BINUS University

Academic Career: ***Undergraduate / ~~Master~~ / ~~Doctoral~~ *)***			**Class Program:** ***~~International~~ / Regular / ~~Smart Program~~ /*** ***~~Global Class~~ / ~~BINUS Online Learning~~ *)***		
☑ Mid Exam **☐ Final Exam**	**☐ Compact Term Exam** **☐ Others Exam : _____**		**Term : Odd / ~~Even~~ / ~~Compact~~ *)** **Period (Only for BOL) : ~~1~~ / ~~2~~ *)**		
☑ Kemanggisan **☑ Alam Sutera** **☑ Bekasi**	**☐ Senayan** **☐ Semarang** **☐ Bandung** **☐ Malang**		**Academic Year :** **2022 / 2023**		
Exam Type*	: ~~Onsite~~ / Online		Faculty / Dept.	:	School of Computer Science
Day / Date**	: Saturday / Nov 26th 2022		Code - Course	:	COMP7116001 – Computer Vision COMP7116016 – Computer Vision
Time**	: 17:00		Code - Lecturer	:	Team Teaching
Exam Specification***	: ☐ Open Book ☐ Open Notes ☐ Close Book ☐ Submit Project ☐ Open E-Book ☐ Oral Test		BULC (Only for BOL)	:	-
			Class	:	All Classes
Equipment***	:		Student ID ***	:	
☐ Exam Booklet ☐ Calculator ☐ Dictionary	☐ Laptop ☐ Tablet ☐ Smartphone	☐ Drawing Paper – A3 ☐ Drawing Paper – A2 ☐ Notes	Name ***	:	
			Signature ***	:	

☐) *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 g1(x, y). Then It is convolved again with another spatial mask with a large variance to produce an output g2(x, y). The bandpass filter version of the input image is obtained as the difference between g1(x, y) – g2(x, y). The aforementioned steps are known as spatial filtering. However, iIf 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

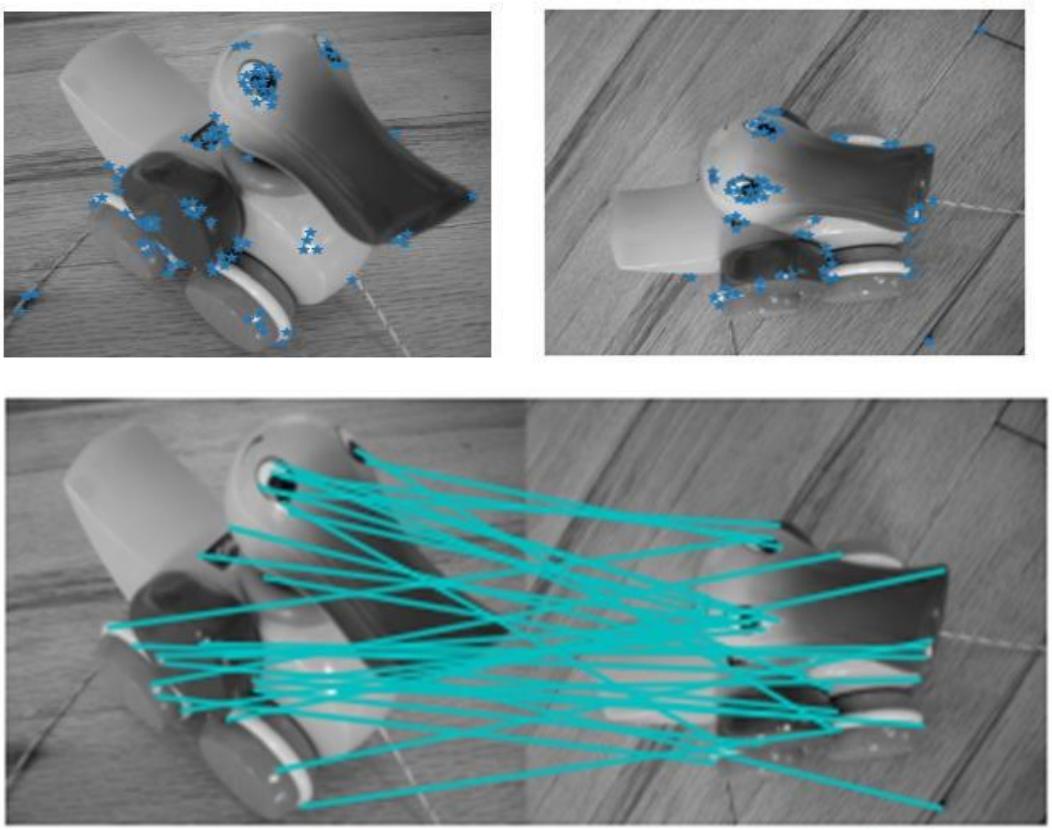
NOISY LENA

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 bothmethods (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 thefollowing 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 windowaround 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-maximumsuppression)

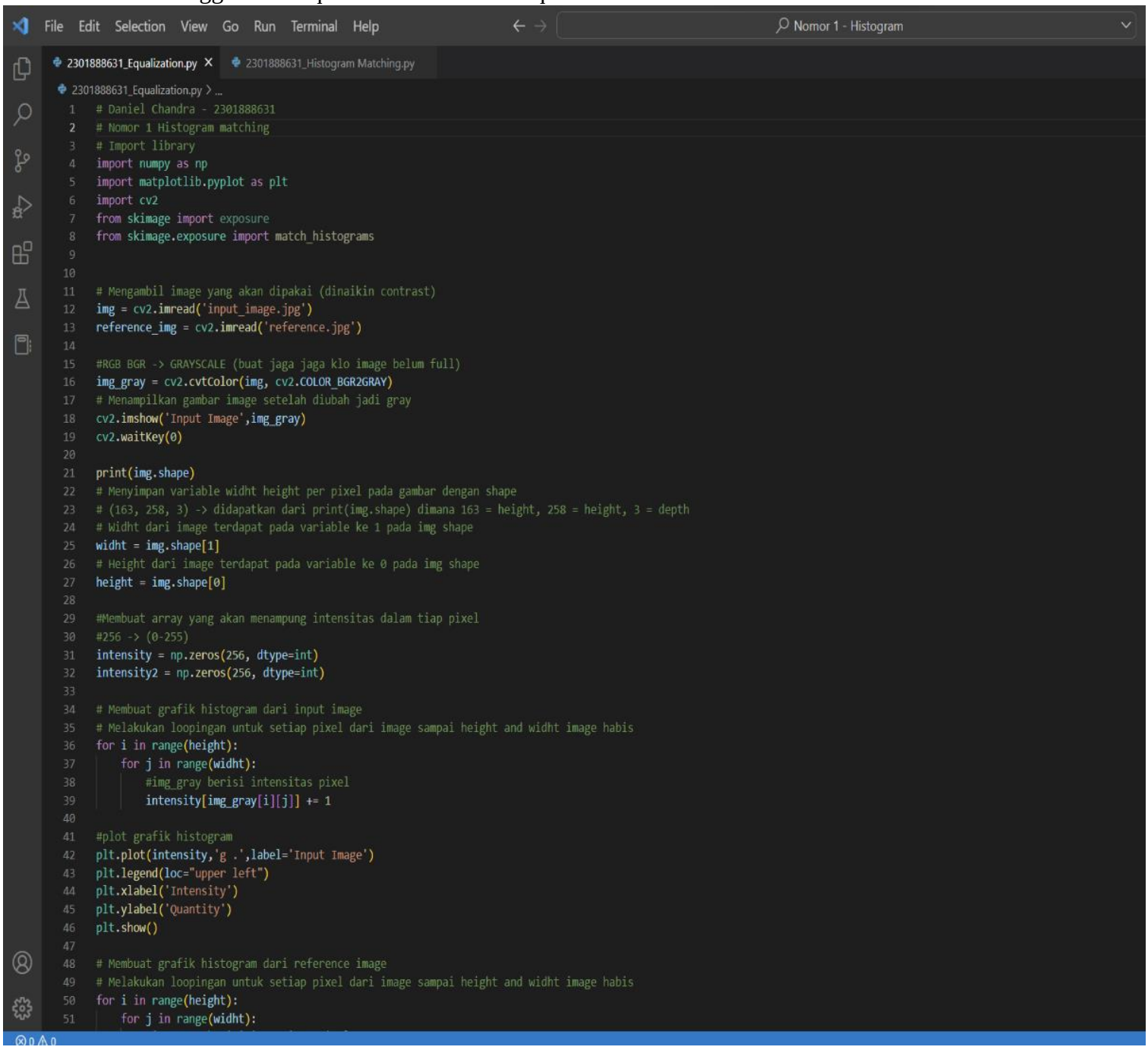
5. **[LO 1, LO 2, LO3, 20 points]** Given some key-points in the image and their corresponding key- points in the other image as they are shown in the below figures, compute homographic from random correspondences using RANSAC algorithm. As you may know already, the homographic array will be very useful for image alignment, image stitching and other application.



JAWABAN

Daniel Chandra – 2301888631

1. **Enchance Constrast of input image**
   Menggunakan equalization dan clahe equalization

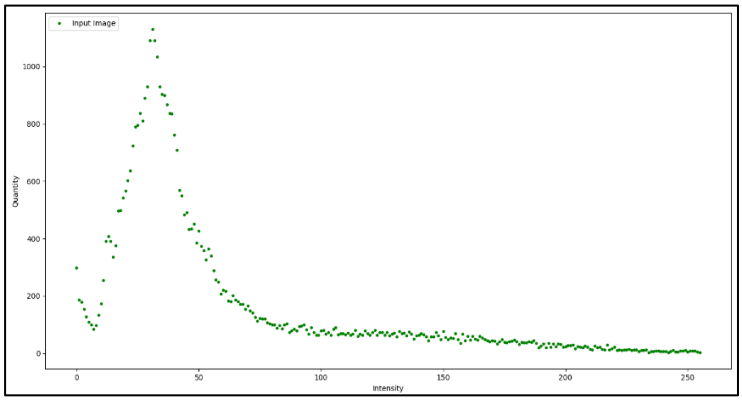


```python
# Daniel Chandra - 2301888631
# Nomor 1 Histogram matching
# Import library
import numpy as np
import matplotlib.pyplot as plt
import cv2
from skimage import exposure
from skimage.exposure import match_histograms


# Mengambil image yang akan dipakai (dinaikin contrast)
img = cv2.imread('input_image.jpg')
reference_img = cv2.imread('reference.jpg')

#RGB BGR -> GRAYSCALE (buat jaga jaga klo image belum full)
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# Menampilkan gambar image setelah diubah jadi gray
cv2.imshow('Input Image',img_gray)
cv2.waitKey(0)

print(img.shape)
# Menyimpan variable widht height per pixel pada gambar dengan shape
# (163, 258, 3) -> didapatkan dari print(img.shape) dimana 163 = height, 258 = height, 3 = depth
# Widht dari image terdapat pada variable ke 1 pada img shape
widht = img.shape[1]
# Height dari image terdapat pada variable ke 0 pada img shape
height = img.shape[0]

#Membuat array yang akan menampung intensitas dalam tiap pixel
#256 -> (0-255)
intensity = np.zeros(256, dtype=int)
intensity2 = np.zeros(256, dtype=int)

# Membuat grafik histogram dari input image
# Melakukan loopingan untuk setiap pixel dari image sampai height and widht image habis
for i in range(height):
    for j in range(widht):
        #img_gray berisi intensitas pixel
        intensity[img_gray[i][j]] += 1

#plot grafik histogram
plt.plot(intensity,'g .',label='Input Image')
plt.legend(loc="upper left")
plt.xlabel('Intensity')
plt.ylabel('Quantity')
plt.show()

# Membuat grafik histogram dari reference image
# Melakukan loopingan untuk setiap pixel dari image sampai height and widht image habis
for i in range(height):
    for j in range(widht):
```

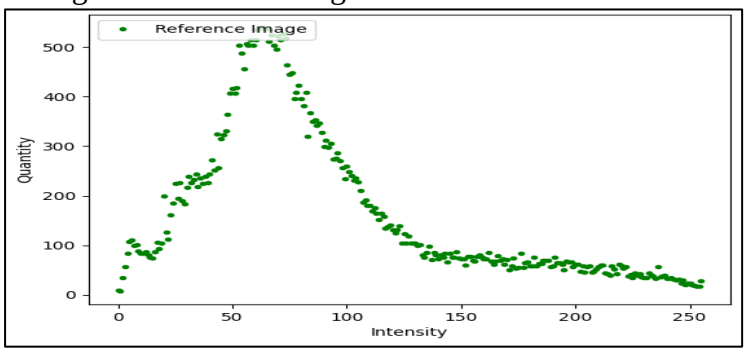
```
#img_gray berisi intensitas pixel
intensity2[reference_img[i][j]] += 1

#plot grafik histogram
plt.plot(intensity2,'g .',label='Reference Image')
plt.legend(loc="upper left")
plt.xlabel('Intensity')
plt.ylabel('Quantity')
plt.show()

#Equalization
equalize_img = cv2.equalizeHist(img_gray)
equalize_intensity = np.zeros(256,dtype=int)

for i in range(height):
    for j in range(widht):
        equalize_intensity[equalize_img[i][j]] += 1

plt.figure(1, (16,8))
plt.subplot(1,2,1)
plt.plot(intensity, 'g .', label='Before')
plt.legend(loc="upper right")
plt.ylabel('Quantity')
plt.xlabel('Intensity')
plt.title('Before')

plt.figure(1, (16,8))
plt.subplot(1,2,2)
plt.plot(equalize_intensity, 'g', label='After')
plt.legend(loc="upper right")
plt.ylabel('Quantity')
plt.xlabel('Intensity')
plt.title('After')
plt.show()

#Clahe Equalization
clahe = cv2.createCLAHE(clipLimit=4.0, tileGridSize=(8,8))
clahe_img = clahe.apply(img_gray)

# Array penumpang, biar bisa ditampilkan semua image dalam satu plot
labels = ['Input Image','Reference Image','Hasil Equalization Image','Hasil Clahe Equalization Image']
images = [img,reference_img,equalize_img,clahe_img]

# Mengatur ukuran plot
plt.figure(figsize=(12,12))
# Melakukan looping untuk menampilkan semua isi array
for i, (lbl,image) in enumerate(zip(labels,images)):
    plt.subplot(2,2,i+1)
    plt.imshow(image,cmap='gray')
    plt.title(lbl)
plt.show()
```

Output:

Histogram input image

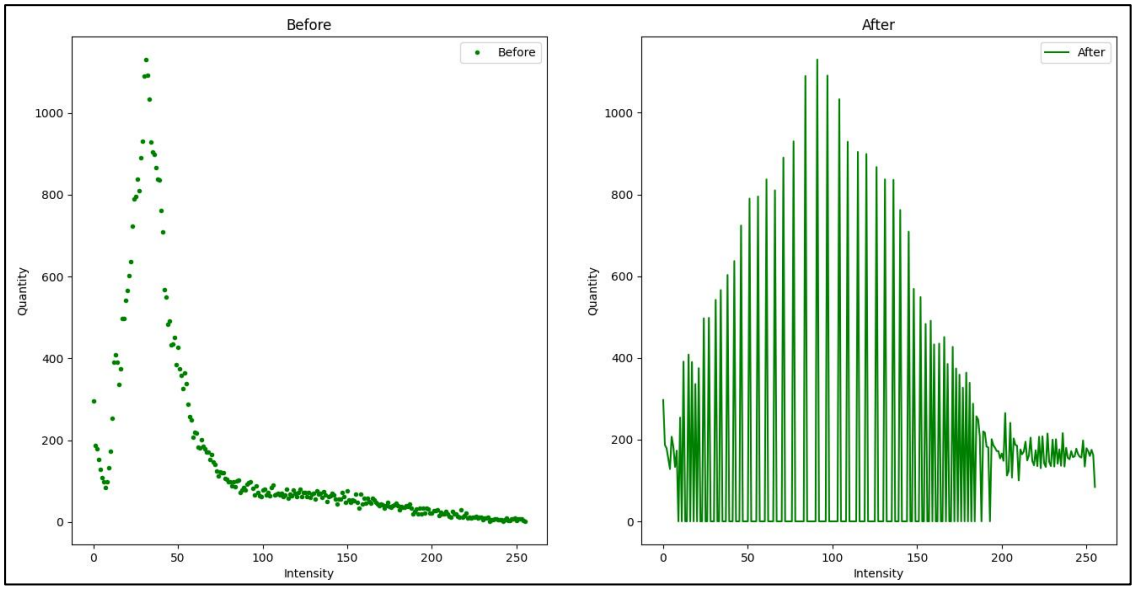


Histogram Reference image

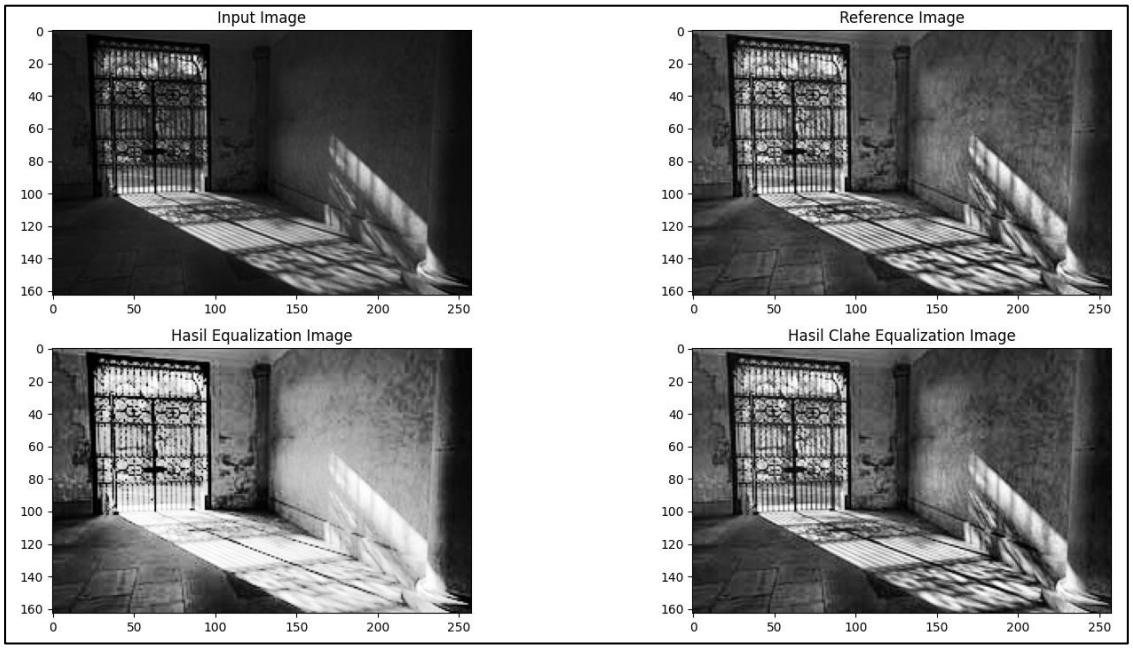


Verified by,
Henry Lucky (D6660) and sent to Program on Oct 31, 2022

Hasil histogram setelah equalization



Hasil equalization dan clahe pada input image



Menggunakan Histogram Matching

File Edit Selection View Go Run Terminal Help — Nomor 1 - Histogram

2301888631_Equalization.py | 2301888631_Histogram Matching.py

2301888631_Histogram Matching.py > ...

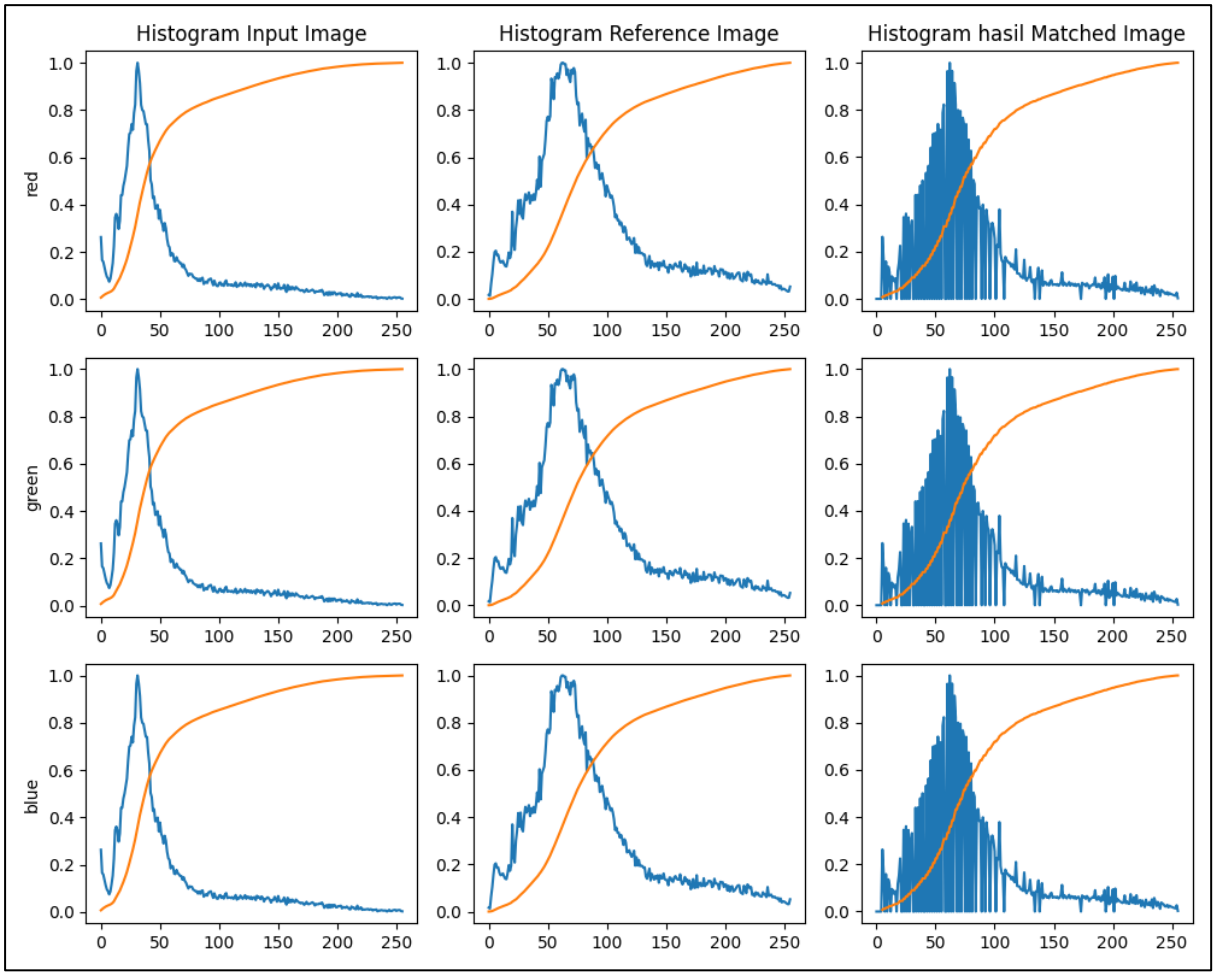
```python
# Import Library
import matplotlib.pyplot as plt
import cv2
# Menggunakan library skimage untuk melakukan match histogram
from skimage import data
from skimage import exposure
from skimage.exposure import match_histograms

# Mengambil image yang akan digunakan
input_image = cv2.imread('input_image.jpg')
reference_image = cv2.imread('reference.jpg')
# Membuat matched image dengan match_histogram() method
matched_image = match_histograms(input_image, reference_image, channel_axis=-1)

# Menampilkan 3 image berdampingan dalam 1 plot: input, reference dan matched
fig, (ax1, ax2, ax3) = plt.subplots(nrows=1, ncols=3, figsize=(8, 5))

ax1.imshow(input_image)
ax1.set_title('Input Image')
ax2.imshow(reference_image)
ax2.set_title('Reference Image')
ax3.imshow(matched_image)
ax3.set_title('Hasil Matched Image')
plt.tight_layout()
plt.show()


# Menampilkan Histogram dari ketiga image
fig, axes = plt.subplots(nrows=3, ncols=3, figsize=(10,8))

# Melakukan loopingan pada image
for i, img in enumerate((input_image, reference_image, matched_image)):
    # Mengdeclare warna setiap pixel yang akan dilooping untuk menghasilkan histogram
    # Red -> input image
    # Green -> Reference image
    # Blue -> Matched image
    for c, c_color in enumerate(('red', 'green', 'blue')):
        # Menggunakan method exposure dari library skimage pd histogram
        img_hist, bins = exposure.histogram(img[..., c], source_range='dtype')
        axes[c, i].plot(bins, img_hist / img_hist.max())
        # Method cumulative distribution
        img_cdf, bins = exposure.cumulative_distribution(img[..., c])
        axes[c, i].plot(bins, img_cdf)
        axes[c, 0].set_ylabel(c_color)
# Judul buat histogram
axes[0, 0].set_title('Histogram Input Image')
axes[0, 1].set_title('Histogram Reference Image')
axes[0, 2].set_title('Histogram hasil Matched Image')
plt.tight_layout()
plt.show()
```

Output

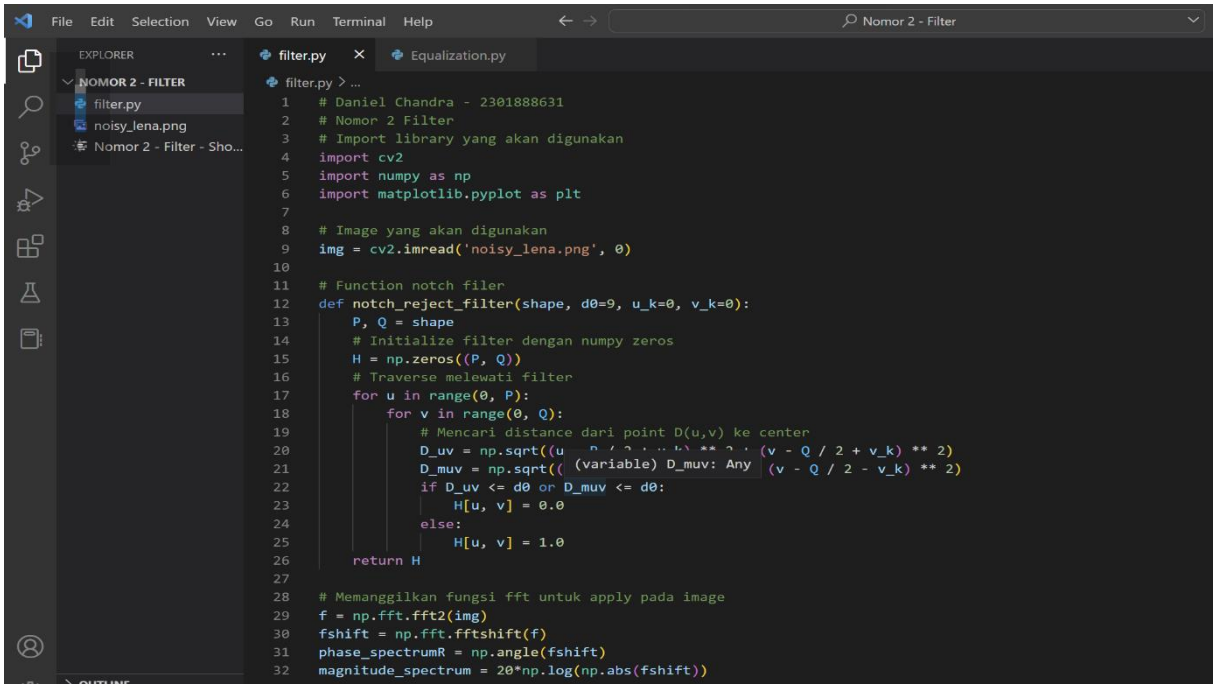
Hasil image histogram matching

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

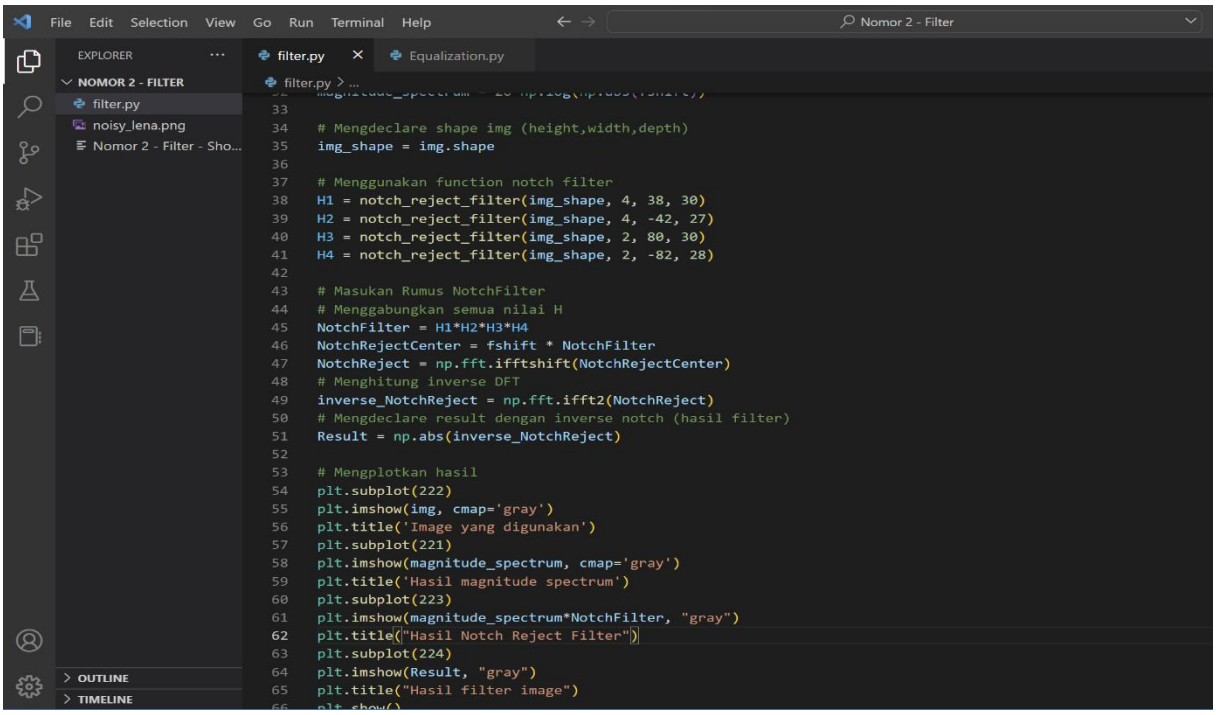
Hasil histogram image matching



2. Filter

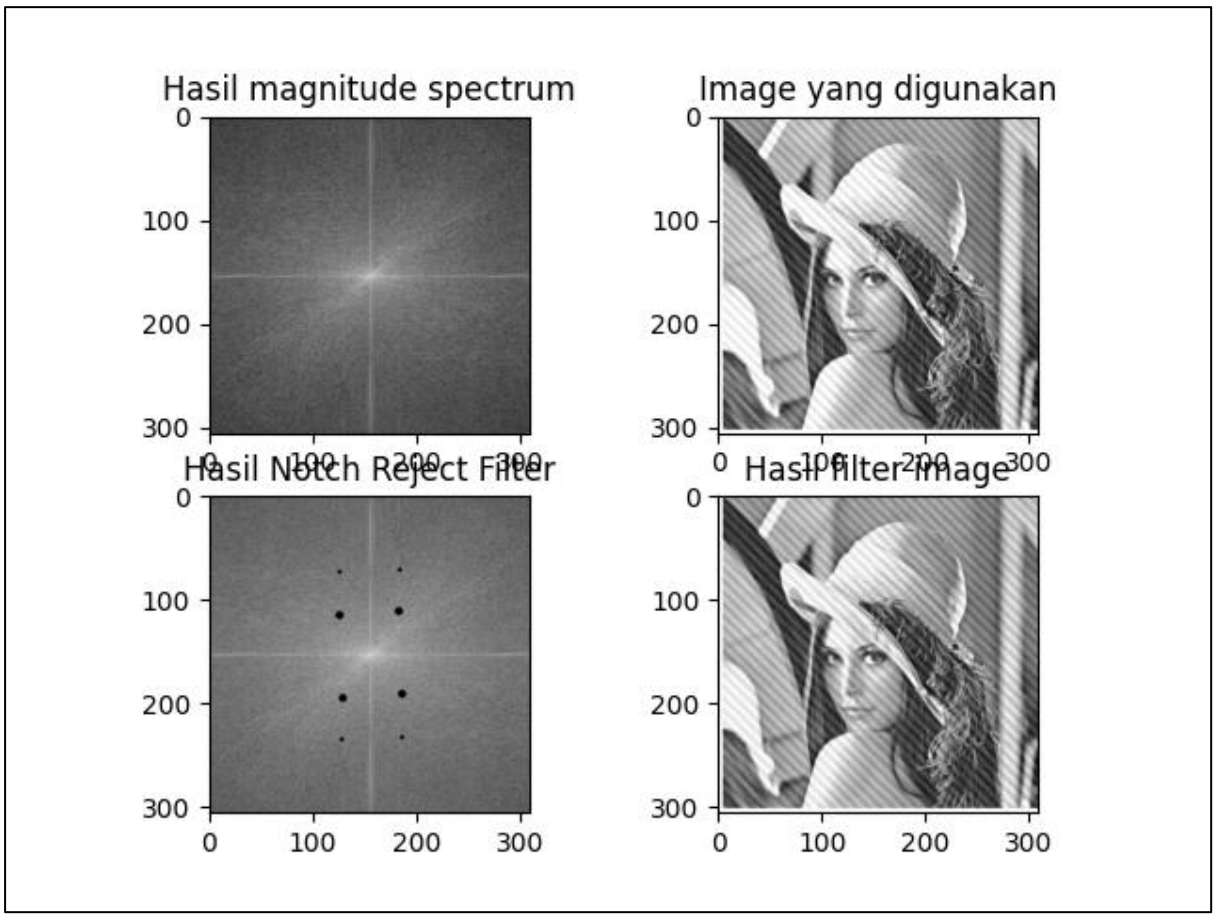


```python
# Daniel Chandra - 2301888631
# Nomor 2 Filter
# Import library yang akan digunakan
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Image yang akan digunakan
img = cv2.imread('noisy_lena.png', 0)

# Function notch filer
def notch_reject_filter(shape, d0=9, u_k=0, v_k=0):
    P, Q = shape
    # Initialize filter dengan numpy zeros
    H = np.zeros((P, Q))
    # Traverse melewati filter
    for u in range(0, P):
        for v in range(0, Q):
            # Mencari distance dari point D(u,v) ke center
            D_uv = np.sqrt((u - P / 2 + u_k) ** 2 + (v - Q / 2 + v_k) ** 2)
            D_muv = np.sqrt((u - P / 2 - u_k) ** 2 + (v - Q / 2 - v_k) ** 2)
            if D_uv <= d0 or D_muv <= d0:
                H[u, v] = 0.0
            else:
                H[u, v] = 1.0
    return H

# Memanggilkan fungsi fft untuk apply pada image
f = np.fft.fft2(img)
fshift = np.fft.fftshift(f)
phase_spectrumR = np.angle(fshift)
magnitude_spectrum = 20*np.log(np.abs(fshift))
```



```python
# Mengdeclare shape img (height,width,depth)
img_shape = img.shape

# Menggunakan function notch filter
H1 = notch_reject_filter(img_shape, 4, 38, 30)
H2 = notch_reject_filter(img_shape, 4, -42, 27)
H3 = notch_reject_filter(img_shape, 2, 80, 30)
H4 = notch_reject_filter(img_shape, 2, -82, 28)

# Masukan Rumus NotchFilter
# Menggabungkan semua nilai H
NotchFilter = H1*H2*H3*H4
NotchRejectCenter = fshift * NotchFilter
NotchReject = np.fft.ifftshift(NotchRejectCenter)
# Menghitung inverse DFT
inverse_NotchReject = np.fft.ifft2(NotchReject)
# Mengdeclare result dengan inverse notch (hasil filter)
Result = np.abs(inverse_NotchReject)

# Mengplotkan hasil
plt.subplot(222)
plt.imshow(img, cmap='gray')
plt.title('Image yang digunakan')
plt.subplot(221)
plt.imshow(magnitude_spectrum, cmap='gray')
plt.title('Hasil magnitude spectrum')
plt.subplot(223)
plt.imshow(magnitude_spectrum*NotchFilter, "gray")
plt.title("Hasil Notch Reject Filter")
plt.subplot(224)
plt.imshow(Result, "gray")
plt.title("Hasil filter image")
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

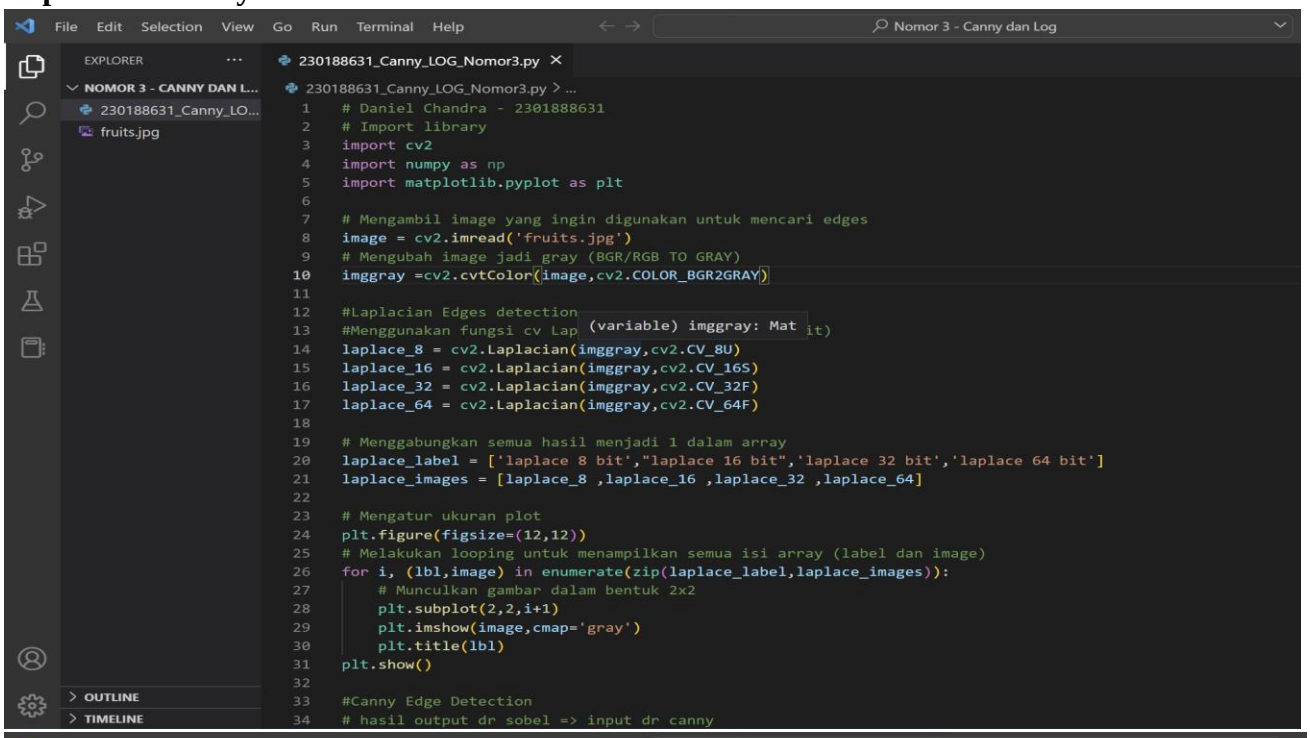
Langkah kerja non-maximum suppression:

- 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 histeresis merupakan sebuah teknik untuk automatic edges detection. Namun, menghitung thresholds yang cukup tinggi dan rendah menggunakan metode tanpa pengawasan tetap menjadi masalah, perhitungan histeresis 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



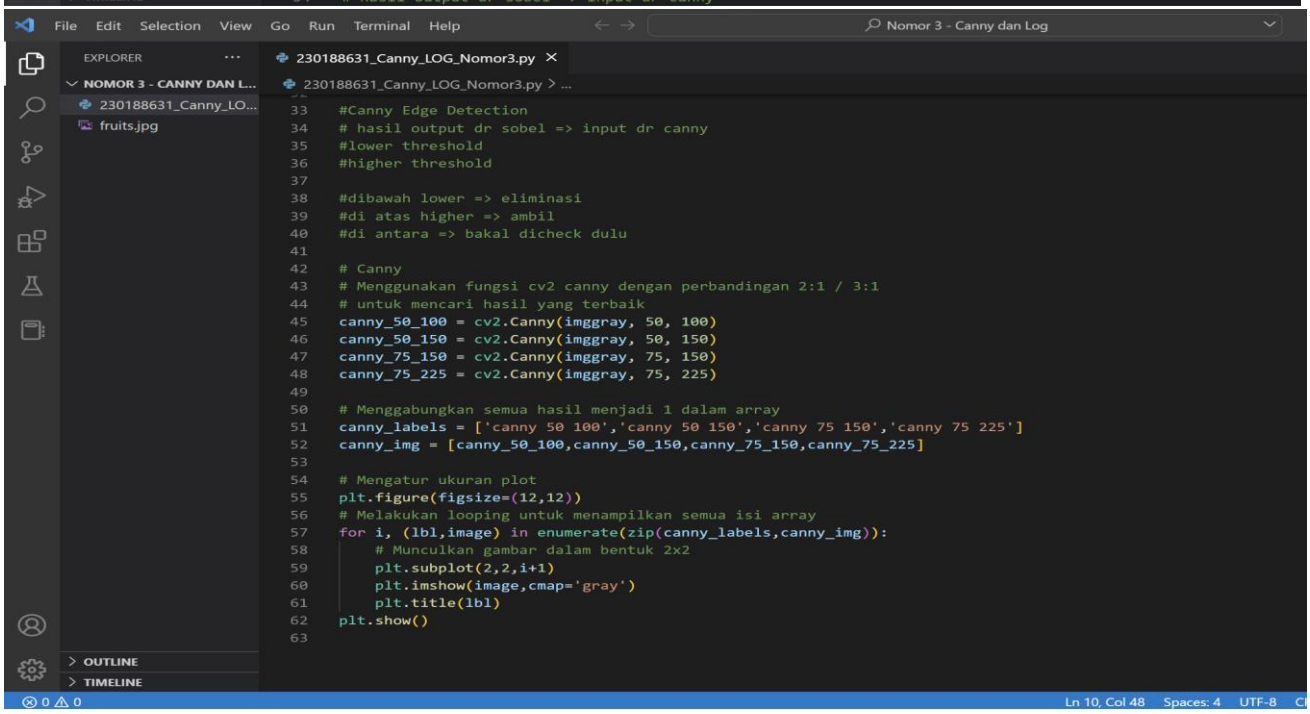
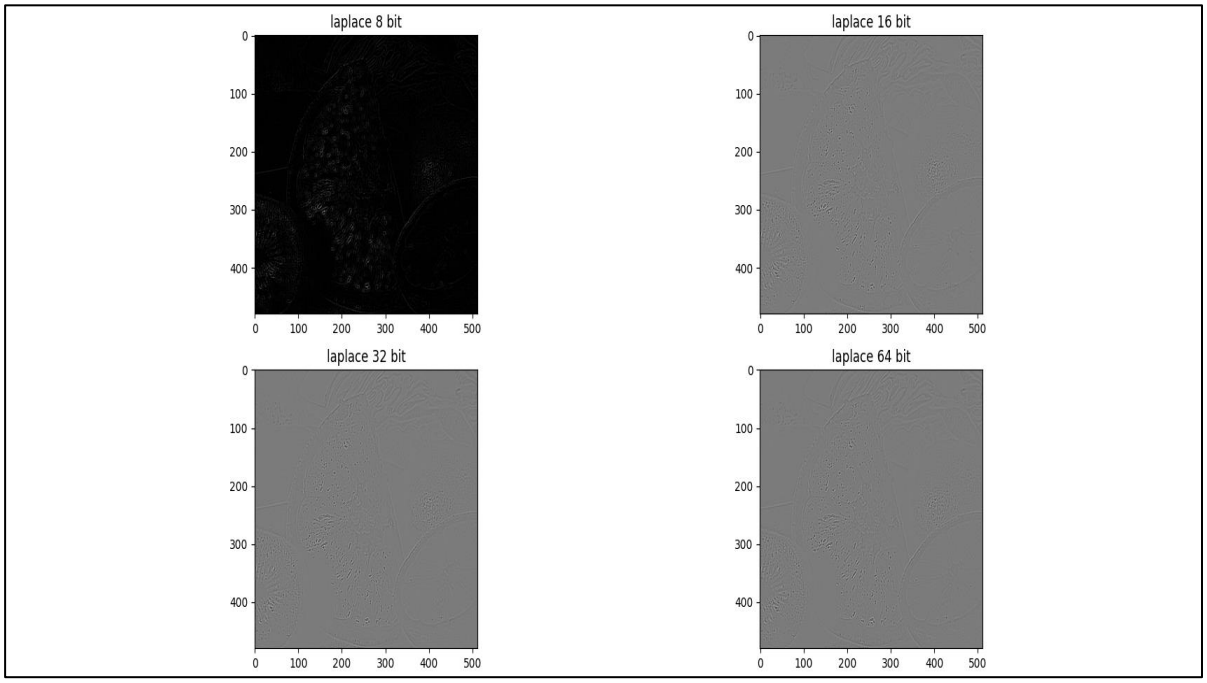


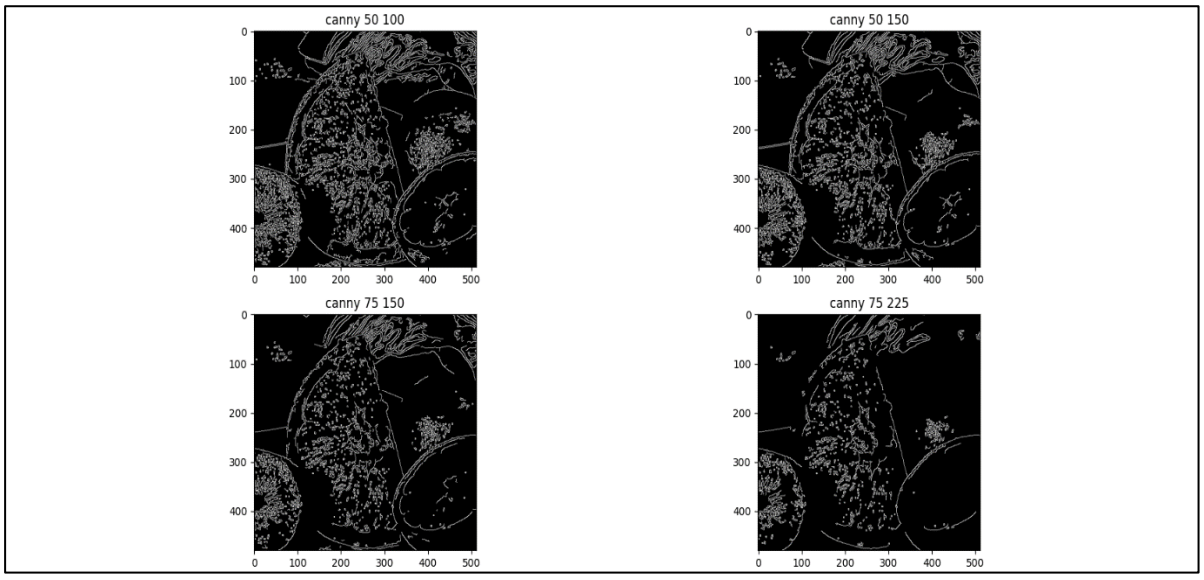
Image yang digunakan

Output

LOG

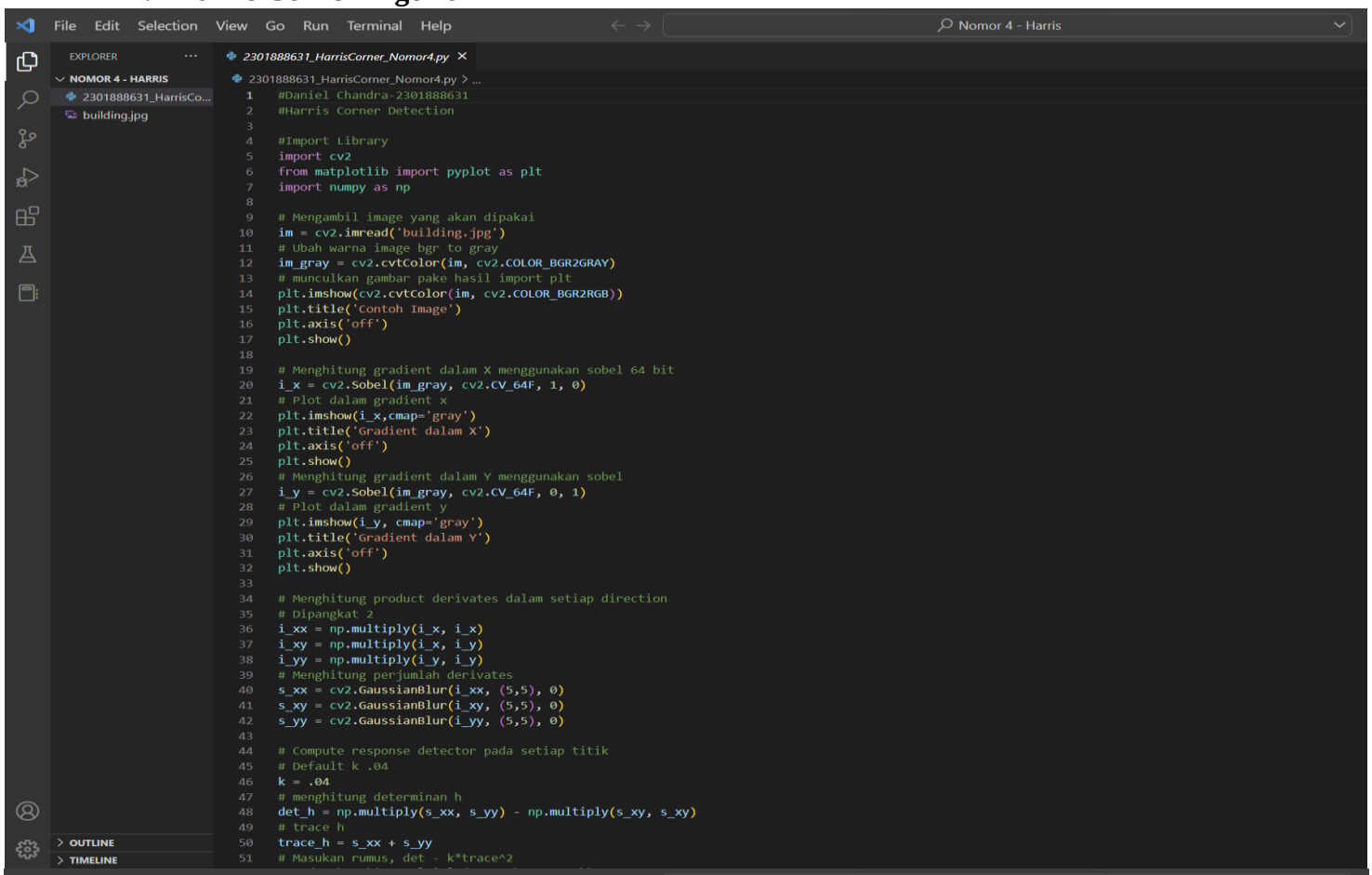


CANNY



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

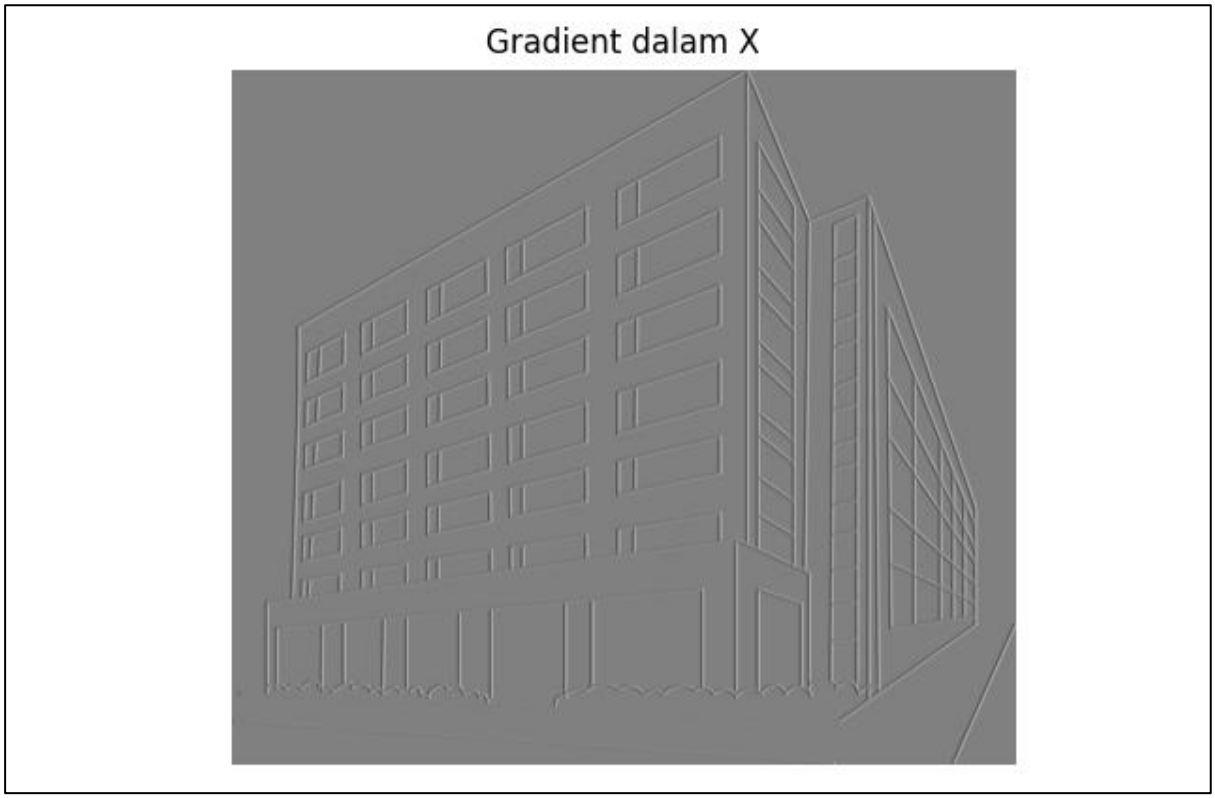
4. Harris Corner Algorithm



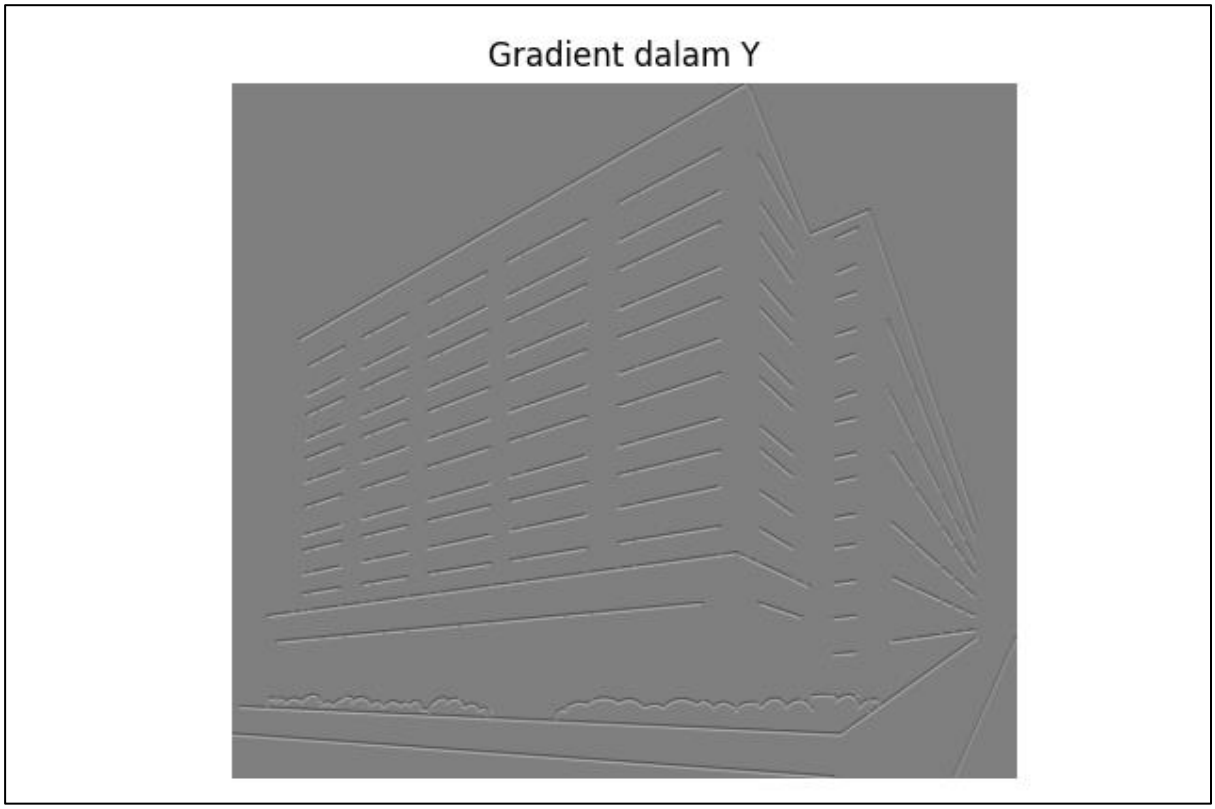

```python
#Daniel Chandra-2301888631
#Harris Corner Detection

#Import Library
import cv2
from matplotlib import pyplot as plt
import numpy as np

# Mengambil image yang akan dipakai
im = cv2.imread('building.jpg')
# Ubah warna image bgr to gray
im_gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
# munculkan gambar pake hasil import plt
plt.imshow(cv2.cvtColor(im, cv2.COLOR_BGR2RGB))
plt.title('Contoh Image')
plt.axis('off')
plt.show()

# Menghitung gradient dalam X menggunakan sobel 64 bit
i_x = cv2.Sobel(im_gray, cv2.CV_64F, 1, 0)
# Plot dalam gradient x
plt.imshow(i_x,cmap='gray')
plt.title('Gradient dalam X')
plt.axis('off')
plt.show()
# Menghitung gradient dalam Y menggunakan sobel
i_y = cv2.Sobel(im_gray, cv2.CV_64F, 0, 1)
# Plot dalam gradient y
plt.imshow(i_y, cmap='gray')
plt.title('Gradient dalam Y')
plt.axis('off')
plt.show()

# Menghitung product derivates dalam setiap direction
# Dipangkat 2
i_xx = np.multiply(i_x, i_x)
i_xy = np.multiply(i_x, i_y)
i_yy = np.multiply(i_y, i_y)
# Menghitung perjumlah derivates
s_xx = cv2.GaussianBlur(i_xx, (5,5), 0)
s_xy = cv2.GaussianBlur(i_xy, (5,5), 0)
s_yy = cv2.GaussianBlur(i_yy, (5,5), 0)

# Compute response detector pada setiap titik
# Default k .04
k = .04
# menghitung determinan h
det_h = np.multiply(s_xx, s_yy) - np.multiply(s_xy, s_xy)
# trace h
trace_h = s_xx + s_yy
# Masukan rumus, det - k*trace^2
R = det_h - k*np.multiply(trace_h, trace_h)
# Threshold
#Tweak
ratio = .2
thresh = abs(R) > ratio * abs(R).max()
# Plot hasil threshold
plt.imshow(thresh, cmap='gray')
plt.title('Thresholded Image')
plt.axis('off')
plt.show()

#non-maximal supression
# Index dari non zero values
non_zero_cords = np.nonzero(thresh)

# Tupled koordinat
tuple_cords = [(i, j) for i,j in zip(non_zero_cords[0], non_zero_cords[1])]
# Value koordinat
values = [abs(R)[i, j] for i, j in tuple_cords]
# Sort the koordinats
sorted_cords = [tuple_cords[i] for i in np.argsort(values)[::-1]]
# Set Distance
distance = 10

# Cords nms
# Mencari corner
nms_cords = []
nms_cords.append(sorted_cords[0])
for cord in sorted_cords:
    for nms_cord in nms_cords:
        if abs(cord[0]-nms_cord[0]) < distance and abs(cord[1]-nms_cord[1]) < distance:
            break
    else:
        nms_cords.append(cord)
print ("sebelum NMS = {} ".format(sum(thresh)))
print ("setelah NMS = {} ".format(len(nms_cords)))


# Timpa gambar dengan titik merah (merah pada setiap corner image)
for nms_cord in nms_cords:
    cv2.circle(im, (nms_cord[1], nms_cord[0]), 4, (0, 0, 255), -1)
# tampilkan hasil akhir result

# Gambarnya agak lama muncul setelah dirun, harus ditunggu beberapa detik
plt.figure(figsize=(12,12))
plt.imshow(cv2.cvtColor(im, cv2.COLOR_BGR2RGB))
plt.title('Hasil Harris Corner (harus di zoom baru nampak titik merah pada setiap korner)')
plt.axis('off')
plt.show()
```

Output Harris Corner algorithm

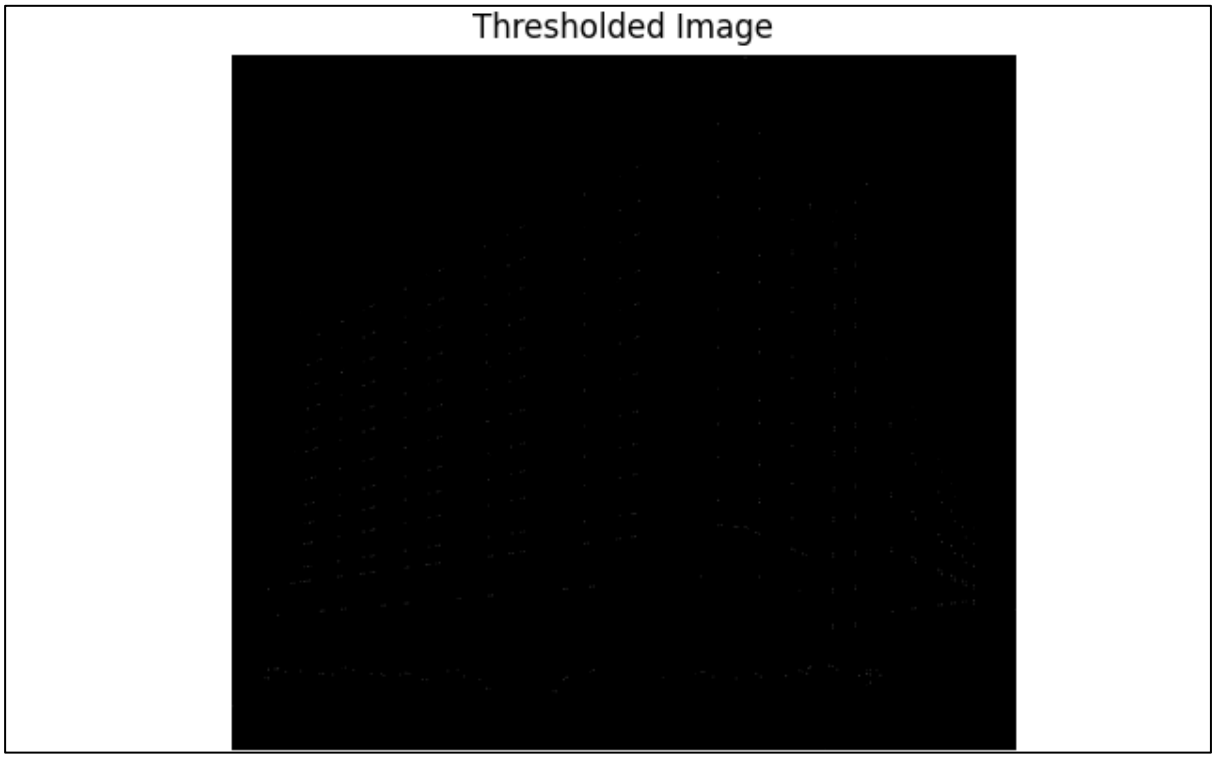
Gradient dalam x



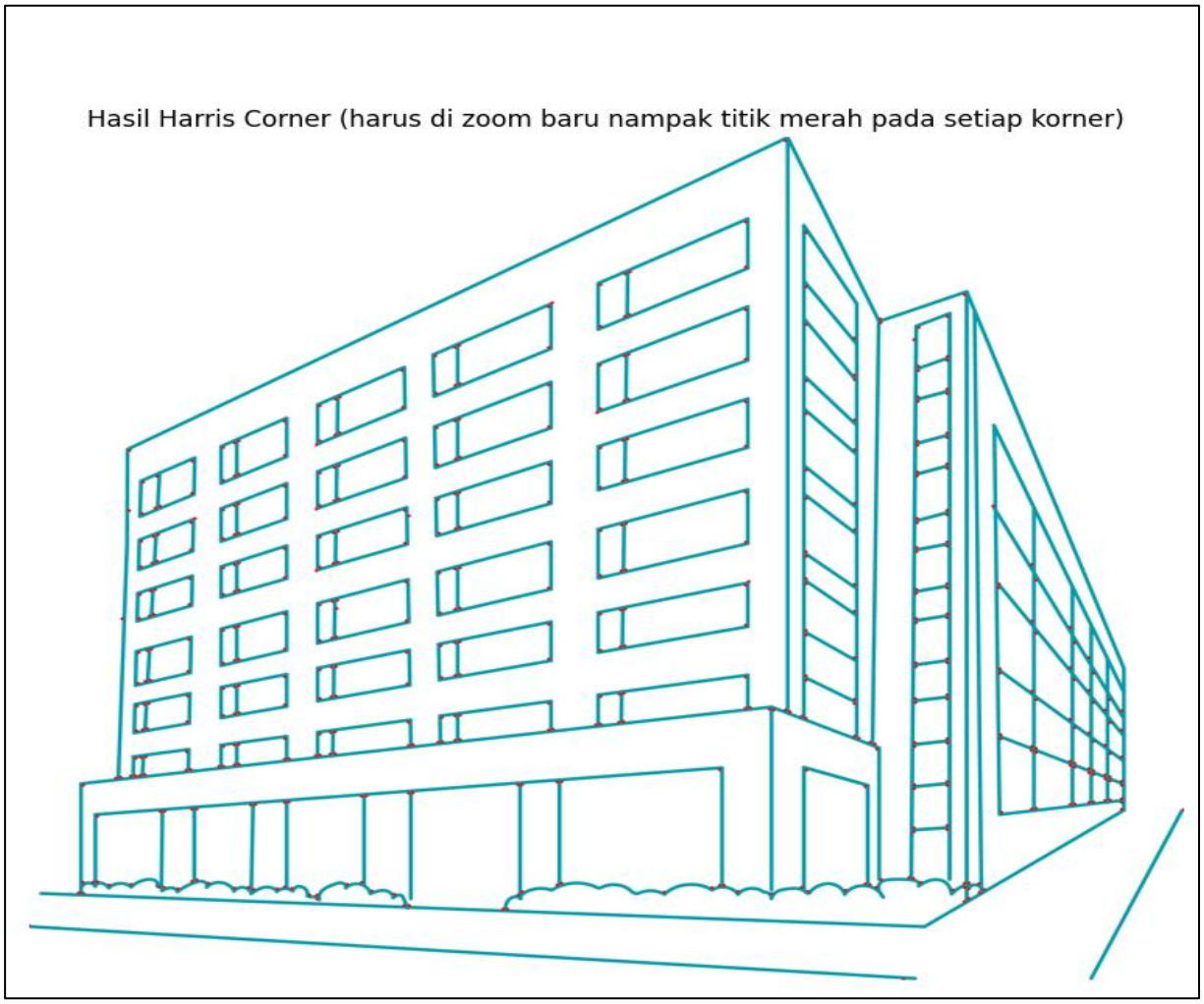
Gradient dalam y



Threshold

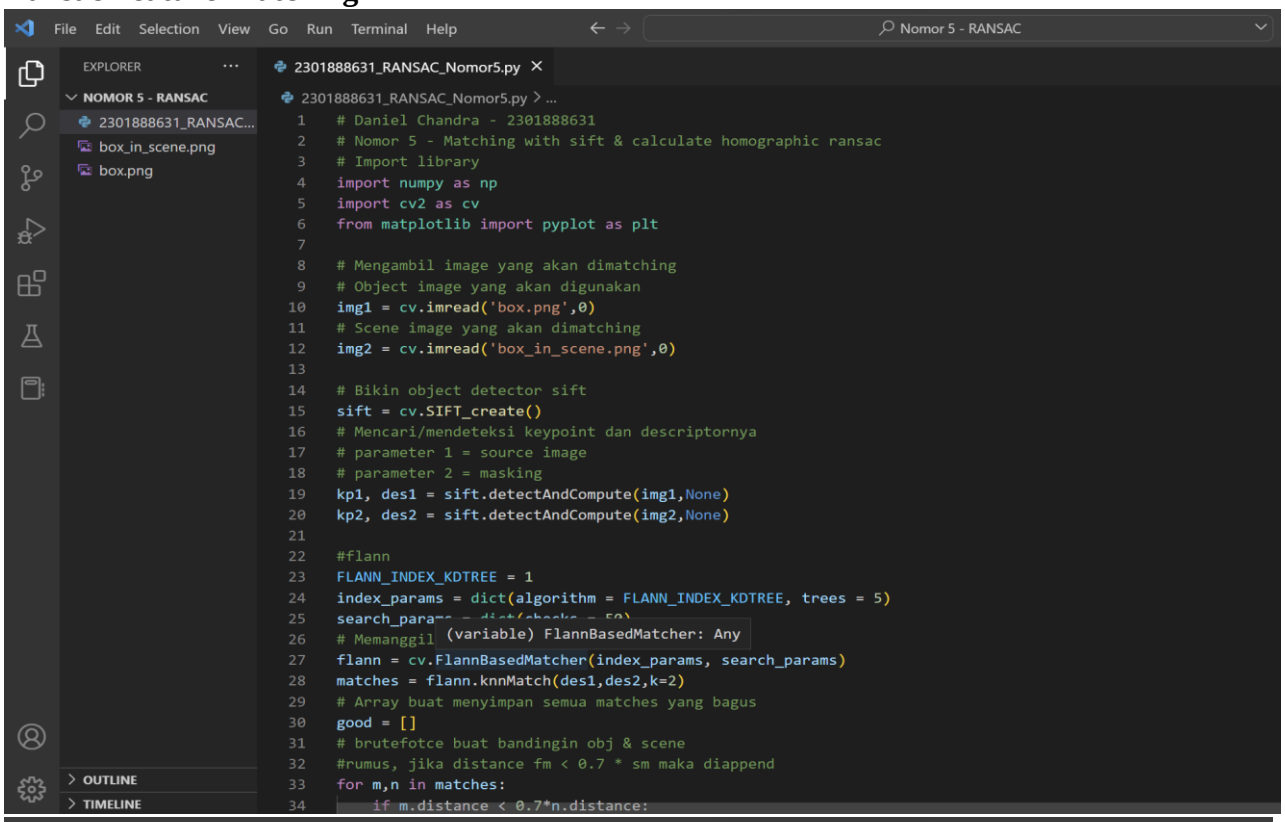


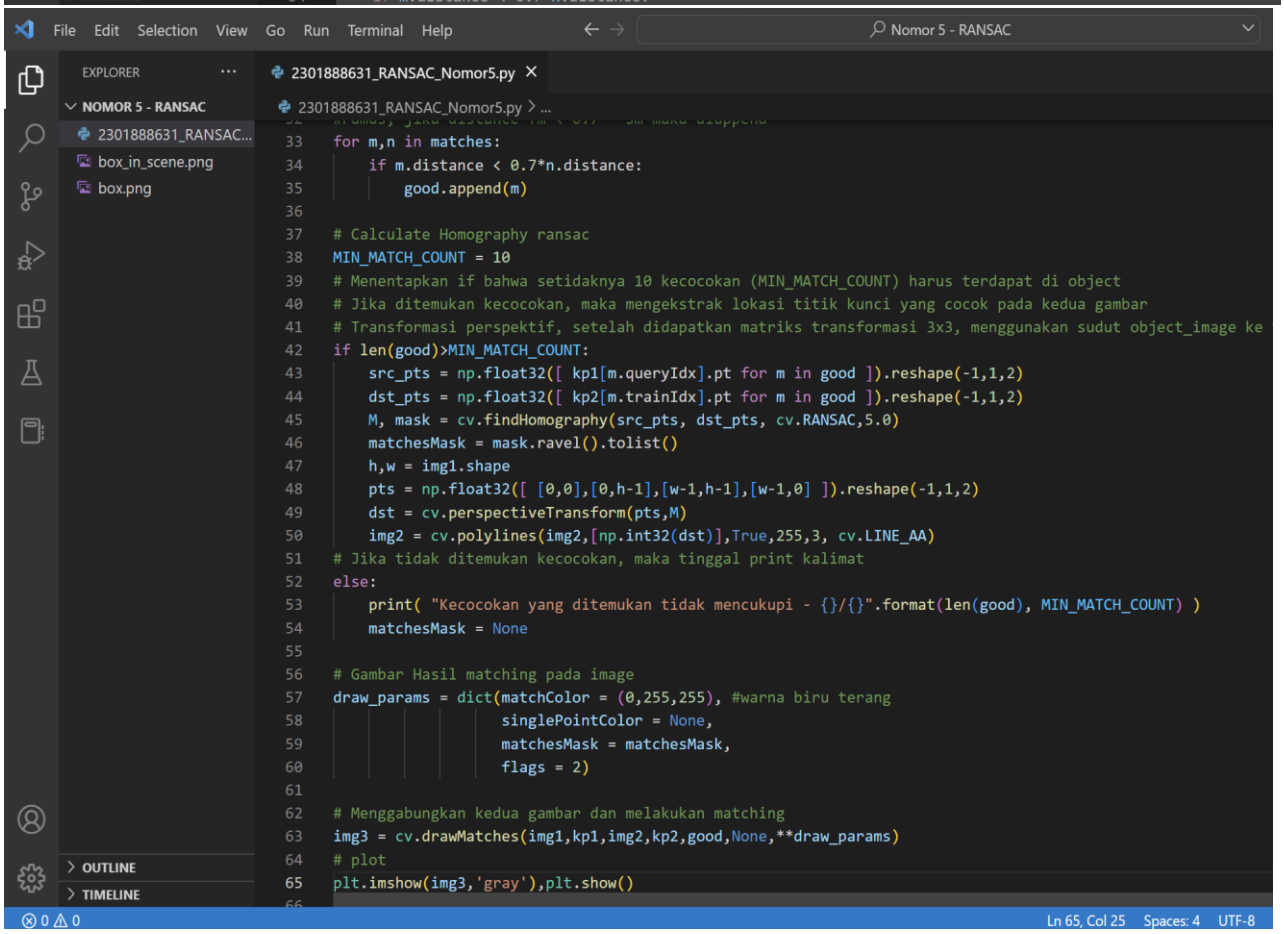
Hasil Corner Detection image



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

5. Ransac Feature Matching





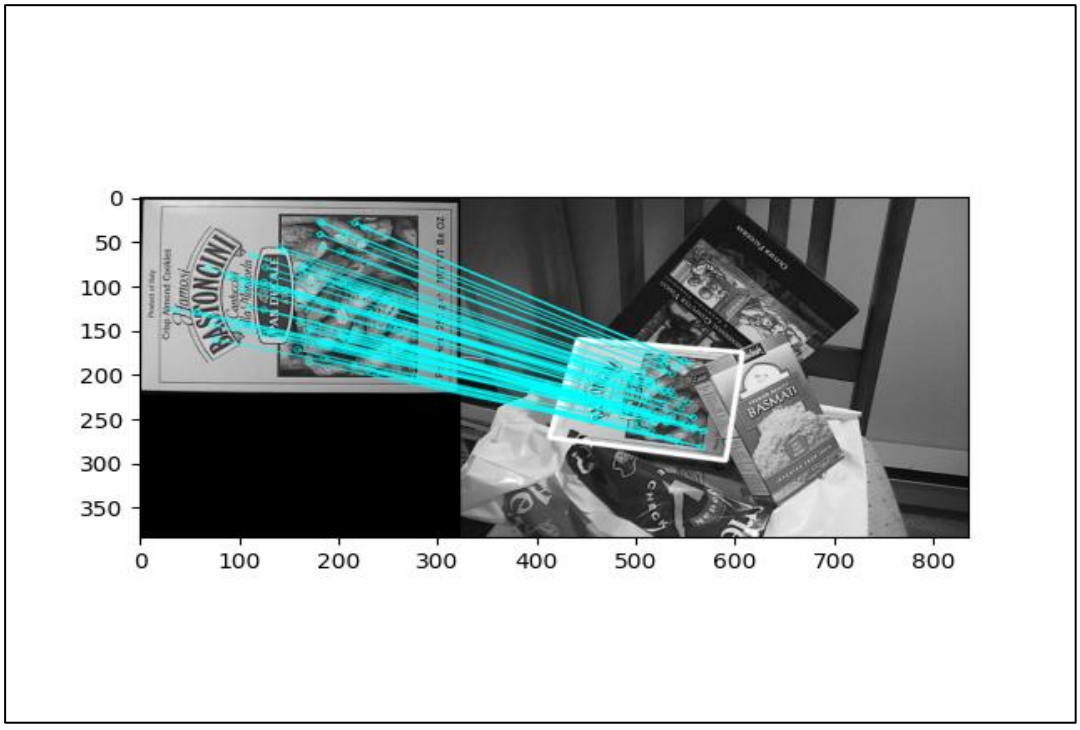
Verified by,
Henry Lucky (D6660) and sent to Program on Oct 31, 2022

Image yang digunakan

Object

Scene

Output



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