

BINUS University

Academic Career: <i>Undergraduate / Master / Doctoral *)</i>		Class Program: <i>International / Regular / Smart Program / Global Class / BINUS Online Learning *)</i>	
<input checked="" type="checkbox"/> Mid Exam	<input type="checkbox"/> Compact Term Exam	Term : Odd / Even / Compact *)	
<input type="checkbox"/> Final Exam	<input type="checkbox"/> Others Exam : _____	Period (Only for BOL) : 1 / 2 *)	
<input checked="" type="checkbox"/> Kemanggisan	<input type="checkbox"/> Senayan <input type="checkbox"/> Semarang	Academic Year :	
<input checked="" type="checkbox"/> Alam Sutera	<input type="checkbox"/> Bandung	2022 / 2023	
<input checked="" type="checkbox"/> Bekasi	<input type="checkbox"/> Malang		
Exam Type* : Onsite / Online		Faculty / Dept. : School of Computer Science	
Day / Date** : Saturday / Nov 26 th 2022		Code - Course : COMP7116001 – Computer Vision COMP7116016 – Computer Vision	
Time** : 17:00		Code - Lecturer : Team Teaching	
Exam Specification*** : <input type="checkbox"/> Open Book <input type="checkbox"/> Open Notes <input type="checkbox"/> Close Book <input type="checkbox"/> Submit Project <input type="checkbox"/> Open E-Book <input type="checkbox"/> Oral Test		BULC (Only for BOL) : -	
		Class : All Classes	
Equipment*** :		Student ID *** :	
<input type="checkbox"/> Exam Booklet	<input type="checkbox"/> Laptop <input type="checkbox"/> Drawing Paper – A3	Name *** :	
<input type="checkbox"/> Calculator	<input type="checkbox"/> Tablet <input type="checkbox"/> Drawing Paper – A2	Signature *** :	
<input type="checkbox"/> Dictionary	<input type="checkbox"/> Smartphone <input type="checkbox"/> Notes		
*) Strikethrough the unnecessary items **) For Online Exam, this is the due date ***) Only for Onsite Exam			
Please insert the test paper into the exam booklet and submit both papers after the test.			
The penalty for CHEATING is DROP OUT!			

Nama: Daniel Chandra

NIM: 2301888631

Kelas: LA05

Learning Outcomes:**LO 1 :** Describe various computational principles and standard image processing operators in computer vision**LO 2 :** Explain the local features with their detectors and descriptors in computer vision**LO 3 :** Employ various features to find the correspondence between images and perform recognition in computer vision

Verified by,
Henry Lucky (D6660) and sent to Program on Oct 31, 2022

I. ESSAY (100 %)

1. **[LO 1, LO 2, 15 points]** We have two images with different contrast as they are shown in the below figure. As a computer vision engineer you might be asked to enhance the contrast of the input image to match with the contrast of the reference image. You would think such a work is some kind of histogram matching meaning that the histogram of the input image is supposed to be similar (match) with the histogram specified by the reference image. You could then employ histogram matching (specification) algorithm to accomplish the aforementioned contrast enhancement task. Please implement the algorithm step-by-step in python notebook and show side-by-side the output image and the reference image. Please also noted that throughout the exam questions you are free to use any relevance libraries both from OpenCV and Python



INPUT IMAGE



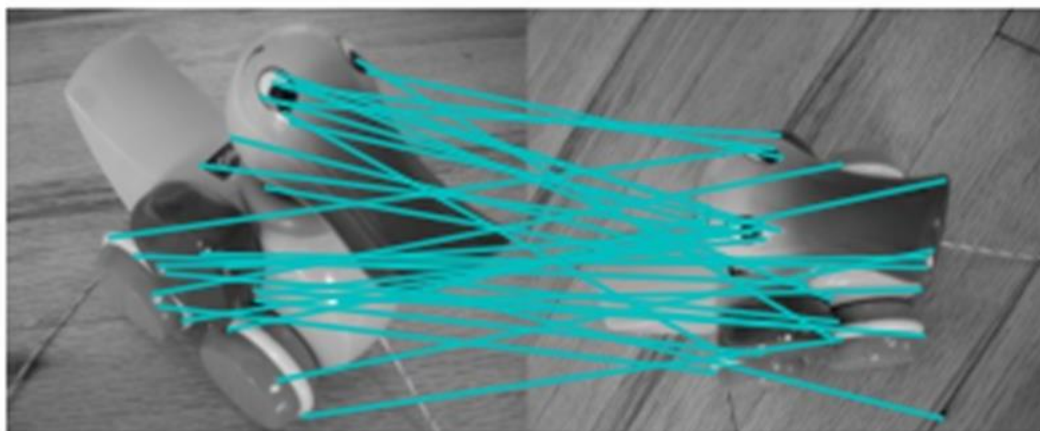
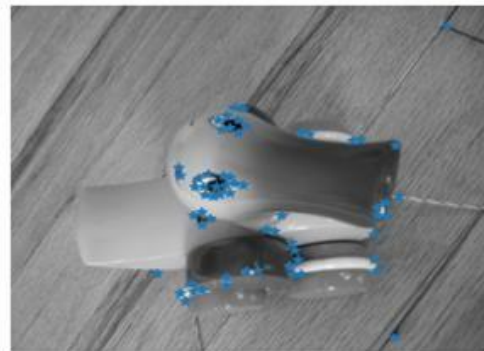
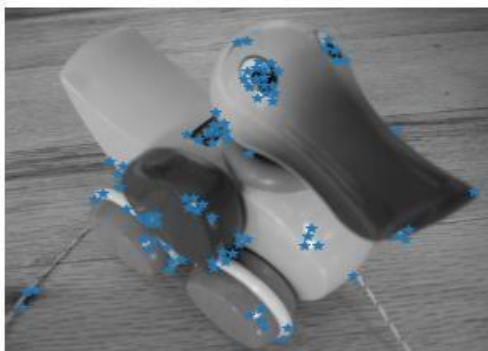
REFERENCE IMAGE

2. **[LO 1, LO 2, 15 points]** Bandpass filters are useful for removing background noise as you can see in the below figure without completely eliminating the background information. A bandpass filter may be implemented by a spatial mask such as a Gaussian filter and the step will be as follows. The original image $f(x, y)$ is first convolved with a spatial mask with a small variance to produce an output $g_1(x, y)$. Then It is convolved again with another spatial mask with a large variance to produce an output $g_2(x, y)$. The bandpass filter version of the input image is obtained as the difference between $g_1(x, y) - g_2(x, y)$. The aforementioned steps are known as spatial filtering. However, if you are familiar with filtering process in the frequency domain, employing FFT (Fast Fourier Transform) will be a great help for you to complete the problem



NOISY LENA

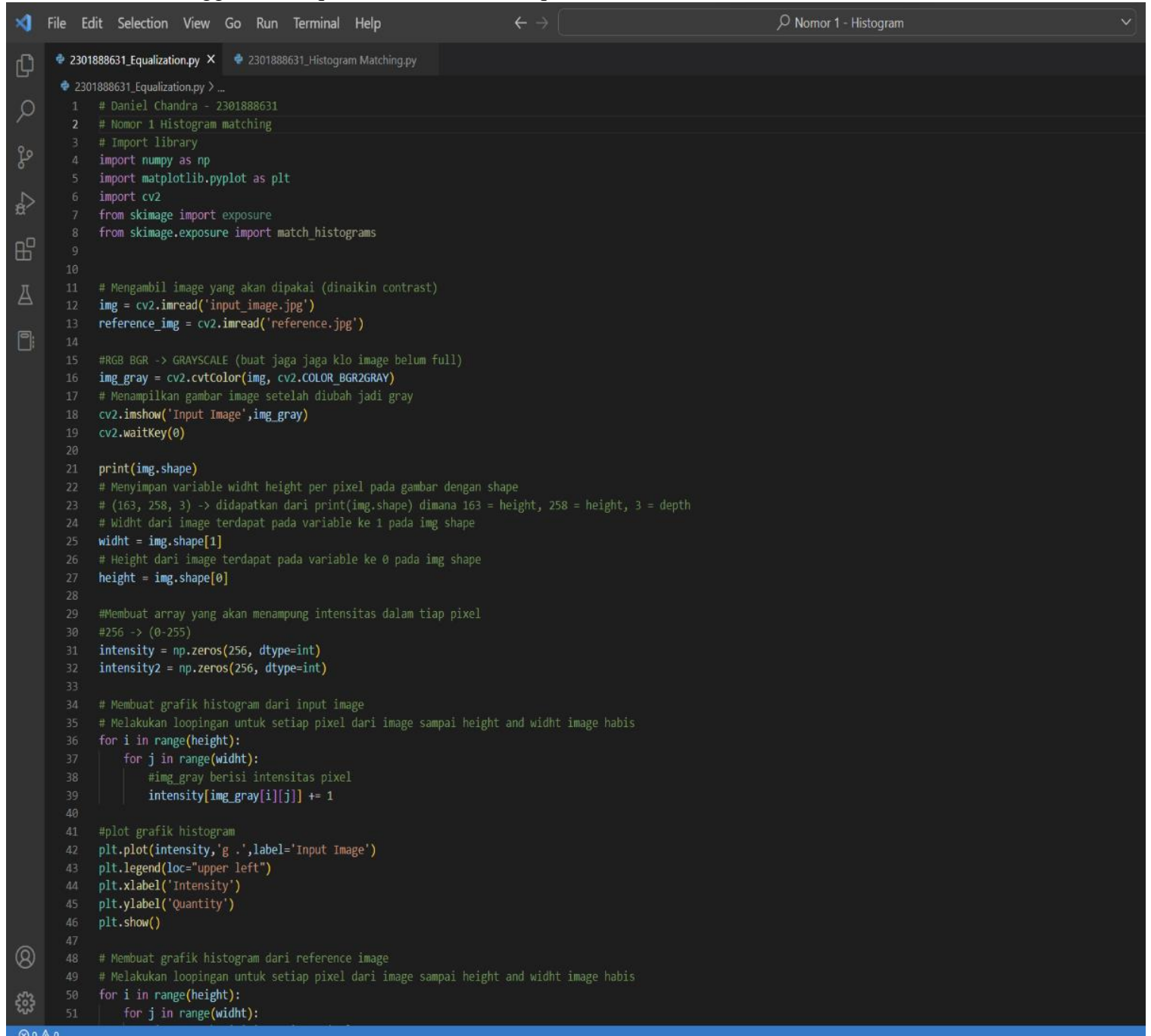
3. **[LO 1, LO 2, 25 points]** Give your thorough explanation on (A) non-maximum suppression (5%) and (B) hysteresis thresholding in the Canny Edge Detector algorithm (5%). Use diagram if necessary to show how both techniques are performed. (C) Implement Canny and LOG approaches in python notebook and demonstrate the results. Give your analysis on both methods (15%). You may use relevant python libraries and a sample image of your choice.
4. Using python notebook, demonstrate every single step of Harris corner algorithm based on the following steps (pick an image and a Gaussian filter of your own):
 - A. **[LO 1, LO 2, LO3, 5 points]** Compute Gaussian derivatives at each pixel
 - B. **[LO 1, LO 2, LO3, 5 points]** Compute second moment matrix M in a Gaussian window around each pixel
 - C. **[LO 1, LO 2, LO3, 5 points]** Compute corner response function R
 - D. **[LO 1, LO 2, LO3, 5 points]** Threshold R
 - E. **[LO 1, LO 2, LO3, 5 points]** Find local maxima of response function (non-maximum suppression)
5. **[LO 1, LO 2, LO3, 20 points]** Given some key-points in the image and their corresponding key-points in the other image as they are shown in the below figures, compute homographic from random correspondences using RANSAC algorithm. As you may know already, the homographic array will be very useful for image alignment, image stitching and other application.



JAWABAN

Daniel Chandra – 2301888631

1. Enhance Constrast of input image Menggunakan equalization dan clahe equalization



```

File Edit Selection View Go Run Terminal Help
2301888631_Equalization.py x 2301888631_Histogram Matching.py
2301888631_Equalization.py > ...
1 # Daniel Chandra - 2301888631
2 # Nomor 1 Histogram matching
3 # Import library
4 import numpy as np
5 import matplotlib.pyplot as plt
6 import cv2
7 from skimage import exposure
8 from skimage.exposure import match_histograms
9
10
11 # Mengambil image yang akan dipakai (dinaikin contrast)
12 img = cv2.imread('input_image.jpg')
13 reference_img = cv2.imread('reference.jpg')
14
15 #RGB BGR -> GRAYSCALE (buat jaga jaga klo image belum full)
16 img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
17 # Menampilkan gambar image setelah diubah jadi gray
18 cv2.imshow('Input Image',img_gray)
19 cv2.waitKey(0)
20
21 print(img.shape)
22 # Menyimpan variable widht height per pixel pada gambar dengan shape
23 # (163, 258, 3) -> didapatkan dari print(img.shape) dimana 163 = height, 258 = height, 3 = depth
24 # Widht dari image terdapat pada variable ke 1 pada img shape
25 widht = img.shape[1]
26 # Height dari image terdapat pada variable ke 0 pada img shape
27 height = img.shape[0]
28
29 #Membuat array yang akan menampung intensitas dalam tiap pixel
30 #256 -> (0-255)
31 intensity = np.zeros(256, dtype=int)
32 intensity2 = np.zeros(256, dtype=int)
33
34 # Membuat grafik histogram dari input image
35 # Melakukan looping untuk setiap pixel dari image sampai height and widht image habis
36 for i in range(height):
37     for j in range(widht):
38         #img_gray berisi intensitas pixel
39         intensity[img_gray[i][j]] += 1
40
41 #plot grafik histogram
42 plt.plot(intensity,'g .',label='Input Image')
43 plt.legend(loc="upper left")
44 plt.xlabel('Intensity')
45 plt.ylabel('Quantity')
46 plt.show()
47
48 # Membuat grafik histogram dari reference image
49 # Melakukan looping untuk setiap pixel dari image sampai height and widht image habis
50 for i in range(height):
51     for j in range(widht):

```

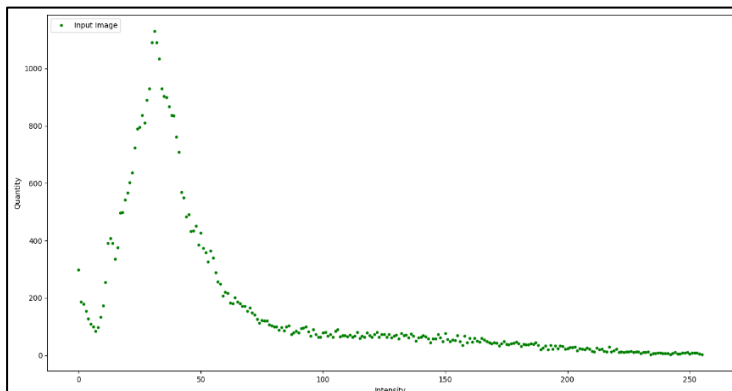
```

File Edit Selection View Go Run Terminal Help
2301888631_Equalization.py x 2301888631_Histogram Matching.py
2301888631_Equalization.py > ...
52 #img_gray berisi intensitas pixel
53 intensity2[reference_img[i][j]] += 1
54
55 #plot grafik histogram
56 plt.plot(intensity2,'g.',label='Reference Image')
57 plt.legend(loc="upper left")
58 plt.xlabel('Intensity')
59 plt.ylabel('Quantity')
60 plt.show()
61
62 #Equalization
63 equalize_img = cv2.equalizeHist(img_gray)
64 equalize_intensity = np.zeros(256,dtype=int)
65
66 for i in range(height):
67     for j in range(width):
68         equalize_intensity[equalize_img[i][j]] += 1
69
70 plt.figure(1, (16,8))
71 plt.subplot(1,2,1)
72 plt.plot(intensity, 'g.', label='Before')
73 plt.legend(loc="upper right")
74 plt.ylabel('Quantity')
75 plt.xlabel('Intensity')
76 plt.title('Before')
77
78 plt.figure(1, (16,8))
79 plt.subplot(1,2,2)
80 plt.plot(equalize_intensity, 'g', label='After')
81 plt.legend(loc="upper right")
82 plt.ylabel('Quantity')
83 plt.xlabel('Intensity')
84 plt.title('After')
85 plt.show()
86
87 #Clahe Equalization
88 clahe = cv2.createCLAHE(clipLimit=4.0, tileGridSize=(8,8))
89 clahe_img = clahe.apply(img_gray)
90
91 # Array penumpang, biar bisa ditampilkan semua image dalam satu plot
92 labels = ['Input Image','Reference Image','Hasil Equalization Image','Hasil Clahe Equalization Image']
93 images = [img,reference_img,equalize_img,clahe_img]
94
95 # Mengatur ukuran plot
96 plt.figure(figsize=(12,12))
97 # Melakukan looping untuk menampilkan semua isi array
98 for i, (lbl,image) in enumerate(zip(labels,images)):
99     plt.subplot(2,2,i+1)
100     plt.imshow(image,cmap='gray')
101     plt.title(lbl)
102     plt.show()

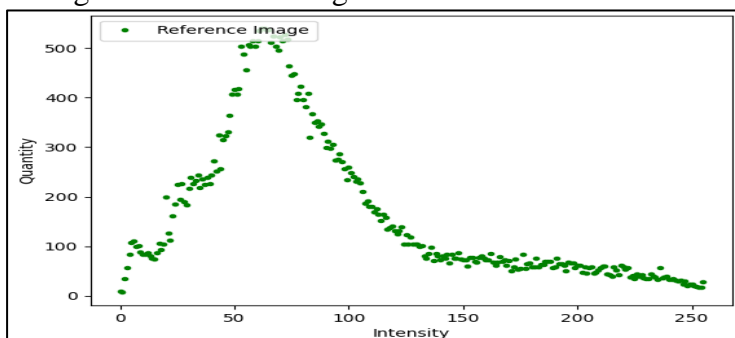
```

Output:

Histogram input image

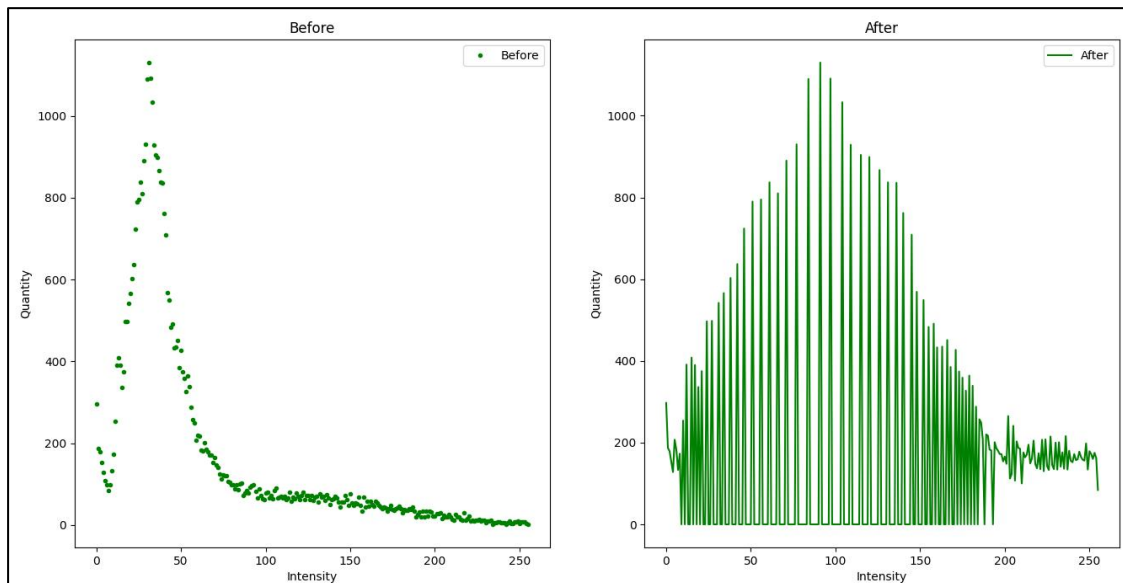


Histogram Reference image

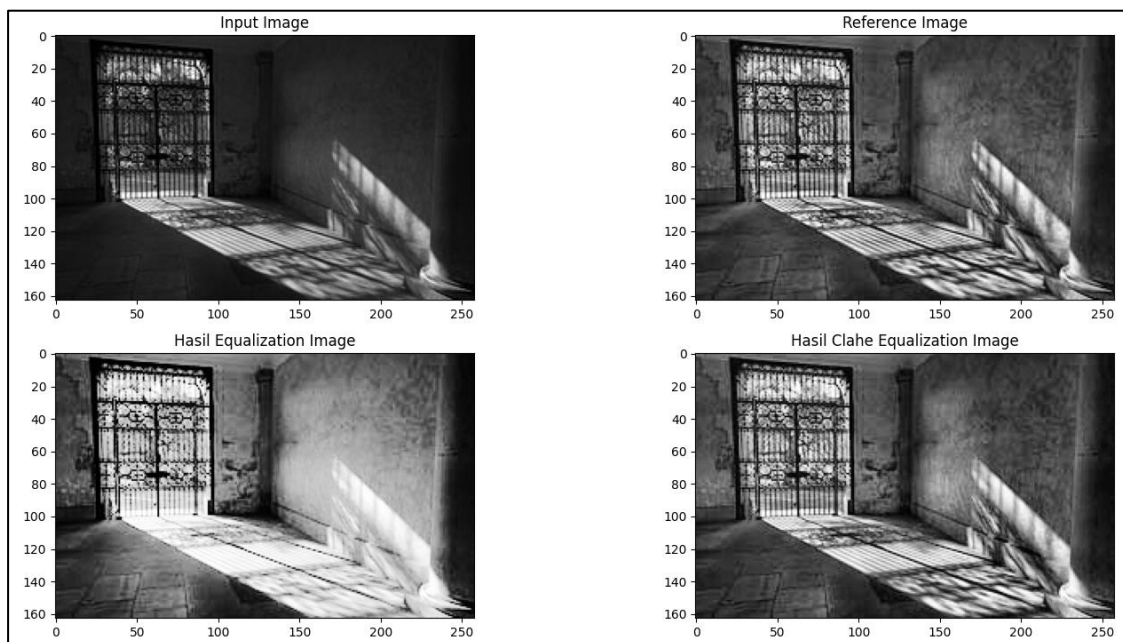


Verified by,
Henry Lucky (D6660) and sent to Program on Oct 31, 2022

Hasil histogram setelah equalization



Hasil equalization dan clahe pada input image



Menggunakan Histogram Matching

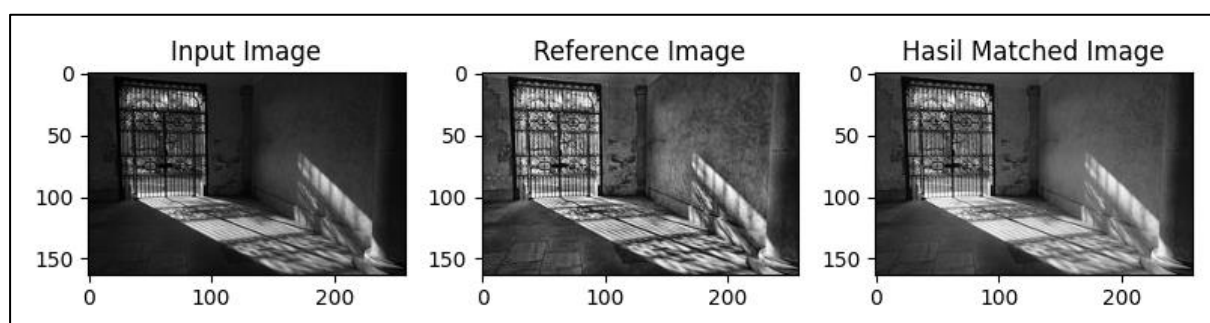
```

File Edit Selection View Go Run Terminal Help
2301888631_Equalization.py 2301888631_Histogram Matching.py X
2301888631_Histogram Matching.py > ...
1 # Import Library
2 import matplotlib.pyplot as plt
3 import cv2
4 # Menggunakan library skimage untuk melakukan match histogram
5 from skimage import data
6 from skimage import exposure
7 from skimage.exposure import match_histograms
8
9 # Mengambil image yang akan digunakan
10 input_image = cv2.imread('input_image.jpg')
11 reference_image = cv2.imread('reference.jpg')
12 # Membuat matched image dengan match_histograms() method
13 matched_image = match_histograms(input_image, reference_image, channel_axis=-1)
14
15 # Menampilkan 3 image berdampingan dalam 1 plot: input, reference dan matched
16 fig, (ax1, ax2, ax3) = plt.subplots(nrows=1, ncols=3, figsize=(8, 5))
17
18 ax1.imshow(input_image)
19 ax1.set_title('Input Image')
20 ax2.imshow(reference_image)
21 ax2.set_title('Reference Image')
22 ax3.imshow(matched_image)
23 ax3.set_title('Hasil Matched Image')
24 plt.tight_layout()
25 plt.show()
26
27
28 # Menampilkan Histogram dari ketiga image
29 fig, axes = plt.subplots(nrows=3, ncols=3, figsize=(10,8))
30
31 # Melakukan looping pada image
32 for i, img in enumerate((input_image, reference_image, matched_image)):
33     # Mendeklarasi warna setiap pixel yang akan di looping untuk menghasilkan histogram
34     # Red -> input image
35     # Green -> Reference image
36     # Blue -> Matched image
37     for c, c_color in enumerate(('red', 'green', 'blue')):
38         # Menggunakan method exposure dari library skimage pd histogram
39         img_hist, bins = exposure.histogram(img[..., c], source_range='dtype')
40         axes[c, i].plot(bins, img_hist / img_hist.max())
41         # Method cumulative distribution
42         img_cdf, bins = exposure.cumulative_distribution(img[..., c])
43         axes[c, i].plot(bins, img_cdf)
44         axes[c, 0].set_ylabel(c_color)
45 # Judul buat histogram
46 axes[0, 0].set_title('Histogram Input Image')
47 axes[0, 1].set_title('Histogram Reference Image')
48 axes[0, 2].set_title('Histogram Hasil Matched Image')
49 plt.tight_layout()
50 plt.show()

```

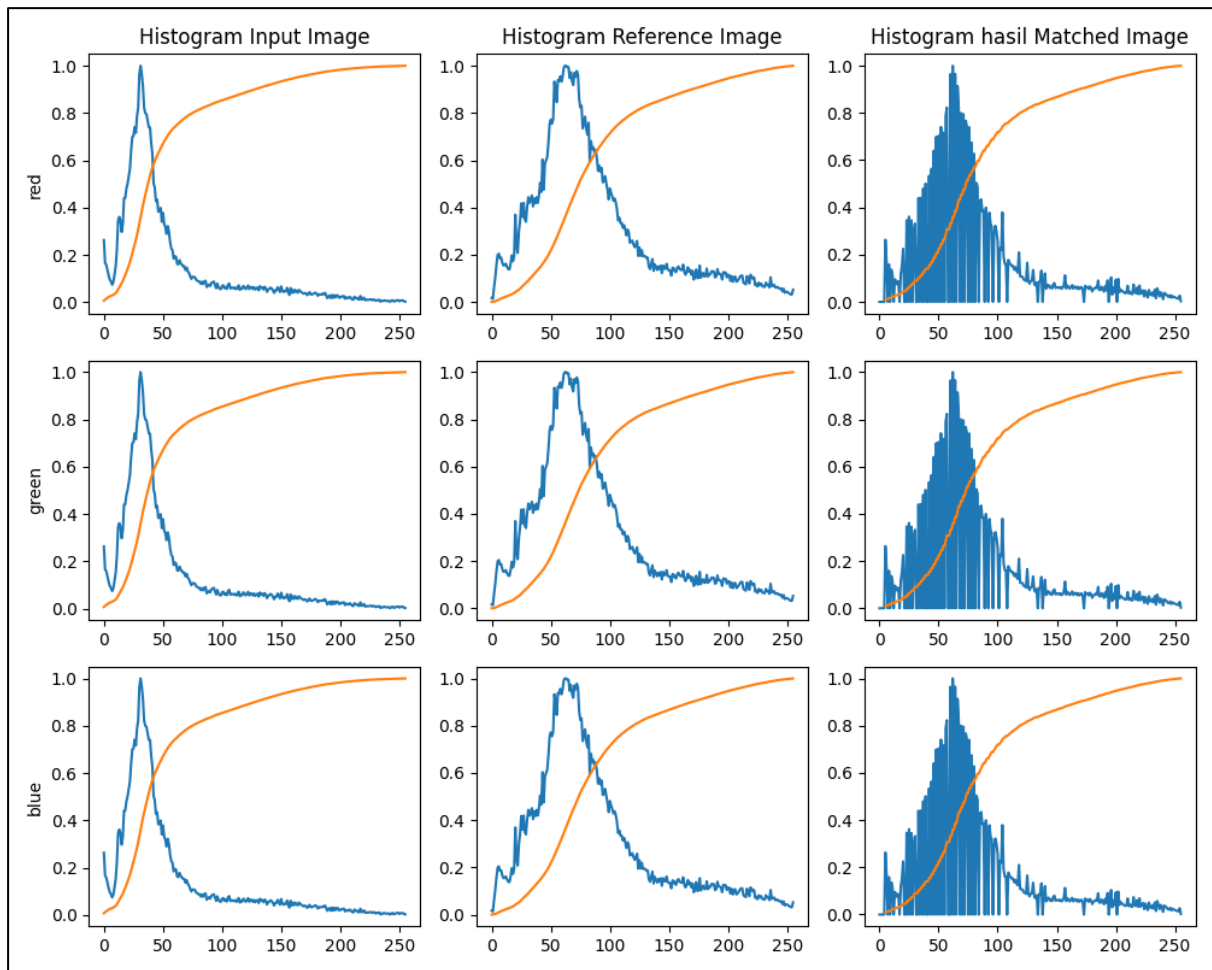
Output

Hasil image histogram matching



Verified by,
Henry Lucky (D6660) and sent to Program on Oct 31, 2022

Hasil histogram image matching



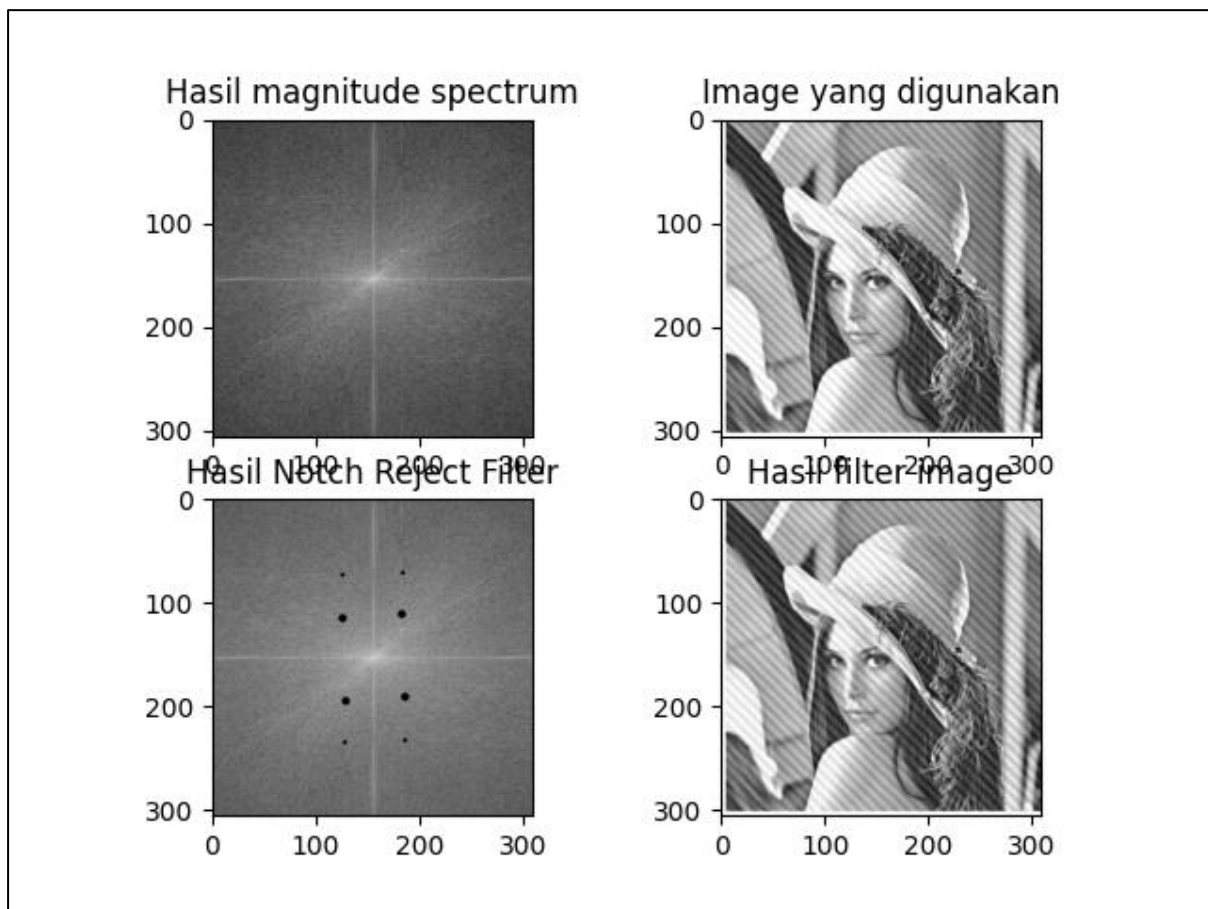
2. Filter

```

1 # Daniel Chandra - 2301888631
2 # Nomor 2 Filter
3 # Import library yang akan digunakan
4 import cv2
5 import numpy as np
6 import matplotlib.pyplot as plt
7
8 # Image yang akan digunakan
9 img = cv2.imread('noisy_lena.png', 0)
10
11 # Function notch filter
12 def notch_reject_filter(shape, d0=9, u_k=0, v_k=0):
13     P, Q = shape
14     # Initialize filter dengan numpy zeros
15     H = np.zeros((P, Q))
16     # Traverse melewati filter
17     for u in range(0, P):
18         for v in range(0, Q):
19             # Mencari distance dari point D(u,v) ke center
20             D_uv = np.sqrt((u - P/2)**2 + (v - Q/2)**2)
21             D_muv = np.sqrt((u - u_k)**2 + (v - v_k)**2)
22             if D_uv <= d0 or D_muv <= d0:
23                 H[u, v] = 0.0
24             else:
25                 H[u, v] = 1.0
26     return H
27
28 # Mengambil fungsi fft untuk apply pada image
29 f = np.fft.fft2(img)
30 fshift = np.fft.fftshift(f)
31 phase_spectrumR = np.angle(fshift)
32 magnitude_spectrum = 20*np.log(np.abs(fshift))
33
34 # Mengdeclare shape img (height,width,depth)
35 img_shape = img.shape
36
37 # Menggunakan function notch filter
38 H1 = notch_reject_filter(img_shape, 4, 38, 30)
39 H2 = notch_reject_filter(img_shape, 4, -42, 27)
40 H3 = notch_reject_filter(img_shape, 2, 80, 30)
41 H4 = notch_reject_filter(img_shape, 2, -82, 28)
42
43 # Masukkan Rumus NotchFilter
44 # Menggabungkan semua nilai H
45 NotchFilter = H1*H2*H3*H4
46 NotchRejectCenter = fshift * NotchFilter
47 NotchReject = np.fft.ifftshift(NotchRejectCenter)
48 # Menghitung inverse DFT
49 inverse_NotchReject = np.fft.ifft2(NotchReject)
50 # Mengdeclare result dengan inverse notch (hasil filter)
51 Result = np.abs(inverse_NotchReject)
52
53 # Mengplotkan hasil
54 plt.subplot(222)
55 plt.imshow(img, cmap='gray')
56 plt.title('Image yang digunakan')
57 plt.subplot(221)
58 plt.imshow(magnitude_spectrum, cmap='gray')
59 plt.title('Hasil magnitude spectrum')
60 plt.subplot(223)
61 plt.imshow(magnitude_spectrum*NotchFilter, "gray")
62 plt.title('Hasil Notch Reject Filter')
63 plt.subplot(224)
64 plt.imshow(Result, "gray")
65 plt.title('Hasil filter image')
66 plt.show()

```

Output



3. Edges

a. Non-Maximum Suppression

Non maximum suppression sering digunakan bersama dengan algoritma edge detection. Dimana gambar akan discan sepanjang arah gradien gambar, dan jika piksel bukan bagian dari maxima lokal, mereka diset ke nol. Ini memiliki efek menekan semua informasi gambar yang bukan merupakan bagian dari maxima lokal.

Non maximum suppression bekerja dengan mencari piksel dengan nilai maksimum pada edges. Non maximum suppression dapat dicapai dengan menginterpolasi piksel untuk akurasi yang lebih besar

Langkah kerja non-maximum suppression:

- Membuat matriks yang diinisialisasi ke 0 dengan ukuran yang sama dari matriks intensitas gradien asli
- Identifikasi edge direction berdasarkan nilai angle dari matriks angle
- Periksa apakah piksel dalam angle yang sama memiliki intensitas yang lebih tinggi dari piksel yang sedang diproses
- Mengreturn gambar yang diproses dengan algoritma non-max suppression

b. Hysteresis Thresholding

Thresholding hysteresis merupakan sebuah teknik untuk automatic edges detection. Namun, menghitung thresholds yang cukup tinggi dan rendah menggunakan metode tanpa pengawasan tetap menjadi masalah, perhitungan hysteresis goresan dengan menetapkan batas nilai tepi atas dan bawah. Mempertimbangkan segmen garis, jika suatu nilai terletak di atas batas atas, itu segera diterima. Jika nilainya berada di bawah ambang batas rendah maka langsung ditolak. Penggunaan dari Hysteresis edge tracking biasanya kepada piksel tepi yang lemah yang disebabkan dari tepi yang sebenarnya akan terhubung ke piksel tepi yang kuat sementara respon tidak terhubung

c. Implement Canny dan LOG

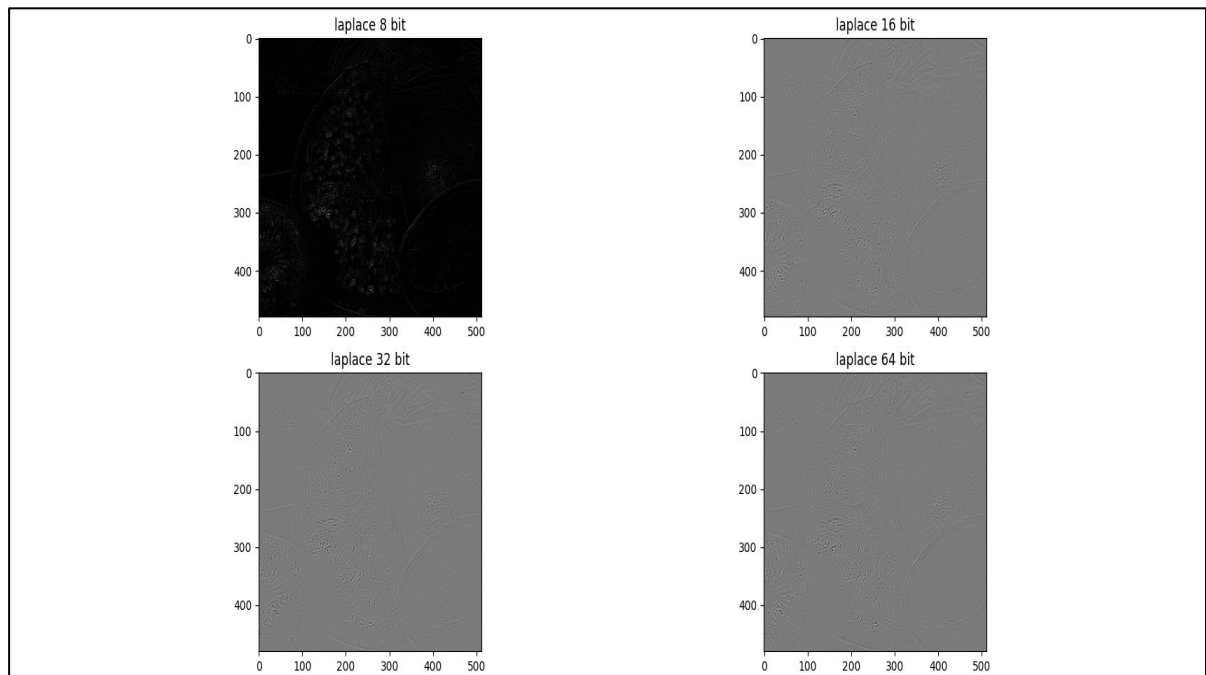
```

File Edit Selection View Go Run Terminal Help
230188631_Canny_LOG_Nomor3.py
1 # Daniel Chandra - 230188631
2 # Import library
3 import cv2
4 import numpy as np
5 import matplotlib.pyplot as plt
6
7 # Mengambil image yang ingin digunakan untuk mencari edges
8 image = cv2.imread('fruits.jpg')
9 # Mengubah image jadi gray (BGR/RGB TO GRAY)
10 imggray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
11
12 #Laplacian Edges detection
13 #Menggunakan fungsi cv Lap (variable) imggray: Mat
14 laplace_8 = cv2.Laplacian(imggray, cv2.CV_8U)
15 laplace_16 = cv2.Laplacian(imggray, cv2.CV_16S)
16 laplace_32 = cv2.Laplacian(imggray, cv2.CV_32F)
17 laplace_64 = cv2.Laplacian(imggray, cv2.CV_64F)
18
19 # Menggabungkan semua hasil menjadi 1 dalam array
20 laplace_label = ['laplace 8 bit', 'laplace 16 bit', 'laplace 32 bit', 'laplace 64 bit']
21 laplace_images = [laplace_8, laplace_16, laplace_32, laplace_64]
22
23 # Mengatur ukuran plot
24 plt.figure(figsize=(12,12))
25 # Melakukan looping untuk menampilkan semua isi array (label dan image)
26 for i, (lbl, image) in enumerate(zip(laplace_label, laplace_images)):
27     # Munculkan gambar dalam bentuk 2x2
28     plt.subplot(2,2,i+1)
29     plt.imshow(image, cmap='gray')
30     plt.title(lbl)
31 plt.show()
32
33 #Canny Edge Detection
34 # hasil output dr sobel => input dr canny
35 #lower threshold
36 #higher threshold
37
38 #dibawah lower => eliminasi
39 #di atas higher => ambil
40 #di antara => bakal dicek dulu
41
42 # Canny
43 # Menggunakan fungsi cv2 canny dengan perbandingan 2:1 / 3:1
44 # untuk mencari hasil yang terbaik
45 canny_50_100 = cv2.Canny(imggray, 50, 100)
46 canny_50_150 = cv2.Canny(imggray, 50, 150)
47 canny_75_150 = cv2.Canny(imggray, 75, 150)
48 canny_75_225 = cv2.Canny(imggray, 75, 225)
49
50 # Menggabungkan semua hasil menjadi 1 dalam array
51 canny_labels = ['canny 50 100', 'canny 50 150', 'canny 75 150', 'canny 75 225']
52 canny_img = [canny_50_100, canny_50_150, canny_75_150, canny_75_225]
53
54 # Mengatur ukuran plot
55 plt.figure(figsize=(12,12))
56 # Melakukan looping untuk menampilkan semua isi array
57 for i, (lbl, image) in enumerate(zip(canny_labels, canny_img)):
58     # Munculkan gambar dalam bentuk 2x2
59     plt.subplot(2,2,i+1)
60     plt.imshow(image, cmap='gray')
61     plt.title(lbl)
62 plt.show()
63

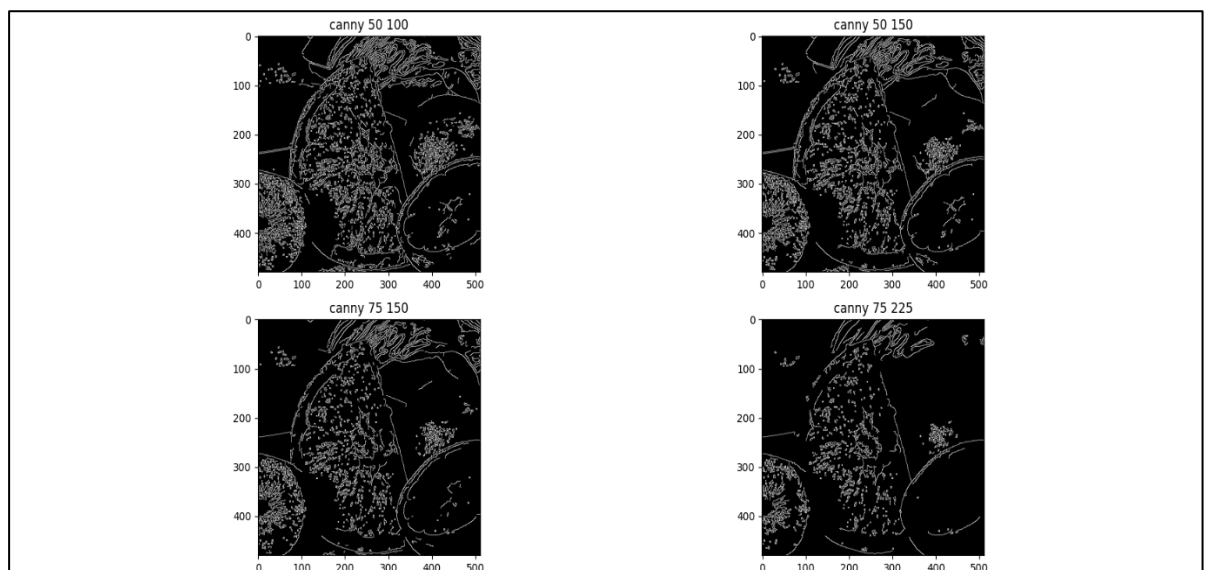
```

Image yang digunakanOutput

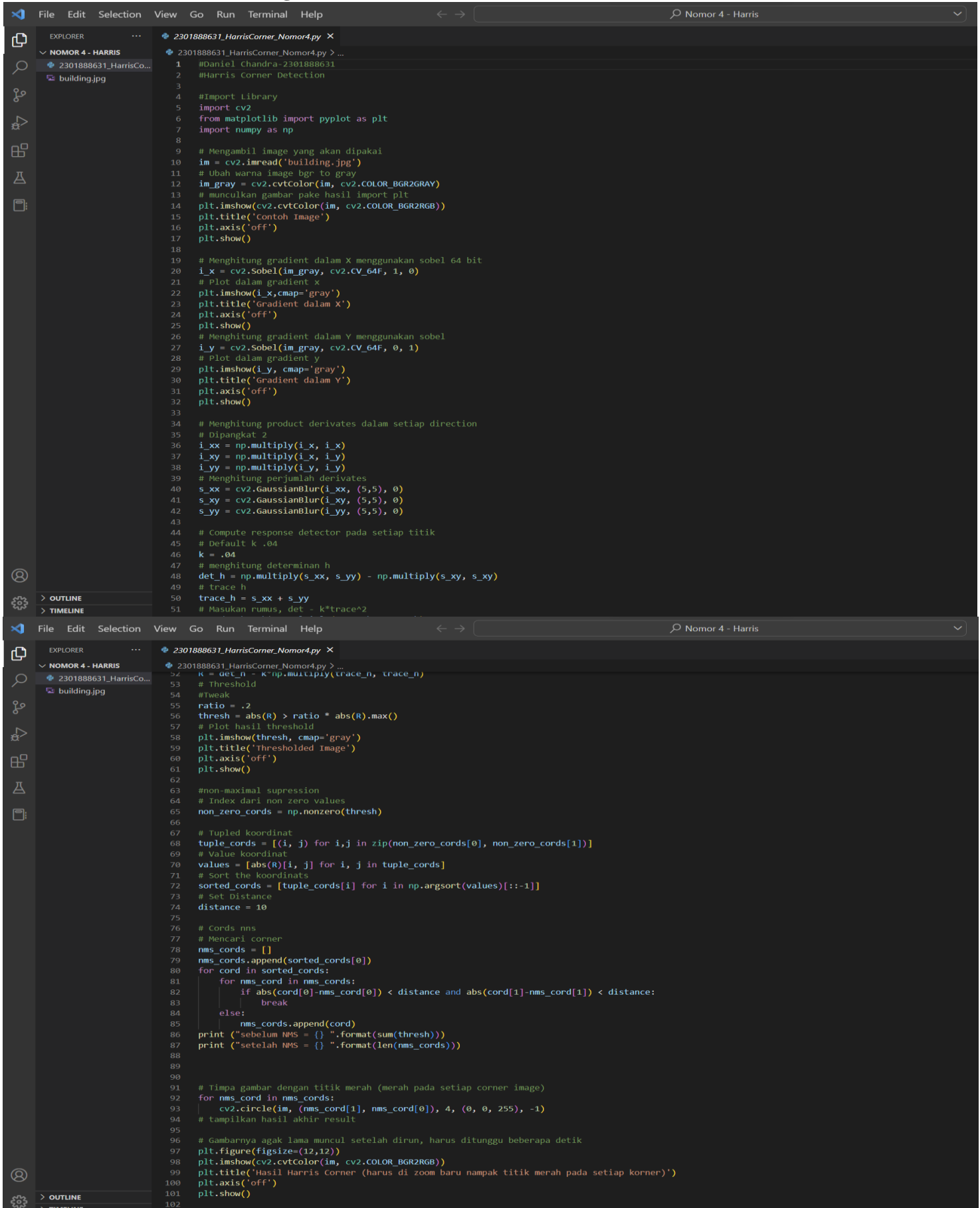
LOG



CANNY



4. Harris Corner Algorithm



```

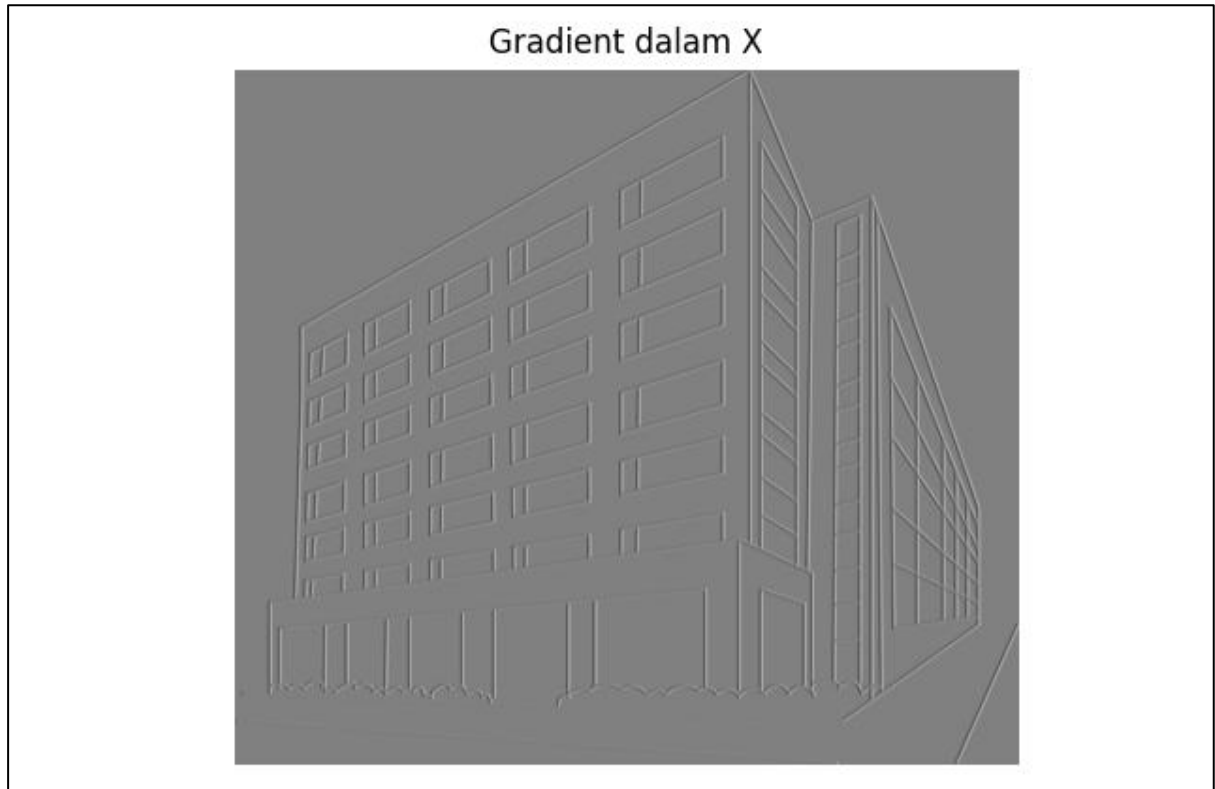
File Edit Selection View Go Run Terminal Help
2301888631_HarrisCorner_Nomor4.py
1 #Daniel Chandra-2301888631
2 #Harris Corner Detection
3
4 #Import Library
5 import cv2
6 from matplotlib import pyplot as plt
7 import numpy as np
8
9 # Mengambil image yang akan dipakai
10 im = cv2.imread('building.jpg')
11 # Ubah warna image bgr to gray
12 im_gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
13 # munculkan gambar pake hasil import plt
14 plt.imshow(cv2.cvtColor(im, cv2.COLOR_BGR2RGB))
15 plt.title('contoh Image')
16 plt.axis('off')
17 plt.show()
18
19 # Menghitung gradient dalam X menggunakan sobel 64 bit
20 i_x = cv2.Sobel(im_gray, cv2.CV_64F, 1, 0)
21 # Plot dalam gradient x
22 plt.imshow(i_x, cmap='gray')
23 plt.title('gradient dalam X')
24 plt.axis('off')
25 plt.show()
26 # Menghitung gradient dalam Y menggunakan sobel
27 i_y = cv2.Sobel(im_gray, cv2.CV_64F, 0, 1)
28 # Plot dalam gradient y
29 plt.imshow(i_y, cmap='gray')
30 plt.title('gradient dalam Y')
31 plt.axis('off')
32 plt.show()
33
34 # Menghitung product derivatives dalam setiap direction
35 # Dipangkat 2
36 i_xx = np.multiply(i_x, i_x)
37 i_xy = np.multiply(i_x, i_y)
38 i_yy = np.multiply(i_y, i_y)
39 # Menghitung perjumlahan derivatives
40 s_xx = cv2.GaussianBlur(i_xx, (5,5), 0)
41 s_xy = cv2.GaussianBlur(i_xy, (5,5), 0)
42 s_yy = cv2.GaussianBlur(i_yy, (5,5), 0)
43
44 # Compute response detector pada setiap titik
45 # Default k .04
46 k = .04
47 # menghitung determinan h
48 det_h = np.multiply(s_xx, s_yy) - np.multiply(s_xy, s_xy)
49 # trace h
50 trace_h = s_xx + s_yy
51 # Masukan rumus, det - k*trace^2
52 R = det_h - k*np.multiply(trace_h, trace_h)
53 # Threshold
54 #Tweak
55 ratio = .2
56 thresh = abs(R) > ratio * abs(R).max()
57 # Plot hasil threshold
58 plt.imshow(thresh, cmap='gray')
59 plt.title('Thresholded Image')
60 plt.axis('off')
61 plt.show()
62
63 #non-maximal suppression
64 # Index dari non zero values
65 non_zero_cords = np.nonzero(thresh)
66
67 # Tupled koordinat
68 tuple_cords = [(i, j) for i, j in zip(non_zero_cords[0], non_zero_cords[1])]
69 # Value koordinat
70 values = [abs(R)[i, j] for i, j in tuple_cords]
71 # Sort the koordinats
72 sorted_cords = [tuple_cords[i] for i in np.argsort(values)[::-1]]
73 # Set Distance
74 distance = 10
75
76 # Cords nms
77 # Mencari corner
78 nms_cords = []
79 nms_cords.append(sorted_cords[0])
80 for cord in sorted_cords:
81     for nms_cord in nms_cords:
82         if abs(cord[0]-nms_cord[0]) < distance and abs(cord[1]-nms_cord[1]) < distance:
83             break
84     else:
85         nms_cords.append(cord)
86 print ("sebelum NMS = {}".format(sum(thresh)))
87 print ("setelah NMS = {}".format(len(nms_cords)))
88
89
90
91 # Timpas gambar dengan titik merah (merah pada setiap corner image)
92 for nms_cord in nms_cords:
93     cv2.circle(im, (nms_cord[1], nms_cord[0]), 4, (0, 0, 255), -1)
94 # tampilkan hasil akhir result
95
96 # Gambarnya agak lama muncul setelah dirun, harus ditunggu beberapa detik
97 plt.figure(figsize=(12,12))
98 plt.imshow(cv2.cvtColor(im, cv2.COLOR_BGR2RGB))
99 plt.title('Hasil Harris Corner (harus di zoom baru nampak titik merah pada setiap korne)')
100 plt.axis('off')
101 plt.show()
102

```

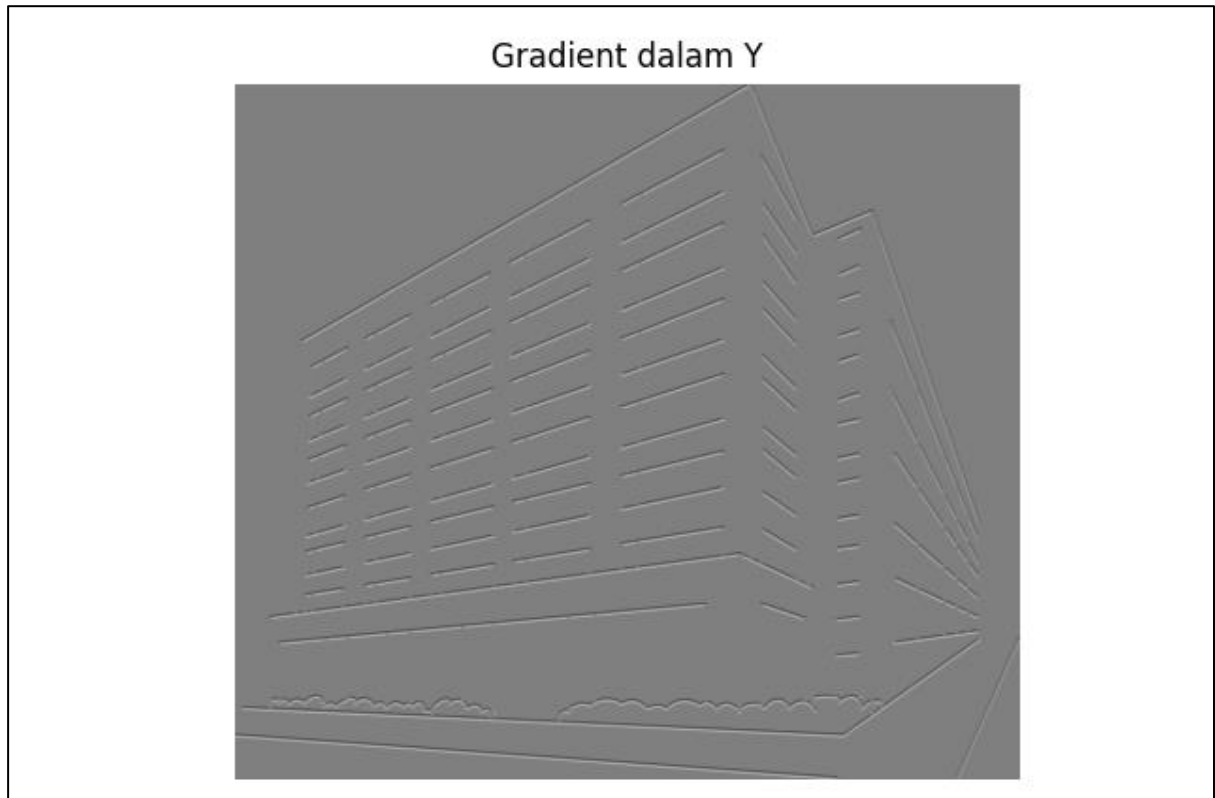
Verified by,
Henry Lucky (D6660) and sent to Program on Oct 31, 2022

Output Harris Corner algorithm

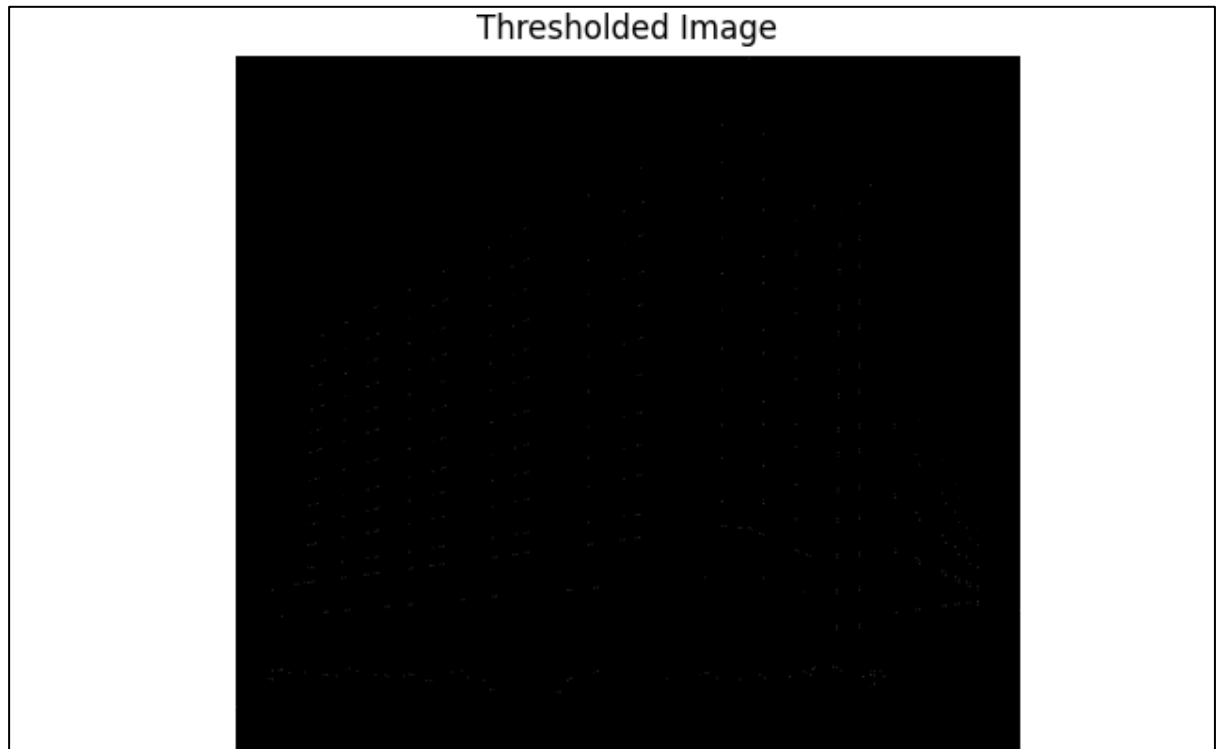
Gradient dalam x



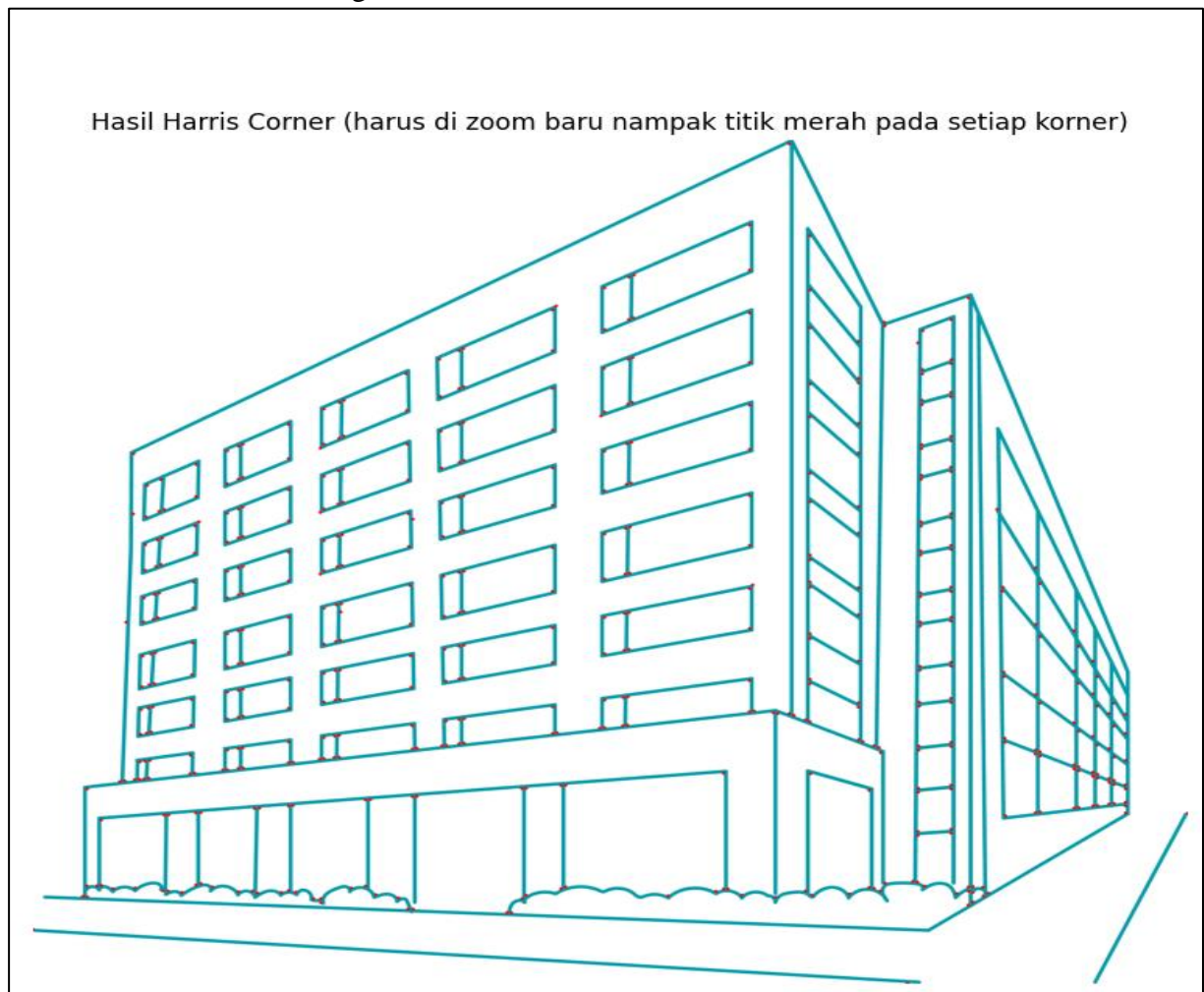
Gradient dalam y



Threshold



Hasil Corner Detection image



Verified by,
Henry Lucky (D6660) and sent to Program on Oct 31, 2022

5. Ransac Feature Matching

```

1  # Daniel Chandra - 2301888631
2  # Nomor 5 - Matching with sift & calculate homographic ransac
3  # Import library
4  import numpy as np
5  import cv2 as cv
6  from matplotlib import pyplot as plt
7
8  # Mengambil image yang akan dimatching
9  # Object image yang akan digunakan
10 img1 = cv.imread('box.png',0)
11 # Scene image yang akan dimatching
12 img2 = cv.imread('box_in_scene.png',0)
13
14 # Bikin object detector sift
15 sift = cv.SIFT_create()
16 # Mencari/mendeteksi keypoint dan deskriptornya
17 # parameter 1 = source image
18 # parameter 2 = masking
19 kp1, des1 = sift.detectAndCompute(img1,None)
20 kp2, des2 = sift.detectAndCompute(img2,None)
21
22 #flann
23 FLANN_INDEX_KDTREE = 1
24 index_params = dict(algorithm = FLANN_INDEX_KDTREE, trees = 5)
25 search_params = dict(checks = 50)
26 # Menggil (variable) FlannBasedMatcher: Any
27 flann = cv.FlannBasedMatcher(index_params, search_params)
28 matches = flann.knnMatch(des1,des2,k=2)
29 # Array buat menyimpan semua matches yang bagus
30 good = []
31 # bruteforce buat bandingin obj & scene
32 #rumus, jika distance fm < 0.7 * sm maka diappend
33 for m,n in matches:
34     if m.distance < 0.7*n.distance:
35         good.append(m)
36
37 # Calculate Homography ransac
38 MIN_MATCH_COUNT = 10
39 # Menetapkan if bahwa setidaknya 10 kecocokan (MIN_MATCH_COUNT) harus terdapat di object
40 # Jika ditemukan kecocokan, maka mengekstrak lokasi titik kunci yang cocok pada kedua gambar
41 # Transformasi perspektif, setelah didapatkan matriks transformasi 3x3, menggunakan sudut object_image ke
42 if len(good)>MIN_MATCH_COUNT:
43     src_pts = np.float32([ kp1[m.queryIdx].pt for m in good ]).reshape(-1,1,2)
44     dst_pts = np.float32([ kp2[m.trainIdx].pt for m in good ]).reshape(-1,1,2)
45     M, mask = cv.findHomography(src_pts, dst_pts, cv.RANSAC,5.0)
46     matchesMask = mask.ravel().tolist()
47     h,w = img1.shape
48     pts = np.float32([ [0,0],[0,h-1],[w-1,h-1],[w-1,0] ]).reshape(-1,1,2)
49     dst = cv.perspectiveTransform(pts,M)
50     img2 = cv.polylines(img2,[np.int32(dst)],True,255,3, cv.LINE_AA)
51 # Jika tidak ditemukan kecocokan, maka tinggal print kalimat
52 else:
53     print( "Kecocokan yang ditemukan tidak mencukupi - {}/{}".format(len(good), MIN_MATCH_COUNT) )
54     matchesMask = None
55
56 # Gambar Hasil matching pada image
57 draw_params = dict(matchColor = (0,255,255), #warna biru terang
58                     singlePointColor = None,
59                     matchesMask = matchesMask,
60                     flags = 2)
61
62 # Menggabungkan kedua gambar dan melakukan matching
63 img3 = cv.drawMatches(img1,kp1,img2,kp2,good,None,**draw_params)
64 # plot
65 plt.imshow(img3,'gray'),plt.show()
66

```

Image yang digunakan

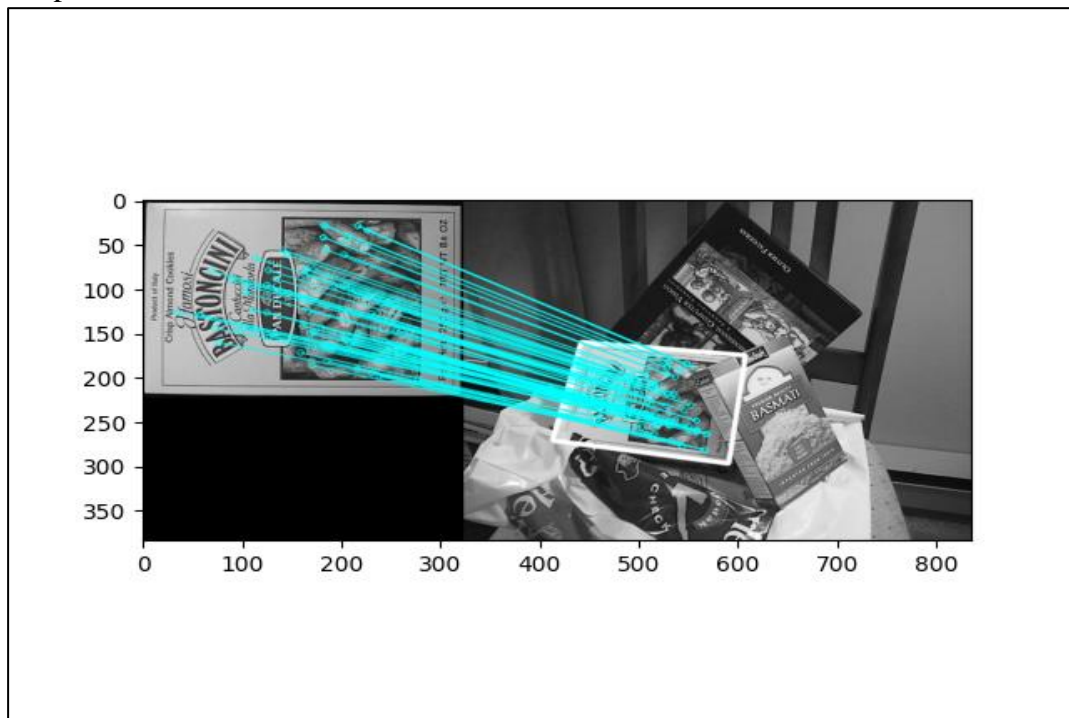
Object



Scene



Output



Verified by,
Henry Lucky (D6660) and sent to Program on Oct 31, 2022