Lembar Praktikum 9
Mata Kuliah Pengenalan Citra Digital
Semester Genap Tahun Akademik 2018/2019

Nama : Ahmad Maulvi Alfansuri
NIM : G64160081
Topik: Hough Algorithm

Pada tugas kali ini, praktikan diharuskan untuk menjelaskan codingan dari hough transform, baik houghlines dan juga houghcircles.

Diberikan gambar sudo.jpg seperti dibawah

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Untuk script houghlines dapat dilihat pada gambar dibawah ini

```
# import library yang digunakan
import cv2
import numpy as np

# baca gambar sudo.pmg
img = cv2.imread("sudo.png")

# konversi gambar bgr menjadi grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# lakukan edge detection gambar grayscale yang didapatkan
menggunakan canny edge detection
edges = cv2.Canny(gray, 50, 150, apertureSize=3)

# gunakan fungsi houghlines untuk mendapatkan garis.
# penjelasan parameter akan dijelaskan dibawah
```

```
# hasil kembalian adalah list berisi rho dan theta

lines = cv2.HoughLines(edges, 1, np.pi/180, 200)
# @param image 8-bit, single-channel binary source image. The
image may be modified by the function.
# @param rho Distance resolution of the accumulator in pixels.
# @param theta Angle resolution of the accumulator in radians.
# @param threshold Accumulator threshold parameter. Only those
lines are returned that get enough votes

# print lines[0][0]
print len(lines) # hasil kembalian berbentuk [[[rho theta(dalam
rad)]]]
# looping sebanyak jumlah lines yang ditemukan

for x in range(0, len(lines)):
    # looping setiap rho dan theta dari garis tersebut
    for rho, theta in lines[x]:

        # transformasi kembali garis dalam bentuk rho dan theta
        # agar bisa di draw
        #
https://stackoverflow.com/questions/7613955/hough-transform-equ
ation
        a = np.cos(theta)
        b = np.sin(theta)

        # x0 menyimpan nilai r cos theta
        x0 = a*rho
        # y0 menyimpan nilai r sin theta
        y0 = b*rho

        # x1 menyimpan nilai r cos theta + 1000(- sin theta)
        x1 = int(x0 + 1000*(-b))
        # y1 menyimpan nilai r sin theta + 1000( cos theta)
        y1 = int(y0 + 1000*(a))

        # x2 menyimpan nilai r cos theta - 1000(- sin theta)
        x2 = int(x0 - 1000*(-b))

        # x2 menyimpan nilai r cos theta - 1000(- sin theta)
        y2 = int(y0 - 1000*(a))

        # print koordinat penggaris
        print x1, y1
        print x2, y2

        # gambarkan garis berwarna (183,166,20)
        # dengan koordinat (x1, y1) dan (x2, y2)
        # setebal 6 pixel
        cv2.line(img, (x1,y1), (x2,y2), (183,166,20), 6)
```

```
# print lines
print(lines)
# tampilkan hasil edges detection
cv2.imshow('edges', edges)
# tampilkan gambar dengan line detection
cv2.imshow('final', img)
cv2.waitKey(0)
```

Sekarang kita coba fokus kedalam parameter nya

Dikutip dari dokumentasi opencv pada link berikut,

1. https://docs.opencv.org/3.1.0/dd/d1a/group__imgproc__feature.html#ga46b4e588934f6c8dfd509cc6e0e4545a
2. https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_houghlines/py_houghlines.html.

> Now let's see how Hough Transform works for lines. Any line can be represented in these two terms, (ρ, θ). So first it creates a 2D array or accumulator (to hold values of two parameters) and it is set to 0 initially. Let rows denote the ρ and columns denote the θ. Size of array depends on the accuracy you need. Suppose you want the accuracy of angles to be 1 degree, you need 180 columns. For ρ, the maximum distance possible is the diagonal length of the image. So taking one pixel accuracy, number of rows can be diagonal length of the image.

Pada kutipan diatas dijelaskan bagaimana houghtransform bekerja. Dibuat 2D array untuk menyimpan suatu nilai, dimana array tersebut terdiri dari rho dan theta. Nilai theta didapat dengan mengincrement nilai sudut yang diinginkan. Anggap nilai ini sebagai accuracy.

> Now take the second point on the line. Do the same as above. Increment the the values in the cells corresponding to (ρ, θ) you got. This time, the cell (50,90) = 2. What you actually do is voting the (ρ, θ) values. You continue this process for every point on the line. At each point, the cell (50,90) will be incremented or voted up, while other cells may or may not be voted up. This way, at the end, the cell (50,90) will have maximum votes. So if you search the accumulator for maximum votes, you get the value (50,90) which says, there is a line in this image at distance 50 from origin and at angle 90 degrees. It is well shown in below animation (Image Courtesy: Amos Storkey)

Setiap point yang dideteksi di transformasi menjadi bentuk rho dan sudut dengan nilai sudut dalam range yang sudah ditentukan.

Lalu akan dibuat tabel frequency nilai rho dan nilai theta.

Jumlah tabel frequency yang ada diatas threshold lah yang akan dipilih sebagai lines terseleksi yang berhasil di deteksi oleh fungsi houghTransform

Hough Tranform in OpenCV

Everything explained above is encapsulated in the OpenCV function, **cv2.HoughLines()**. It simply returns an array of (ρ, θ) values. ρ is measured in pixels and θ is measured in radians. First parameter, Input image should be a binary image, so apply threshold or use canny edge detection before finding applying hough transform. Second and third parameters are ρ and θ accuracies respectively. Fourth argument is the *threshold*, which means minimum vote it should get for it to be considered as a line. Remember, number of votes depend upon number of points on the line. So it represents the minimum length of line that should be detected.

Berikut penjelasan parameter houghtransform dari ope cv

```
lines = cv2.HoughLines(edges, 1, np.pi/180, 200)
# @param image 8-bit, single-channel binary source image. The
image may be modified by the function.
# @param rho Distance resolution of the accumulator in pixels.
# @param theta Angle resolution of the accumulator in radians.
# @param threshold Accumulator threshold parameter. Only those
lines are returned that get enough votes
```

Terdapat 4 parameter yang harus diset pada fungsi Houghlines.

Parameter pertama adalah gambar yang ingin dideteksi. Gambar harus berbentuk binary image.

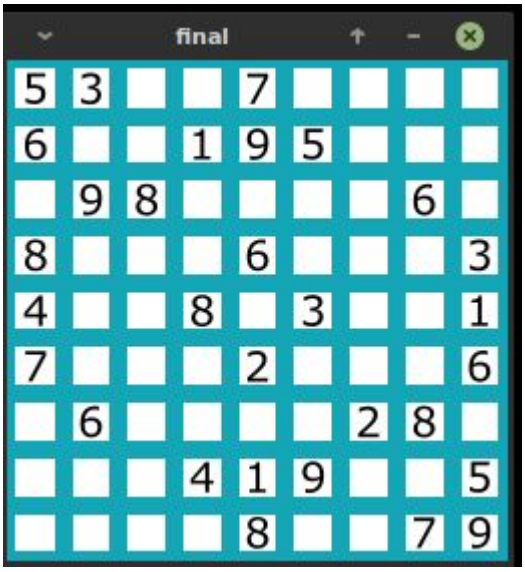
Parameter kedua adalah parameter rho distance resolution. Parameter ini adalah parameter ketelitian dari rho seperti yang sudah dijelaskan sebelumnya. Seberapa keperbedaan antara kedua rho satu sama lain. Kita memilih nilai 1 karena untuk melihat perbedaan antara minimal 1 pixel (satuan terkecil pada gambar).

Parameter ketiga adalah parameter theta Angle resolution. Parameter ini adalah akurasi pada theta, seberapa nilai increment yang akan digunakan untuk membuat array. Semakin kecil nilai ini maka ketelitian akurasi akan semakin meningkat. Diset nilai pi / 180 alias 1 derajat, agar dibuat tabel sebanyak 180 kolom (sudut)

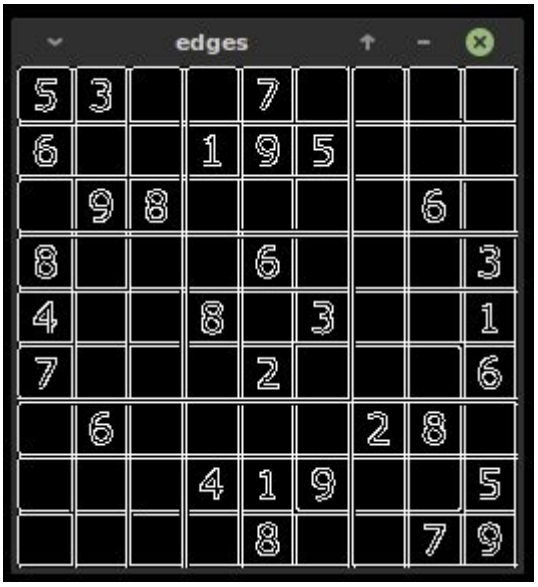
Parameter keempat adalah parameter threshold. Nilai threshold ini akan menjadi penentu mana saja garis yang berhasil dideteksi dengan melihat table frequency yang berada diatas ambang threshold ini.

Berikut adalah hasilnya

Hasil deteksi garis



Hasil canny edge detection



Untuk mendeteksi lingkaran opencv mempunyai fungsi HoughCircles. Berikut adalah contoh untuk mendeteksi lingkaran pada gambar coin.png

Gambar coin.png

Script untuk mendeteksi lingkaran menggunakan HoughCircles.

```
import cv2
import numpy as np
# import library opencv dan numpy

img = cv2.imread('coin.jpg')
# baca gambar

img = cv2.medianBlur(img, 5)
# lakukan medianblur pada image dengan kernel 5 x 5

cv2.imshow('detected circles', img)
# tampilkan gambar sebelum dideteksi

cimg = cv2.cvtColor(img, cv2.COLOR_GRAY2BGR)
# ubah gambar menjadi grayscale

circles = cv2.HoughCircles(img, cv2.cv.CV_HOUGH_GRADIENT, 1,
20, param1=290, param2=55, minRadius=0, maxRadius=0)
# fungsi HoughCircles pada opencv.
# digunakan untuk mendeteksi lingkaran pada sebuah image
# hasil kembalian adalah list berisi tuples
# tuples terdiri dari koordinat x , koordinat y, dan juga
radius dari lingkarang
# yang berhasil dideteksi

print circles # nyoba print lingkaran
```

```
# loop semua tuples
for i in circles[0, :]:
    # draw lingkaran pada cimg
    # dengan koordinat x, y = i[0], i[1]
    # dengan radius i[2]
    # dengan warna (0, 255, 0)
    # dengan tebal garis 2 pixel
    cv2.circle(cimg, (i[0], i[1]), i[2], (0, 255, 0), 2)

# tampilkan gambar
cv2.imshow('detected circles', cimg)

cv2.waitKey(0)
```

Sekarang kita mencoba fokus untuk fungsi di HoughCircles.

Melalui Referensi pada link berikut

1. https://docs.opencv.org/2.4/modules/imgproc/doc/feature_detection.html?highlight=houghcircles

Didapat penjelasan parameter sebagai berikut

Python: `cv2.HoughCircles(image, method, dp, minDist[, circles[, param1[, param2[, minRadius[, maxRadius]]]]]) → circles`

Parameters:
- **image** – 8-bit, single-channel, grayscale input image.
- **circles** – Output vector of found circles. Each vector is encoded as a 3-element floating-point vector $(x, y, radius)$.
- **circle_storage** – In C function this is a memory storage that will contain the output sequence of found circles.
- **method** – Detection method to use. Currently, the only implemented method is `CV_HOUGH_GRADIENT`, which is basically *21HT*, described in [Yuen90].
- **dp** – Inverse ratio of the accumulator resolution to the image resolution. For example, if `dp=1`, the accumulator has the same resolution as the input image. If `dp=2`, the accumulator has half as big width and height.
- **minDist** – Minimum distance between the centers of the detected circles. If the parameter is too small, multiple neighbor circles may be falsely detected in addition to a true one. If it is too large, some circles may be missed.
- **param1** – First method-specific parameter. In case of `CV_HOUGH_GRADIENT`, it is the higher threshold of the two passed to the `Canny()` edge detector (the lower one is twice smaller).
- **param2** – Second method-specific parameter. In case of `CV_HOUGH_GRADIENT`, it is the accumulator threshold for the circle centers at the detection stage. The smaller it is, the more false circles may be detected. Circles, corresponding to the larger accumulator values, will be returned first.
- **minRadius** – Minimum circle radius.
- **maxRadius** – Maximum circle radius.

Penjelasan lebih detail akan dibahas dibawah ini

Snippet code

```
circles = cv2.HoughCircles(
    img,
    cv2.cv.CV_HOUGH_GRADIENT, # metode deteksi yang digunakan
    1,  # inverse rasio akumulator. (ukuran tabel deteksi)
        # nilai 1 berarti ukuran akumulator sama dengan ukuran
pixel
    20, # minDist between center of circle
        # ukuran minimal antara kedua titik dari pusat
lingkaran
    # pada parameter dibawah ini. Digunakan untuk deteksi
sesuai dengan
    # method yang digunakan
    # parameter 1 mengacu pada parameter yang digunakan di
canny edge detection
    # karena gambar harus dirubah dahulu menjadi biner dengan
canny
    param1=290,
    # parameter 2 adalah threshold yang harus dicapai oleh
Frequensi table
    # Untuk menentukan apakah objek yang dideteksi adalah
lingkaran atau tidak
    param2=55,
    minRadius=0,# default
    maxRadius=0 # default
    )
```

Parameter pertama adalah gambar yang ingin dideteksi

Parameter kedua adalah metode deteksi yang digunakan. Untuk saat ini secara default opencv menyediakan CV_HOUGH_GRADIENT untuk mendeteksi.

Parameter ketiga adalah ukuran akumulator dibagi dengan ukuran image. Dipilih 1 karena akumulator (tabel deteksi) dibuat seukuran dengan image sesungguhnya

Parameter keempat adalah jarak minimal antara kedua pusat lingkaran. Jika berdekatan lingkaran dapat saling bertumpang tindih satu sama lain. Jika nilai tinggi beberapa lingkaran bisa dapat tidak terdeteksi

Parameter kelima adalah parameter yang digunakan untuk mengubah gambar ke edge detection terlebih dahulu menggunakan canny. Berbeda dengan sebelumnya dimana input gambar dihoughlines detection harus dirubah terlebih dahulu, pada fungsi houghcircles, fungsi sudah memasukkan metode pengubahan edgedetection dalam fungsi houghcircles. Kita tinggal mengubah parameter dari canny

Parameter keenam adalah parameter threshold, untuk menentukan apakah objek yang dideteksi adalah lingkaran atau tidak. Frekuensi objek harus melebihi batas threshold untuk mendefiniskan objek tersebut adalah lingkaran

Parameter ketujuh adalah minRadius. Diset 0, default deteksi semua lingkaran dengan radius berapapun

Parameter kedelapan adalah maxRadius. Diset 0, default deteksi semua lingkaran dengan radius berapapun

Hasil gambar