
rho_idxs, theta_idxs = lines[:, 0], lines[:, 1]
r, t = rhos[rho_idxs], thetas[theta_idxs]

for ys in rho_values:
  subplot3.plot(thetas, ys, color="white", alpha=0.05)

subplot3.plot([t], [r], color="yellow", marker='o')

for line in lines:
  y, x = line
  rho = rhos[y]
  theta = thetas[x]
  a = np.cos(np.deg2rad(theta))
  b = np.sin(np.deg2rad(theta))
  x0 = (a * rho) + edge_width_half
  y0 = (b * rho) + edge_height_half
  x1 = int(x0 + 1000 * (-b))
  y1 = int(y0 + 1000 * (a))
  x2 = int(x0 - 1000 * (-b))
  y2 = int(y0 - 1000 * (a))
  subplot3.plot([theta], [rho], marker='o', color="yellow")
  subplot4.add_line(mlines.Line2D([x1, x2], [y1, y2]))

subplot3.invert_yaxis()
subplot3.invert_xaxis()

subplot1.title.set_text("Original Image")
subplot2.title.set_text("Edge Image")
subplot3.title.set_text("Hough Space")
subplot4.title.set_text("Detected Lines")
```

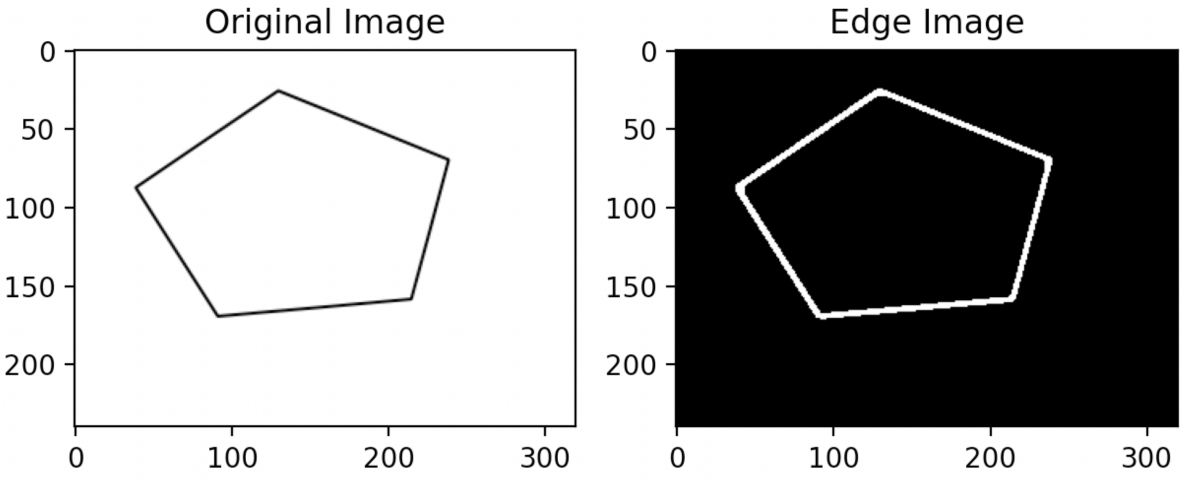
```
  plt.show()
  return accumulator, rhos, thetas


if __name__ == "__main__":
  for i in range(3):
    image = cv2.imread(f"star.png")
    edge_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    edge_image = cv2.GaussianBlur(edge_image, (3, 3), 1)
    edge_image = cv2.Canny(edge_image, 100, 200)
    edge_image = cv2.dilate(
        edge_image,
        cv2.getStructuringElement(cv2.MORPH_RECT, (5, 5)),
        iterations=1
    )
    edge_image = cv2.erode(
        edge_image,
        cv2.getStructuringElement(cv2.MORPH_RECT, (5, 5)),
        iterations=1
    )
    line_detection_vectorized(image, edge_image)
```

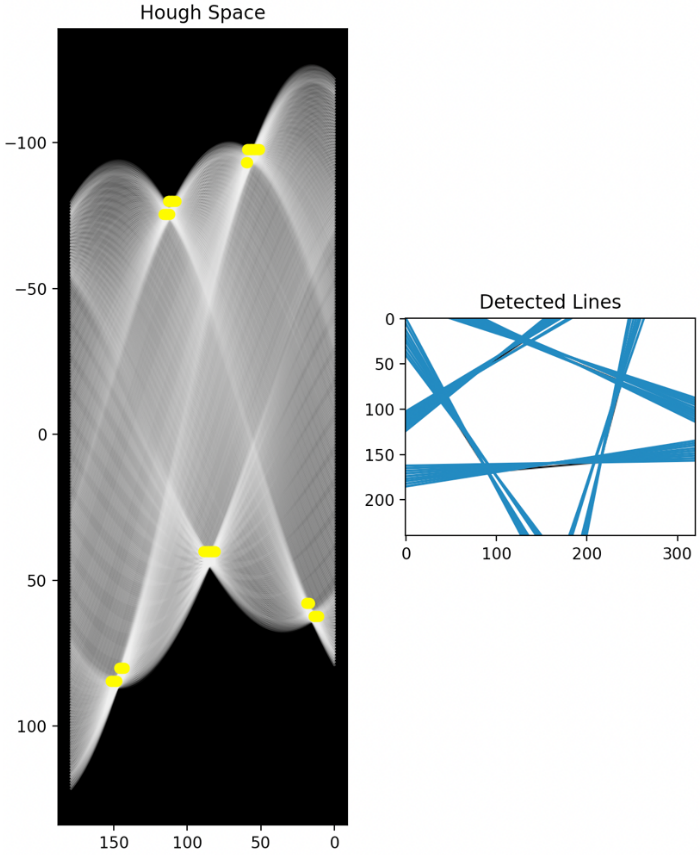
Output- a.


Original Image
0
50
100
0
100
200
300
Edge Image
0
50
100
0
100
200
300

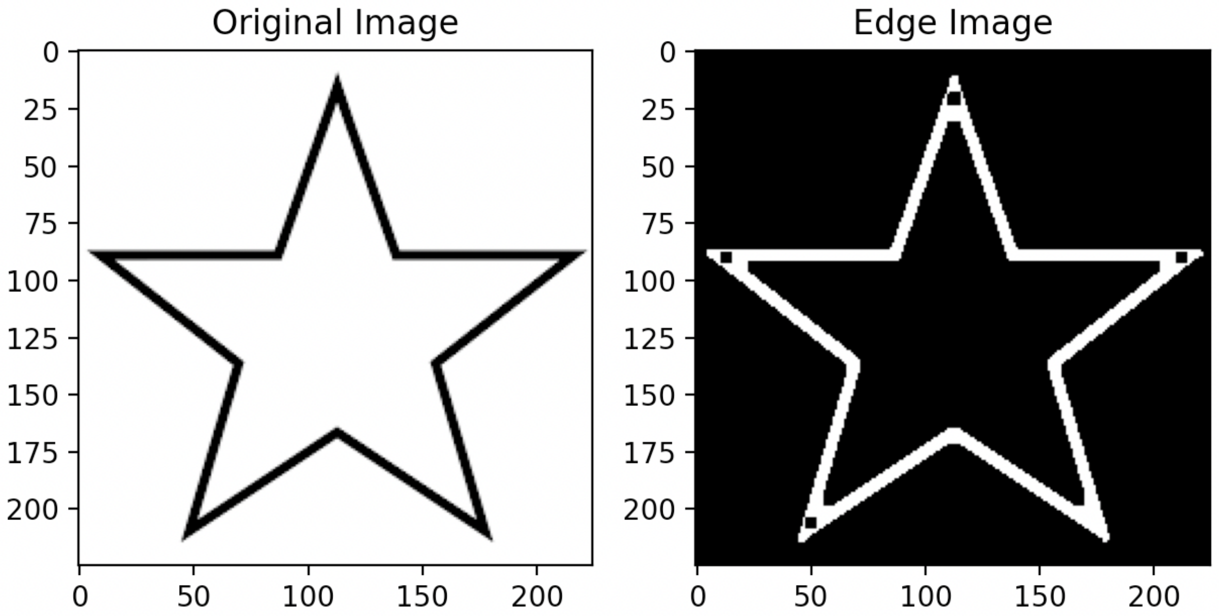

Hough Space
−75
−50
−25
0
25
50
75
150
100
50
0
Detected Lines
0
50
100
0
100
200
300

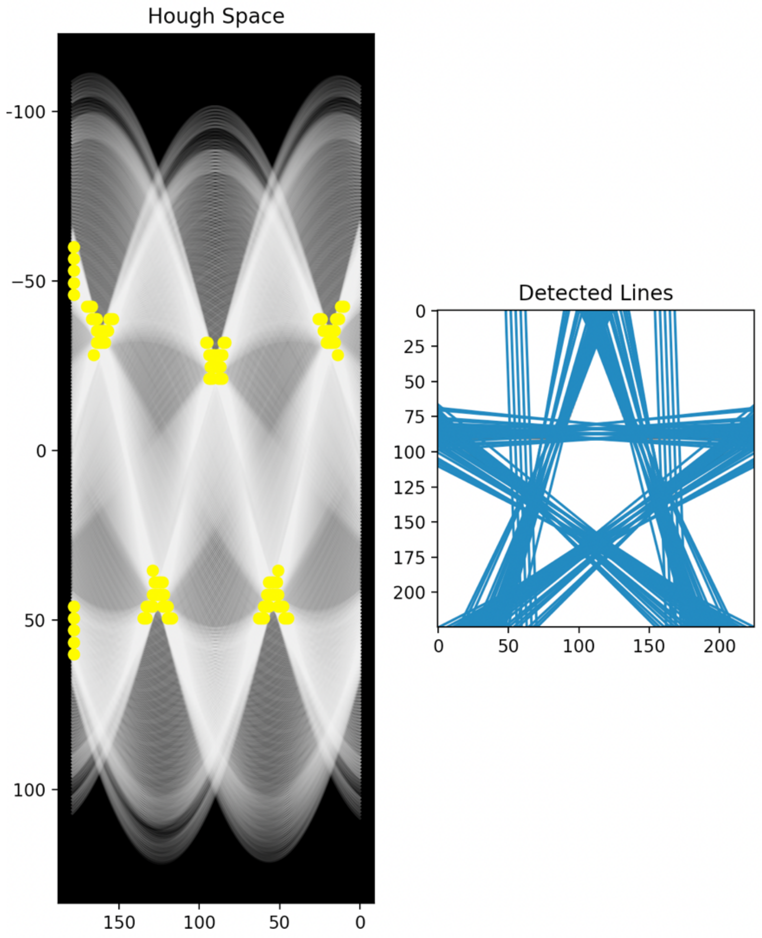
b.


Original Image
Edge Image
0
50
100
150
200
0
100
200
300


Hough Space
−100
−50
0
50
100
150
100
50
0
Detected Lines
0
50
100
150
200
0
100
200
300

<u>c.</u>


Original Image
Edge Image
0
25
50
75
100
125
150
175
200
0
50
100
150
200
0
25
50
75
100
125
150
175
200
0
50
100
150
200



Hough transform code using OpenCV and HoughLine method

```
import cv2
import numpy as np

img = cv2.imread('dave.jpg')
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
edges = cv2.Canny(gray,50,150,apertureSize = 3)

lines = cv2.HoughLines(edges,1,np.pi/180,200)
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)

cv2.imwrite('houghlines3.jpg',img)
```

Reason- Difference between traditional and OpenCV method is **time complexity** and Space complexity. Apart from both of these the code version with Hough line is able to describe lines in image with more accuracy.