edge

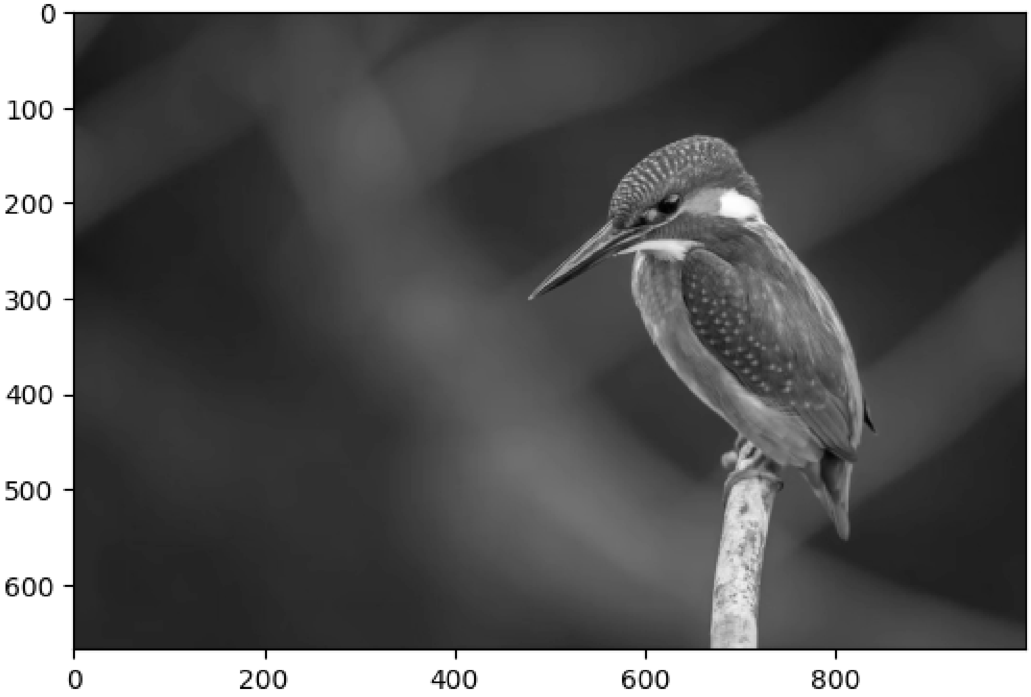
February 21, 2024

[2]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

[3]:
```
# read the image
image = cv2.imread("/home/nmit/Downloads/bird.jpg")
```

[4]:
```
# convert it to grayscale
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

[5]:
```
# show the grayscale image
plt.imshow(gray, cmap="gray")
plt.show()
```

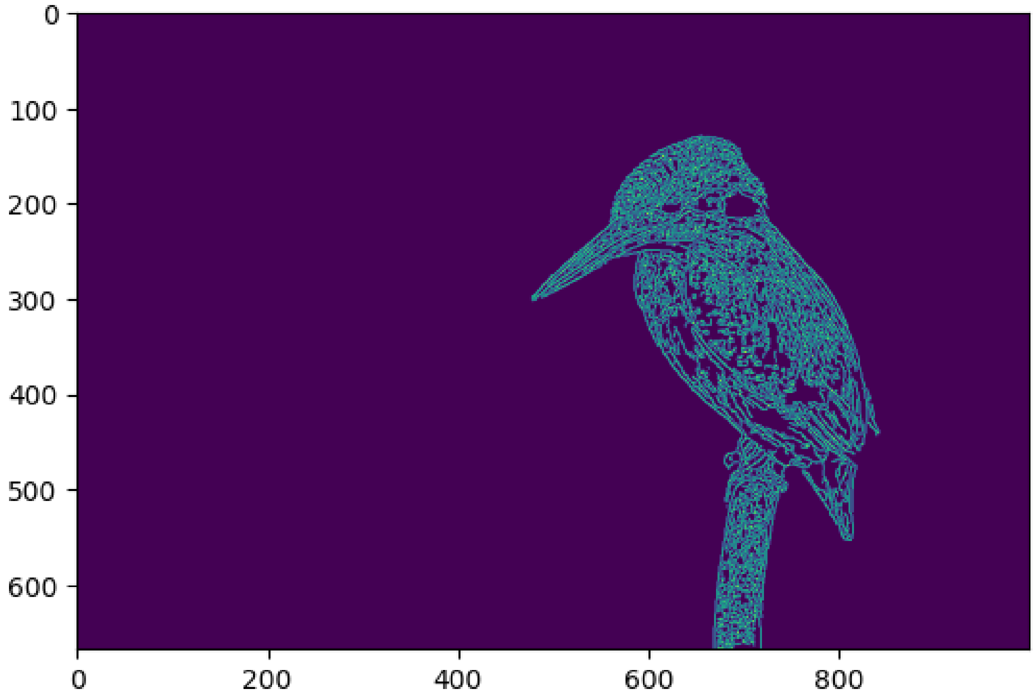


[6]:

```
# perform the canny edge detector to detect image edges
edges = cv2.Canny(gray, threshold1=30, threshold2=100)
```

[8]:

```
plt.imshow(edges)
plt.show()
```



[11]:

```
# Python program to illustrate HoughLine
# method for line detection
import cv2
import numpy as np

# Reading the required image in
# which operations are to be done.
# Make sure that the image is in the same
# directory in which this python program is
img = cv2.imread('/home/nmit/Downloads/line.jpg')

# Convert the img to grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Apply edge detection method on the image
edges = cv2.Canny(gray, 50, 150, apertureSize=3)

# This returns an array of r and theta values
```

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

# The below for loop runs till r and theta values
# are in the range of the 2d array
for r_theta in lines:
    arr = np.array(r_theta[0], dtype=np.float64)
    r, theta = arr
    # Stores the value of cos(theta) in a
    a = np.cos(theta)

    # Stores the value of sin(theta) in b
    b = np.sin(theta)

    # x0 stores the value rcos(theta)
    x0 = a*r

    # y0 stores the value rsin(theta)
    y0 = b*r

    # x1 stores the rounded off value of (rcos(theta)-1000sin(theta))
    x1 = int(x0 + 1000*(-b))

    # y1 stores the rounded off value of (rsin(theta)+1000cos(theta))
    y1 = int(y0 + 1000*(a))

    # x2 stores the rounded off value of (rcos(theta)+1000sin(theta))
    x2 = int(x0 - 1000*(-b))

    # y2 stores the rounded off value of (rsin(theta)-1000cos(theta))
    y2 = int(y0 - 1000*(a))

    # cv2.line draws a line in img from the point(x1,y1) to (x2,y2).
    # (0,0,255) denotes the colour of the line to be
    # drawn. In this case, it is red.
    cv2.line(img, (x1, y1), (x2, y2), (0, 0, 255), 2)

# All the changes made in the input image are finally
# written on a new image houghlines.jpg
cv2.imwrite('linesDetected.jpg', img)
```

[11]:

```
True
```

[12]:

```
import cv2
import numpy as np

# Read image
image = cv2.imread('/home/nmit/Downloads/line.jpg')
```

```
# Convert image to grayscale
gray = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)

# Use canny edge detection
edges = cv2.Canny(gray,50,150,apertureSize=3)

# Apply HoughLinesP method to
# to directly obtain line end points
lines_list =[]
lines = cv2.HoughLinesP(
    edges, # Input edge image
    1, # Distance resolution in pixels
    np.pi/180, # Angle resolution in radians
    threshold=100, # Min number of votes for valid line
    minLineLength=5, # Min allowed length of line
    maxLineGap=10 # Max allowed gap between line for joining them
)

# Iterate over points
for points in lines:
    # Extracted points nested in the list
    x1,y1,x2,y2=points[0]
    # Draw the lines joing the points
    # On the original image
    cv2.line(image,(x1,y1),(x2,y2),(0,255,0),2)
    # Maintain a simples lookup list for points
    lines_list.append([(x1,y1),(x2,y2)])

# Save the result image
cv2.imwrite('detectedLines.png',image)
```

```
[12]: True
```

```
[ ]:
```