Fall 2021, Computer Vision
Homework 2

by 64160010 - Rumeysa ÇELİK

January 2, 2022

Problem Set 2

Problem 1: Design and implement Hough Transform method for finding lines.

You can use OpenCV functions to find edges, such as Canny operator, but you may not use any Hough Transform functions. Convert a color image to grayscale before applying an edge detector.

Specifically, create a Hough transform class including the following functions:

- hough_lines_acc: Takes an edge image as input, and two optional parameters Theta Resolution and Rho Resolution, which indicates the cell size for theta and rho parameters.Returns an accumulator matrix H, and cell locations theta and rho.
- hough_peaks:Returns the peak locations (theta and rho) for a given H, theta, and rho arrays. The function also should take a threshold value t to eliminate weak peaks that are less than t, and another parameter s to return the strongest s peaks.
- hough_lines_draw: Draw color lines corresponding to the peaks found.

Finally, write a script file to get the results on some input images.Compare your results with the results obtained using OpenCV Hough transform.

OpenCV functions were used to find edges like the Canny operator, but no Hough Transform functions were used.
Specifically, a Hough transform class was created that includes "hough_lines_acc, hough_peaks and hough_lines_draw".

INPUT IMAGE:

Code:

```
1
2  # -*- coding: utf-8 -*-
3  """
4  Created on Fri Nov 12 19:11:12 2021
5
6  @author: Rumeysa CELIK
7  """
8
9  import numpy as np
10 import cv2
11
12 #hough_lines_acc:Takes an edge image as input, and two
       optionalparameters ThetaResolutionand RhoResolution, which indicates
        the cell size for theta and rho parameters.Returns an accumulator
       matrix H, and cell locations thetaand rho.
13
14 #hough_peaks:Returns the peak locations (theta and rho) for a given H,
       theta, and rhoarrays. The function also should take a threshold
       value tto eliminate weak peaks that are less than t, and another
       parameter sto return the strongest s peaks.
15
16 #hough_lines_draw:Draw color lines corresponding to the peaks found.
17
18 image = cv2.imread('input/pokut.jpeg')
19 RGB_img = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
20 imageGray = cv2.cvtColor(RGB_img, cv2.COLOR_RGB2GRAY)
21 imageBlur = cv2.GaussianBlur(imageGray, (5, 5), 1.5)
22 canny_edges = cv2.Canny(imageBlur, 50, 50)
23
24
25 def generateHoughLines(img, indices, rhos, thetas):
26     for i in range(len(indices)):
27         rho = rhos[indices[i][0]]
28         theta = thetas[indices[i][1]]
29         a = np.cos(theta)
30         b = np.sin(theta)
31         x0 = a * rho
32         y0 = b * rho
33         x1 = int(x0 + 1000 * -b)
34         y1 = int(y0 + 1000 * a)
35         x2 = int(x0 - 1000 * -b)
36         y2 = int(y0 - 1000 * a)
37         cv2.line(img, (x1, y1), (x2, y2), (255, 0, 0), 2)
38
39
40 def houghLocations(H, num_peaks):
41     size = 3
42     indices = []
43     a = np.copy(H)
44     for i in range(num_peaks):
45         idx = np.argmax(a)
46         AIdx = np.unravel_index(idx, a.shape)
47         indices.append(AIdx)
48
```

```
        idx_y, idx_x = AIdx
        if (idx_x - (size / 2)) < 0:
            min_x = 0
        else:
            min_x = idx_x - (size / 2)
        if (idx_x + (size / 2) + 1) > H.shape[1]:
            max_x = H.shape[1]
        else:
            max_x = idx_x + (size / 2) + 1

        if (idx_y - (size / 2)) < 0:
            min_y = 0
        else:
            min_y = idx_y - (size / 2)
        if (idx_y + (size / 2) + 1) > H.shape[0]:
            max_y = H.shape[0]
        else:
            max_y = idx_y + (size / 2) + 1

        for x in range(int(min_x), int(max_x)):
            for y in range(int(min_y), int(max_y)):
                a[y, x] = 0
                if x == min_x or x == (max_x - 1):
                    H[y, x] = 255
                if y == min_y or y == (max_y - 1):
                    H[y, x] = 255
    return indices, H


def hough(img, rho_resolution=1, theta_resolution=1):
    height, width = img.shape
    img_diagonal = np.ceil(np.sqrt(height ** 2 + width ** 2))
    rhos = np.arange(-img_diagonal, img_diagonal + 1, rho_resolution)
    thetas = np.deg2rad(np.arange(-90, 90, theta_resolution))

    H = np.zeros((len(rhos), len(thetas)), dtype=np.uint64)
    y_idxs, x_idxs = np.nonzero(img)
    for i in range(len(x_idxs)):
        x = x_idxs[i]
        y = y_idxs[i]
        for j in range(len(thetas)):
            rho = int((x * np.cos(thetas[j]) +
                       y * np.sin(thetas[j])) + img_diagonal)
            H[rho, j] += 1
    return H, rhos, thetas


if __name__ == "__main__":
    H, rhos, thetas = hough(canny_edges)
    indices, H = houghLocations(H, 25)

    generateHoughLines(image, indices, rhos, thetas)

    cv2.imshow('Image with Lines', image)
    cv2.imwrite('output.jpeg', image)
```

```
104     cv2.waitKey(0)
```

OUTPUT:

Problem 2: Write a script file to demonstrate your Hough based line detector on live video.

A script was written to show the Hough-based line detector in live video.

Code:

```
1
2  # -*- coding: utf-8 -*-
3  """
4  Created on Sun Nov 14 23:11:12 2021
5
6  @author: Rumeysa CELIK
7  """
8
9  import numpy as np
10 import cv2
11
12 #Write a script file to demonstrate your Hough based line detector on
       live video.
13
14 def generateHoughLines(img, indices, rhos, thetas):
15     for i in range(len(indices)):
16         rho = rhos[indices[i][0]]
17         theta = thetas[indices[i][1]]
18         a = np.cos(theta)
19         b = np.sin(theta)
20         x0 = a * rho
21         y0 = b * rho
22         x1 = int(x0 + 1000 * -b)
23         y1 = int(y0 + 1000 * a)
24         x2 = int(x0 - 1000 * -b)
25         y2 = int(y0 - 1000 * a)
26         cv2.line(img, (x1, y1), (x2, y2), (255, 0, 0), 2)
27
28
29 def hough(img, rho_resolution=1, theta_resolution=1):
30     height, width = img.shape
31     img_diagonal = np.ceil(np.sqrt(height ** 2 + width ** 2))
```

```
    rhos = np.arange(-img_diagonal, img_diagonal + 1, rho_resolution)
    thetas = np.deg2rad(np.arange(-90, 90, theta_resolution))

    H = np.zeros((len(rhos), len(thetas)), dtype=np.uint64)
    y_idxs, x_idxs = np.nonzero(img)
    for i in range(len(x_idxs)):
        x = x_idxs[i]
        y = y_idxs[i]
        for j in range(len(thetas)):
            rho = int((x * np.cos(thetas[j]) +
                       y * np.sin(thetas[j])) + img_diagonal)
            H[rho, j] += 1
    return H, rhos, thetas


def houghLocations(H, num_peaks):
    indices = []
    size = 10
    a = np.copy(H)
    for i in range(num_peaks):
        idx = np.argmax(a)
        AInd = np.unravel_index(idx, a.shape)
        indices.append(AInd)

        idx_y, idx_x = AInd
        if (idx_x - (size / 2)) < 0:
            min_x = 0
        else:
            min_x = idx_x - (size / 2)
        if (idx_x + (size / 2) + 1) > H.shape[1]:
            max_x = H.shape[1]
        else:
            max_x = idx_x + (size / 2) + 1

        if (idx_y - (size / 2)) < 0:
            min_y = 0
        else:
            min_y = idx_y - (size / 2)
        if (idx_y + (size / 2) + 1) > H.shape[0]:
            max_y = H.shape[0]
        else:
            max_y = idx_y + (size / 2) + 1

        for x in range(int(min_x), int(max_x)):
            for y in range(int(min_y), int(max_y)):
                a[y, x] = 0
                if x == min_x or x == (max_x - 1):
                    H[y, x] = 255
                if y == min_y or y == (max_y - 1):
                    H[y, x] = 255
    return indices, H


if __name__ == "__main__":
    video = cv2.VideoCapture(0)
```

```
    a = 0
    while True:
        a = a + 1
        check, frame = video.read()
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        edges = cv2.Canny(frame, 100, 200)
        H, rhos, thetas = hough(edges)
        indices, H = houghLocations(H, 10)
        generateHoughLines(frame, indices, rhos, thetas)
        cv2.imshow("Captured Video", frame)
        cv2.imwrite('VideOutput.jpg', frame)
        key = cv2.waitKey(1)
        if key == ord('w'):
            break

    video.release()
```

OUTPUT: