Nama : Fadia Ramadhana
NIM : G64170026

LKP 11
PENGOLAHAN CITRA DIGITAL

Straight Line Hough Transform

1. Import *Library*

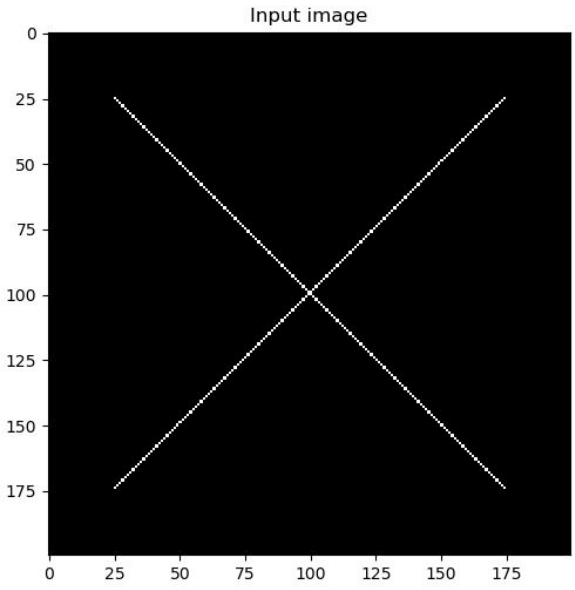
```
5   import numpy as np
6
7   from skimage.transform import hough_line, hough_line_peaks
8   from skimage.feature import canny
9   from skimage import data
10
11  import matplotlib.pyplot as plt
12  from matplotlib import cm
```

Library yang dibutuhkan untuk melakukan hough transform ini adalah ***numpy***, ***skimage***, dan ***matplotlib.*** Dari library tersebut, fungsi utama yang akan digunakan pada line hough transform ini adalah **hough_line()** dan **hough_line_peaks()**.

2. Konstruksi *Test Image*

```
14  # Construct test image
15  image = np.zeros((200, 200))
16  # Pixel index (numpy array dari 25 sampai 175, length = 151)
17  idx = np.arange(25, 175)
18  # Line colour
19  image[idx[::-1], idx] = 255
20  image[idx, idx] = 255
```

Citra yang digunakan untuk melakukan line hough transform merupakan citra yang dikonstruksi sendiri. Untuk mengkonstruksi citra, pertama dapat dilakukan dengan membuat array zeros dengan dimensi 200x200. Kemudian index dari pixel citra akan diset dari 25 sampai 175 terlebih dahulu dengan menggunakan **np.arrange()**. Kemudian dari index yang telah diset sebelumnya, akan dibentuk dua garis berwarna putih. Citra yang diperoleh dari tahapan ini adalah sebagai berikut :



Gambar 1.Hasil konstruksi citra input

3. Line Hough Transform

Secara umum, garis lurus y = mx + b dapat direpresentasikan sebagai titik (b, m) dalam ruang parameter. Untuk alasan komputasi, Duda dan Hart mengusulkan penggunaan bentuk normal Hesse ρ = xcosθ + ysinθ, di mana ρ adalah jarak dari titik asal ke titik terdekat pada garis lurus, dan θ adalah sudut antara sumbu x dan garis yang menghubungkan titik asal dengan titik terdekat itu.

Hough Transform menggunakan mekanisme *voting* sehingga setelah semua piksel gambar diproses, dapat ditentukan piksel mana yang merupakan *background* dari citra dan yang merupakan titik aktual pada garis. Pada python, untuk melakukan hough transform dapat dilakukan dengan menggunakan fungsi yang disediakan oleh library **skimage** yaitu **hough_line()**. Parameter dari fungsi **hough_line()** adalah image yang akan dideteksi dan theta yang merupakan himpunan sudut dari range yang telah didefinisikan yang kemudian akan digunakan untuk mencari nilai rho. Nilai sudut biasanya memiliki range (rentang) nilai [−90◦, 90◦]. Pada kesempatan ini, nilai sudut tersebut di set [-50◦, 50◦]. Untuk lebih jelasnya dapat dilihat pada kode program dibawah ini.

```
24  # set tested angle from -50◦ to 50◦
25  tested_angles = np.linspace(float(-np.pi / 2.0), float(np.pi / 2.0), 360)
26  # get the accumulator, theta, and rho
27  h, theta, d = hough_line(image, theta=tested_angles)
```

Fungsi **hough_line()** mengembalikan tiga buah nilai yaitu akumulator, theta, dan ρ (rho) (dalam hal ini max votes). Gambar ditransformasikan ke hough space dengan menghitung ρ dengan titik di setiap sudut dari −50◦ hingga 50◦ (sudut negatif berlawanan arah jarum jam mulai

secara horizontal dari asal dan sudut positif searah jarum jam). Titik-titik di hough space membuat kurva sinusoidal. Dengan dua parameter ini kita akan melakukan "voting" dalam matriks-2D, itu artinya, kita menambah indeks matriks[theta][rho] dengan satu atau dalam hal ini melakukan increment. Jadi setiap titik yang menghasilkan θ (theta) dan ρ (rho) yang sama akan memberikan satu suara untuk indeks di dalam *voting matrix*. *Voting matrix* disebut juga dengan akumulator (h) yang berada salam hough space. Untuk mencari nilai ρ (rho) dan θ (theta) dengan nilai max votes (peak) dapat dilakukan dengan kode program seperti dibawah ini :

```
33  # Easiest peak finding based on max votes
34  ind = np.argmax(h)
35  rho = d[ind // h.shape[1]]
36  thetas = theta[ind % h.shape[1]]
37  print("rho = {0:.2f}, theta = {1:.0f}".format(rho, np.rad2deg(thetas)))
```

Dari kode program yang telah dijalankan tersebut, diperoleh nilai ρ (rho) = 141.75 dan θ (theta) = 45◦, 135◦.

Selanjutnya, untuk menampilkan hasil dari hough transform pada input image, dapat dilakukan dengan menggunakan fungsi **imshow()** dari library matplotlib dan menerapkanya pada logaritma natural dari nilai akumulator (h) + 1. Pada parameter fungsi, terdapat ***extent*** yang mendefinisikan batas kiri dan kanan, dan batas bawah atau dalam hal ini extent = [horizontal_min, horizontal_max, vertical_min, vertical_max]. Kemudian, terdapat parameter ***aspect*** yang diset 1/1.5. Mengatur rasio ***aspect*** dalam program sangat penting untuk menyesuaikan dimensi grafik tanpa mengubah konten.

```
47  # extent = [horizontal_min, horizontal_max, vertical_min, vertical_max]
48  ax[1].imshow(np.log(1 + h),
49               extent=[np.rad2deg(theta[-1]), np.rad2deg(theta[0]), d[-1], d[0]],
50               cmap=cm.gray, aspect=1/1.5)
51  ax[1].set_title('Hough transform')
52  ax[1].set_xlabel('Angles (degrees)')
53  ax[1].set_ylabel('Distance (pixels)')
54  ax[1].axis('image')
```

4. Mengubah maxima di Hough Space kembali ke koordinat x / y

Selanjutnya, akan digunakan fungsi **hough_line_peaks()** dimana fungsi ini mengembalikan puncak (peaks) dalam transformasi hough yang berbentuk *angle*. Fungsi bekerja dengan cara mengidentifikasi garis paling menonjol yang dipisahkan oleh sudut dan jarak tertentu dalam transformasi hough. Non-maximum suppression dengan ukuran yang berbeda diterapkan secara terpisah dalam dimensi pertama (jarak atau d) dan dimensi kedua (sudut atau theta) hough space untuk mengidentifikasi puncak (peaks). Akan digunakan for loop lain untuk menerapkan nilai angle dan distance ke proses selanjutnya.

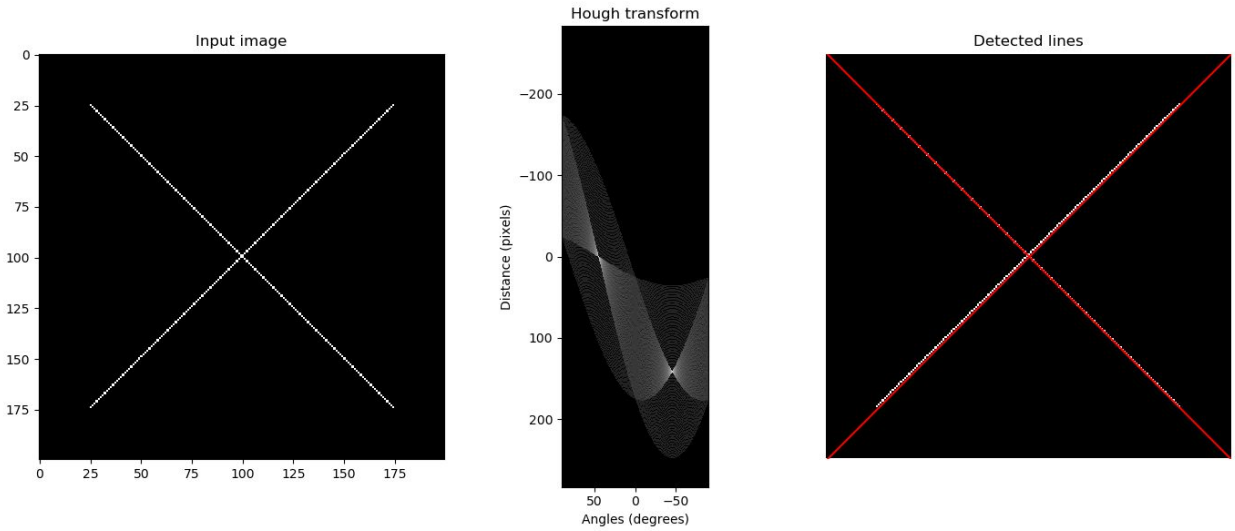
Setelah menemukan garis-garis pada gambar, seringkali diperlukan untuk menampilkan garis garis-garis tersebut pada gambar asli. Ini juga berguna untuk melihat apakah kita benar-benar mendeteksi sesuatu yang menarik. Mengingat kombinasi ρ (rho) dan θ (theta) mana yang menghasilkan nilai-nilai menarik sudah diketahui, garis lainnya dapat diperoleh kembali dari ρ (rho) dan θ (theta) tersebut. Garis yang dicari dijelaskan melalui **ρ = xcosθ + ysinθ**, tetapi persamaan tersebut memiliki dua nilai yang tidak diketahui yaitu: x dan y. Jadi untuk mengetahui nilai x diperlukan nilai y yang diketahui dan sebaliknya.

Jika diasumsikan bahwa garis yang dicari panjangnya tak terhingga, kita tahu bahwa pada titik tertentu, garis tersebut akan berpotongan dengan sumbu x dari citra input di titik 0, dan akan berpotongan dengan sumbu y dari citra input di titik 0. Pada kode program, nilai dari kedua y (y0 dan y1) diperoleh dengan rumus : **(dist - origin * np.cos(angle)) / np.sin(angle)**. Dimana **dist** adalah nilai ρ (rho) dan **origin** adalah nilai di sumbu x pada citra input.

Berikut adalah baris kode program :

```
56    ax[2].imshow(image, cmap=cm.gray)
57    origin = np.array((0, image.shape[1]))
58    for _, angle, dist in zip(*hough_line_peaks(h, theta, d)):
59        y0, y1 = (dist - origin * np.cos(angle)) / np.sin(angle)
60        ax[2].plot(origin, (y0, y1), '-r')
61    ax[2].set_xlim(origin)
62    ax[2].set_ylim((image.shape[0], 0))
63    ax[2].set_axis_off()
64    ax[2].set_title('Detected lines')
65
66    plt.tight_layout()
67    plt.show()
```

Dari serangkaian proses yang dilakukan, selanjutnya adalah melakukan plotting citra output dari masing-masing proses dan diperoleh hasil akhir sebagai berikut :



Gambar 2. Input image (a), hough transform (b), dan garis putih hasil inisialisasi dengan garis merah hasil komputasi sebagai overlay (c)

Dari Gambar 2(b) dapat dilihat local maxima dalam histogram yang dihasilkan menunjukkan parameter garis yang paling mungkin. Dalam citra input, maxima terjadi pada 45◦ dan 135◦, sesuai dengan sudut vektor normal setiap garis. Kedua sudut (angle) tersebut pun diketahui masing-masing nilai distance (rho) nya. Kemudian dari sudut (angle) dan distance (rho) yang diperoleh tersebut, kedua nilai digunakan untuk menampilkan garis-garis yang terdeteksi (garis merah) sebagai overlay pada citra input asli (garis putih) yang telah dilakukan pada proses ke-4 dan hasilnya dapat dilihat seperti yang tampak pada Gambar 2(c).

Probabilistic Hough Transform

Untuk mendapatkan akurasi yang lebih baik, kompromi dengan bidang komputasi tentu diperlukan. Dibutuhkan banyak daya komputasi untuk mengiterasi semua titik dan menambahkan *vote*. Untuk mengurangi perhitungan ini, para peneliti datang dengan beberapa teknik probabilistik yang akan meningkatkan kecepatan komputasi tanpa harus kehilangan banyak akurasi. Hough Transform dianggap probabilistik jika menggunakan pengambilan sampel acak dari titik tepi. Algoritma ini dapat dibagi berdasarkan bagaimana mereka memetakan ruang gambar ke ruang parameter. Salah satu metode probabilistic termudah adalah untuk memilih m titik tepi dari set M titik tepi. Kompleksitas tahap *voting* berkurang dari O(M.NΘ) menjadi O(m.NΘ). Ini bekerja karena subset acak M akan cukup mewakili titik tepi beserta noise dan distorsi di sekitarnya. Nilai m yang lebih kecil akan menghasilkan komputasi cepat tetapi akurasi lebih rendah. Jadi nilai m harus dipilih dengan tepat sehubungan dengan M.

1. Import *Library*

```
5   import numpy as np
6
7   from skimage.transform import hough_line, hough_line_peaks
8   from skimage.feature import canny
9   from skimage import data
10
11  import matplotlib.pyplot as plt
12  from matplotlib import cm
13
14  from skimage.transform import probabilistic_hough_line
```

Library yang dibutuhkan untuk melakukan hough transform ini adalah ***numpy***, ***skimage***, dan ***matplotlib*.** Dari library tersebut, fungsi utama yang akan digunakan pada line hough transform ini adalah **canny()** dan **probabilistic_hough_line()**.

2. Load Image dan Edge Detection

Citra yang akan diterapkan *probabilistic hough transform* adalah image camera yang tersedia pada library skimage. Setelah melakukan load image, langkah yang selanjutnya dilakukan adalah melakukan deteksi tepi pada image camera tersebut. Deteksi tepi dapat dilakukan dengan berbagai algoritma seperti menggunakan canny, sobel, atau thresholding adaptif. Pada kesempatan kali ini, algoritma yang digunakan adalah canny edge detection. Gambar biner / abu-abu yang dihasilkan akan memiliki sekumpulan nilai 0 yang mengindikasikan non-edge dan sekumpulan nilai lebih dari atau sama dengan 1 yang mengindikasikan edge. Pada **canny(),** selain image yang akan dideteksi tepi nya, di set beberapa parameter sebagai berikut : **sigma** = 2**, low threshold** = 1**,** dan **high threshold** = 25

```
78  # Line finding using the Probabilistic Hough Transform
79  image = data.camera()
80  edges = canny(image, 2, 1, 25)
```

Berikut adalah citra yang dihasilkan dua baris kode program diatas :

Gambar 3. Image camera (a), Citra hasil canny edge detection (b)

Citra yang telah melalui proses edge detection seperti yang terlihat pada gambar 3(b) selanjutnya akan digunakan sebagai input image pada probabilistic hough transform.

3. Probabilistic Hough Transform

Untuk menerapkan probabilistic hough transform, dapat dilakukan dengan menggunakan fungsi **probabilistic_hough_line()** yang terdapat pada library skimage. Fungsi tersebut diterapkan pada citra yang telah dideteksi tepinya. Terdapat beberapa parameter yang dapat diatur, yaitu :

1. **threshold** : ambang batas dan nilai yang diatur untuk parameter ini adalah 10.
2. **line_length** : panjang minimum yang diterima dari garis yang terdeteksi. Semakin besar nilainya maka semakin panjang pula garis yang diekstrak. Nilai yang diatur untuk parameter ini adalah 5.
3. **line_gap :** celah maksimum antara piksel untuk tetap membentuk garis. Semakin besar nilainya, semakin agresif garis putus-putus akan digabungkan. Nilai yang diatur untuk parameter ini adalah 3.

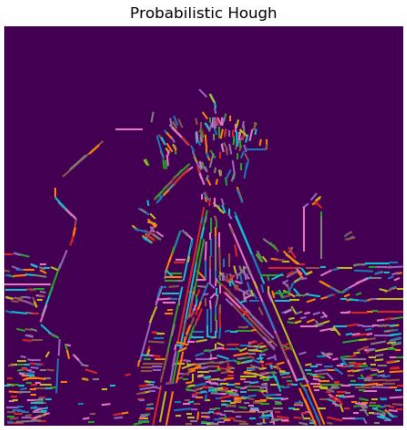
```
81    lines = probabilistic_hough_line(edges, threshold=10, line_length=5,
82                                     line_gap=3)
```

Fungsi mengembalikan list berisikan garis-garis yang teridentifikasi, garis dalam format ((x0, y0), (x1, y0)), yang menunjukkan awal dan akhir garis.

Untuk melakukan plotting terhadap hasil dari probabilistic hough transform dapat dilakukan sebagai berikut

```
94    ax[2].imshow(edges * 0)
95    for line in lines:
96        p0, p1 = line
97        ax[2].plot((p0[0], p1[0]), (p0[1], p1[1]))
98    ax[2].set_xlim((0, image.shape[1]))
99    ax[2].set_ylim((image.shape[0], 0))
100   ax[2].set_title('Probabilistic Hough')
```

Hasil akhir yang diperoleh dari keseluruhan proses adalah sebagai berikut



Gambar 4. Citra hasil *probabilistic hough transform*

Probabilistic Hough Transform memberi kita lebih banyak fleksibilitas dan kemungkinan untuk mendeteksi tepi yang lebih akurat jika kita menghabiskan lebih banyak waktu untuk menyesuaikan hasilnya. Ada banyak parameter - rho, theta, threshold, panjang garis minimum dan celah garis maksimum.

Di sini kita melihat lebih banyak garis, tetapi ujung-ujungnya jauh lebih pendek. Jika banyak garis seperti itu mengikuti arah yang sama, kita dapat mengasumsikan bahwa ada sisi dengan probabilitas yang relatif tinggi. Probabilistic Hough Transform sangat baik digunakan untuk mendeteksi garis pada citra berisi objek yang disusun oleh garis-garis yang tidak semuanya lurus.

Circular Hough Transform

Circular Hough Transform (CHT) adalah teknik ekstraksi fitur dasar yang digunakan dalam pemrosesan gambar digital untuk mendeteksi lingkaran dalam gambar yang tidak sempurna. Kandidat lingkaran diproduksi dengan "voting" di ruang parameter Hough dan kemudian memilih maxima lokal dalam matriks akumulator. Ini adalah salah satu spesialisasi transformasi Hough.
Dalam bentuk 2D, lingkaran dapat dideskripsikan sebagai berikut

$$(x-a)^2 + (y-b)^2 = r^2$$

Lingkaran dapat digambarkan sepenuhnya dengan tiga informasi: pusat (a, b) dan jari-jari. (Pusat terdiri dari dua bagian, maka totalnya tiga)

x = a + Rcosθ

y = b + Rsinθ

Ketika θ bervariasi dari 0 hingga 360, lingkaran yang lengkap dihasilkan dari jari-jari R.

Jadi dengan circular Hough Transform, diharapkan akan menemukan nilai (x, y, R) yang mengindikasikan lingkaran dengan kemungkinan tertinggi dalam gambar. Artinya, kita ingin menemukan tiga parameter tersebut. Dengan demikian, ruang parameternya adalah 3D yang dimana artinya segalanya bisa menjadi jelek jika tidak melangkah perlahan.

Pada circular hough transform, proses dimulai dengan asumsi bahwa kita sedang mencari lingkaran dengan jari-jari tertentu, yang berarti, nilai dari R diketahui. Jadi, setiap titik dalam ruang xy akan setara dengan lingkaran di ruang ab (dimana R bukan parameter). Ini karena saat menata ulang persamaan, diperoleh :

$a = x_1 - R\cos\theta$

$b = y_1 - R\sin\theta$

untuk titik tertentu (x1, y1). Dan θ menelusuri dari 0 hingga 360 derajat.

1. Import *Library*

```
7   import numpy as np
8   import matplotlib.pyplot as plt
9
10  from skimage import data, color
11  from skimage.transform import hough_circle, hough_circle_peaks
12  from skimage.feature import canny
13  from skimage.draw import circle_perimeter
14  from skimage.util import img_as_ubyte
```

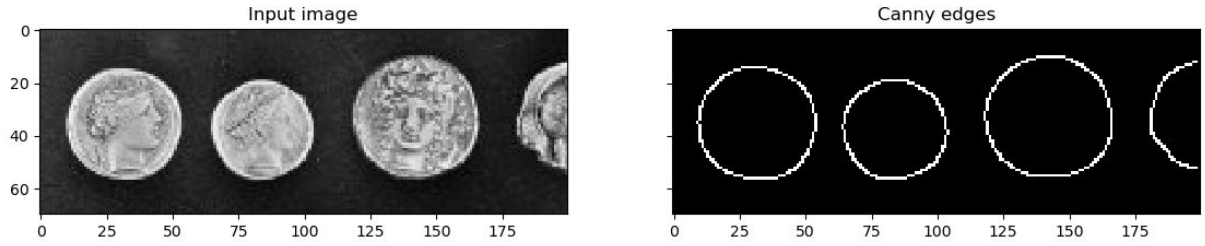
Library yang dibutuhkan untuk melakukan hough transform ini adalah ***numpy***, ***skimage***, dan ***matplotlib*.** Dari library tersebut akan dilakukan import terhadap fungsi-fungsi yang akan membantu dalam proses circular hough transform.

2. Load image dan Edge Detection

Citra yang akan diterapkan *circular hough transform* adalah image coins yang tersedia pada library skimage. Pada image coins dilakukan proses cropping untuk mendapatkan beberapa coin saja. Setelah melakukan load image, langkah yang selanjutnya dilakukan adalah melakukan deteksi tepi pada image coins tersebut. Deteksi tepi dapat dilakukan dengan berbagai algoritma seperti menggunakan canny, sobel, atau thresholding adaptif. Pada kesempatan kali ini, algoritma yang digunakan adalah canny edge detection. Gambar biner/abu-abu yang dihasilkan akan memiliki sekumpulan nilai 0 yang mengindikasikan non-edge dan sekumpulan nilai lebih dari atau sama dengan 1 yang mengindikasikan edge. Pada **canny(),** selain image yang akan dideteksi tepi nya, di set beberapa parameter sebagai berikut : **sigma** = 3**, low threshold** = 10**,** dan **high threshold** = 50

```
# Load picture and detect edges
image = img_as_ubyte(data.coins()[160:230, 70:270])
edges = canny(image, sigma=3, low_threshold=10, high_threshold=50)
```

Berikut adalah citra yang dihasilkan dua baris kode program diatas :



Gambar 5. Image coins(a), Citra hasil canny edge detection(b)

Citra yang telah melalui proses edge detection seperti yang terlihat pada gambar 5(b) selanjutnya akan digunakan sebagai input image pada circular hough transform.

3. Circular Hough Transform

Untuk melakukan circular hough transform, dapat dilakukan dengan menggunakan fungsi **hough_circle()** yang terdapat pada library skimage. Pada baris kode program dibawah, fungsi tersebut memiliki dua buah parameter, yaitu **image** dan **radius**. Parameter image adalah citra input dengan nilai non-zero yang mewakili tepi dari objek pada citra, sedangkan parameter radius adalah nilai skalar atau kumpulan skalar radii untuk menghitung transformasi Hough. Nilai float pada radius akan dikonversi menjadi bilangan integer. Selain itu, parameter radius memiliki beberapa parameter, yaitu **minimum radius**, **maksimum radius**, dan **steps**. Pada kode program, nilai dari minimum radius adalah 20 dan nilai dari maksimum radius adalah 35 dengan jumlah steps adalah 2. Ini artinya, dari mulai radius 20 hingga radius 35 akan dilakukan iterasi dengan langkah 2. Diantara setiap potongan 2, akan dijalankan proses circular hough transform atau dalam hal ini akan mendeteksi lingkaran dan memperbarui gambar dengan kontur yang diperoleh.

```
# Detect two radii
hough_radii = np.arange(20, 35, 2)
hough_res = hough_circle(edges, hough_radii)
```

Fungsi mengembalikan nilai berupa 3D ndarray (indeks jari-jari, (M + 2R, N + 2R) ndarray) yaitu akumulator pada hough transform space untuk setiap radius

4. Memilih lingkaran-lingkaran yang terdeteksi

Lingkaran yang terdeteksi selanjutnya akan dipilih yang paling menonjol dengan menggunakan fungsi **hough_circle_peaks()** dimana fungsi ini bekerja dengan cara mengidentifikasi lingkaran paling menonjol yang dipisahkan oleh jarak tertentu dalam hough space yang diberikan. Non-maximum suppression dengan ukuran yang berbeda diterapkan secara terpisah dalam dimensi pertama dan kedua dari hough space untuk mengidentifikasi puncak. Untuk lingkaran

dengan jari-jari berbeda tetapi jaraknya dekat, hanya lingkaran dengan puncak tertinggi yang akan dipertahankan. Pada baris kode program dibawah, fungsi memiliki tiga buah parameter yaitu, **hough space** yang berasal dari fungsi **hough_circle()**, nilai **radius (Radii)** yang bersesuaian dengan hough space, dan **total maksimum** dari puncak yang di set 3.

```
35  # Select the most prominent 3 circles
36  accums, cx, cy, radii = hough_circle_peaks(hough_res, hough_radii,
37                                             total_num_peaks=3)
```

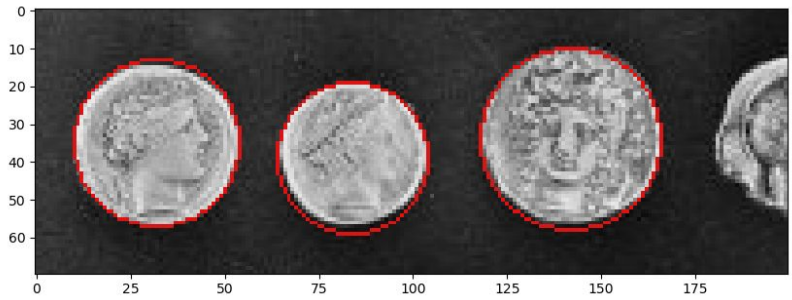
Fungsi mengembalikan nilai puncak (peaks) pada hough space, nilai koordinat pusat x dan y pada image, dan nilai radius (Radii). Masing-masing dari nilai ini merupakan tuple of array.

5. Menggambar kembali lingkaran yang terdeteksi diatas Input image

Untuk menggambar kontur lingkaran yang telah dideteksi, dapat dilakukan dengan mencari perimeter dari objek lingkaran bersangkutan. Pada library **skimage.draw** tersedia fungsi **circle_perimeter()** yang akan menghasilkan koordinat perimeter dari lingkaran. Parameter dari fungsi ini adalah koordinat pusat dari lingkaran (yaitu x dan y), radius lingkaran, dan shape dari image sebagai output.

```
39  # Draw them
40  fig, ax = plt.subplots(ncols=1, nrows=1, figsize=(10, 4))
41  image = color.gray2rgb(image)
42  for center_y, center_x, radius in zip(cy, cx, radii):
43      circy, circx = circle_perimeter(center_y, center_x, radius,
44                                      shape=image.shape)
45      image[circy, circx] = (220, 20, 20)
```

Fungsi akan mengembalikan indeks piksel pada perimeter lingkaran. Kemudian nilai dari warna kontur dapat diatur sesuka hati, namun pada kode program ini, warna kontur lingkaran di set dengan komposisi R = 220, G = 20, dan B = 20. Hasil akhir dari keseluruhan proses adalah sebagai berikut :



Gambar 6. Kontur lingkaran dari Image coins yang terdeteksi melalui circular hough transform

Lingkaran-lingkaran yang terdeteksi pada gambar coins di atas merupakan lingkaran yang memiliki rentang radius antara 20-35 derajat. Lingkaran yang terdeteksi ada sebanyak 3 seperti yang sudah diatur sebelumnya pada total peaks. Koin yang berada di bagian paling kanan tidak terdeteksi sebagai lingkaran karena memang bentuknya tidak menggambarkan lingkaran

sempurna. Lingkaran jari-jari R merupakan kontur yang digambarkan di sekelilingnya. Dan kemudian, “voting” diberikan pada piksel lingkaran-lingkaran ini. Untuk setiap (x, y) dari empat titik dalam gambar asli, dapat menentukan lingkaran di ruang parameter Hough yang berpusat di (x, y) dengan jari-jari r. Matriks akumulator digunakan untuk melacak titik persimpangan. Di ruang parameter, jumlah “voting point” di mana lingkaran yang lewat akan bertambah satu. Kemudian titik local-maxima (titik tengah) dapat ditemukan. Posisi (a, b) dari maxima akan menjadi pusat dari lingkaran asli.

Elliptical Hough Transform

Deteksi kurva elips atau fragmen dari kurva tersebut merupakan tugas penting dalam computer vision. Bentuk ini sering terjadi di banyak jenis pemandangan. Objek buatan bahkan sering memiliki profil melingkar yang, jika dilihat secara miring, memproyeksikan bentuk elips dalam gambar 2D. Pendeteksian elips memerlukan 5 buah parameter. Setiap parameter memiliki kisaran nilai yang besar dan jika diperlukan, ketelitian representasi dari akurasi yang tinggi di luar rentang ini, sehingga source menjadi tidak terlalu besar. Ini lebih baik daripada harus menggunakan tiga buah lingkaran.

Persamaan umum dari sebuah elips adalah,

$$X^2 + B'Y^2 + 2D'XY + 2E'X + 2G'Y + C' = 0$$

di mana B', D', E', G' dan C’ adalah koefisien konstan yang telah dinormalisasi dan berhubungan dengan koefisien X^2. Jika kita gunakan metode HT standar dengan persamaan ini, kita harus membangun ruang parameter lima dimensi dan jika setiap rentang parameter dibagi menjadi interval maka ruang akumulator membutuhkan lokasi penyimpanan a^5.

1. Import *Library*

```
58  import matplotlib.pyplot as plt
59
60  from skimage import data, color, img_as_ubyte
61  from skimage.feature import canny
62  from skimage.transform import hough_ellipse
63  from skimage.draw import ellipse_perimeter
```

Library yang dibutuhkan untuk melakukan hough transform ini adalah ***numpy***, ***skimage***, dan ***matplotlib*.** Dari library tersebut akan dilakukan import terhadap fungsi-fungsi yang akan membantu dalam proses circular hough transform.

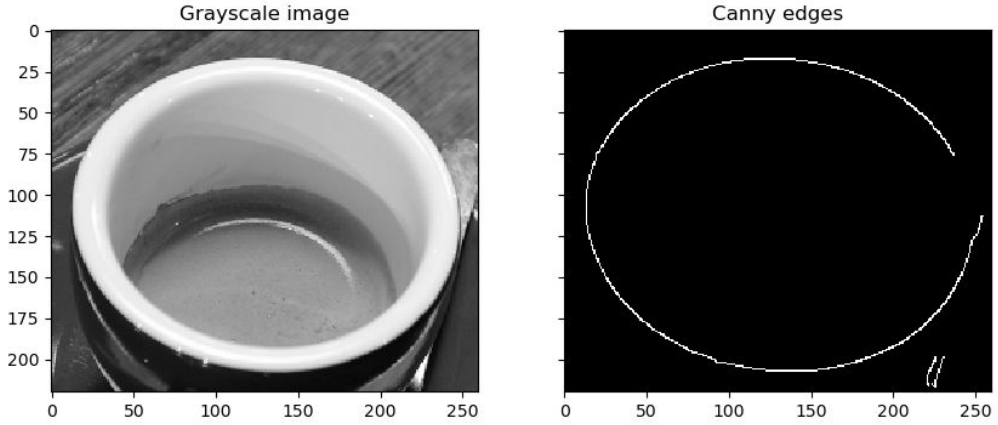
2. Load image, Convert colorspace, dan Edge Detection

Citra yang akan diterapkan *elliptical hough transform* adalah image coffee cup yang tersedia pada library skimage. Pada image coffee cup dilakukan proses cropping terlebih dahulu kemudian hasil dari cropping ini akan diubah colorspace nya dari RGB menjadi Grayscale. Kemudian, langkah yang selanjutnya dilakukan adalah melakukan deteksi tepi pada image

coffee cup tersebut. Deteksi tepi dapat dilakukan dengan berbagai algoritma seperti menggunakan canny, sobel, atau thresholding adaptif. Pada kesempatan kali ini, algoritma yang digunakan adalah canny edge detection. Gambar biner/abu-abu yang dihasilkan akan memiliki sekumpulan nilai 0 yang mengindikasikan non-edge dan sekumpulan nilai lebih dari atau sama dengan 1 yang mengindikasikan edge. Pada **canny(),** selain image yang akan dideteksi tepi nya, di set beberapa parameter sebagai berikut : **sigma** = 2**, low threshold** = 0.55**,** dan **high threshold** = 0.8

```
65  # Load picture, convert to grayscale and detect edges
66  image_rgb = data.coffee()[0:220, 160:420]
67  image_gray = color.rgb2gray(image_rgb)
68  edges = canny(image_gray, sigma=2.0,
69                low_threshold=0.55, high_threshold=0.8)
```

Berikut adalah hasil dari kode program diatas :



Gambar 7. Image coffee cup dalam format grayscale(a), Citra hasil canny edge detection(b)

3. Elliptical Hough Transform

Untuk melakukan circular hough transform, dapat dilakukan dengan menggunakan fungsi **hough_ellipse()** yang terdapat pada library skimage. Pada baris kode program dibawah, fungsi tersebut memiliki lima buah parameter, yaitu **image, accuracy**, **threshold**, **min_size**, dan **max_size**. Parameter image adalah citra input dengan nilai non-zero yang mewakili tepi dari objek pada citra atau dalam hal ini adalah citra hasil canny edge detection. Parameter accuracy adalah ukuran dari *bin* pada sumbu minor yang digunakan dalam akumulator, dalam hal ini nilai accuracy di set 20. Parameter threshold adalah nilai ambang batas dari akumulator, dalam hal ini nilai threshold diset 250. Parameter min_size adalah panjang sumbu utama minimal, dalam hal ini nilai min_size di set 100. Dan Parameter max_size adalah panjang sumbu minor maksimal, dalam hal ini nilai max_size di set 120. Nilai accuracy dipilih untuk mendapatkan satu akumulator tinggi. Sementara nilai threshold digunakan untuk menghilangkan akumulator yang rendah.

```
71  # Perform a Hough Transform
72  # The accuracy corresponds to the bin size of a major axis.
73  # The value is chosen in order to get a single high accumulator.
74  # The threshold eliminates low accumulators
75
76  result = hough_ellipse(edges, accuracy=20, threshold=250,
77                         min_size=100, max_size=120)
78  result.sort(order='accumulator')
```

Fungsi mengembalikan nilai ndarray dengan field [(akumulator, yc, xc, a, b, orientasi)]. Dimana (yc, xc) adalah pusat, (a, b) adalah sumbu utama dan minor. Sementara itu nilai orientasi mengikuti konvensi **skimage.draw.ellipse_perimeter()**. Dari kode program diatas, hasil akumulator yang didapat dari hough_ellipse diurutkan secara *ascending*.

4. Estimasi parameter untuk ellips

Dari hasil akumulator yang telah diurutkan, akan dilakukan estimasi terhadap 5 buah parameter yang di awal telah disinggung.

```
80  # Estimated parameters for the ellipse
81  best = list(result[-1])
82  yc, xc, a, b = [int(round(x)) for x in best[1:5]]
83  orientation = best[5]
```

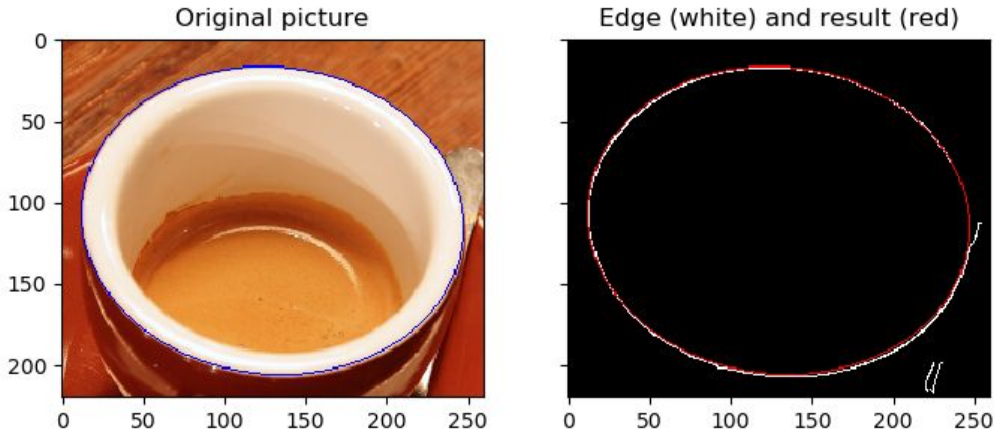
Variabel **best** merepresentasikan nilai akumulator tertinggi atau dalam hal ini menunjuk kepada item pada posisi terakhir pada list **result**. Kemudian nilai **yc** menunjuk kepada nilai pada item pertama dalam list **best**, nilai **xc** menunjuk kepada nilai pada item kedua dalam list **best**, nilai **a** menunjuk kepada nilai pada item ketiga dalam list **best**, dan nilai **b** menunjuk kepada nilai pada item keempat dalam list **best**. Kemudian, **orientation** menunjuk kepada nilai pada item kelima dalam list **best**.

5. Menggambar kembali elips dan edge yang terdeteksi diatas Input image

Untuk menggambar kontur elips yang telah dideteksi, dapat dilakukan dengan mencari perimeter dari objek elips bersangkutan. Pada library **skimage.draw** tersedia fungsi **ellipse_perimeter()** yang akan menghasilkan koordinat piksel dari elips (cy dan cx). Parameter dari fungsi ini adalah koordinat pusat dari elips (yc dan xc), semi-axes minor dan mayor (a dan b), dan orientation. Kemudian, hasil dari fungsi ini yang nantinya akan digunakan untuk menggambarkan kontur elips beserta edge yang terdeteksi pada citra input seperti yang tampak pada baris kode program dibawah :

```
85  # Draw the ellipse on the original image
86  cy, cx = ellipse_perimeter(yc, xc, a, b, orientation)
87  image_rgb[cy, cx] = (0, 0, 255)
88  # Draw the edge (white) and the resulting ellipse (red)
89  edges = color.gray2rgb(img_as_ubyte(edges))
90  edges[cy, cx] = (250, 0, 0)
```

Hasil akhir dari keseluruhan proses dapat dilihat pada gambar dibawah ini



Gambar 8. Citra asli dengan elips yang terdeteksi ditunjukkan dengan warna biru(a), Citra hasil canny edge detection dengan elips yang terdeteksi ditunjukkan dengan warna merah(b)

Dapat dilihat dari hasil output, elips terdeteksi dengan baik pada kedua citra. Walaupun pada citra hasil canny edge detection, edge tidak menutup secara sempurna namun bentuk dari elips masih dapat terdeteksi secara sempurna seperti yang tampak pada Gambar 8(b). Dapat dilihat juga pada sumbu x dan y kedua citra saling berbagi nilai yang sama, sehingga dapat disimpulkan pada kedua buah citra terdapat kontur elips yang sama.

Referensi

https://scikit-image.org/docs/dev/auto_examples/edges/plot_line_hough_transform.html

https://scikit-image.org/docs/dev/api/skimage.transform.html?highlight=hough%20circle

https://alyssaq.github.io/2014/understanding-hough-transform/

http://www.nashweb.de/articles/04-Hough-Transform-Example.html

https://homepages.inf.ed.ac.uk/rbf/HIPR2/hough.htm

http://web.ipac.caltech.edu/staff/fmasci/home/astro_refs/HoughTrans_lines_09.pdf

https://dummeraugust.com/main/content/blog/posts/demos/hough_transform_image/index.php

https://aishack.in/tutorials/circle-hough-transform/

https://scikit-image.org/docs/dev/api/skimage.draw.html

https://www.researchgate.net/publication/255577032_Ellipse_Detection_using_the_Hough_Transform