



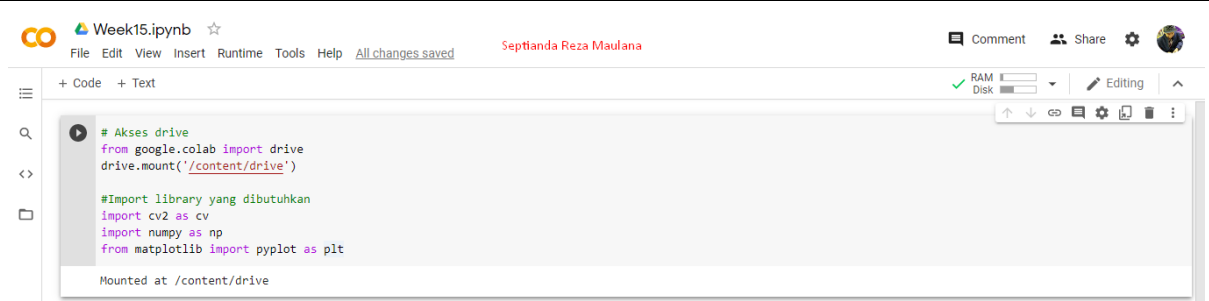
Jurusan Teknologi Informasi Politeknik Negeri Malang  
**Tugas Minggu-15 : Metode Deteksi Objek: Template Matching**  
**Mata Kuliah Pengolahan Citra dan Visi Komputer**  
Pengampu: Dr. Eng CAHYA RAHMAD., ST.,M.KOM.  
*Juni 2021*

---

### Tujuan

1. Mahasiswa dapat memahami konsep deteksi objek.
2. Mahasiswa mampu memahami metode yang dapat digunakan dalam proses deteksi objek.
3. Mahasiswa dapat mengimplementasikan beberapa metode dalam proses deteksi objek menggunakan Python pada Google Colab.

### Tugas Praktikum

Langkah	Keterangan
1	<p>Buka <a href="https://colab.research.google.com/">https://colab.research.google.com/</a>. Setelah dipastikan bahwa google Colab terhubung dengan Github Anda, buat notebook baru dan beri nama “Week11.ipynb”. Kemudian import beberapa library dan akses folder yang ada di Drive Anda dengan cara sebagai berikut.</p>  <pre># Akses drive from google.colab import drive drive.mount('/content/drive')  #Import library yang dibutuhkan import cv2 as cv import numpy as np from matplotlib import pyplot as plt  Mounted at /content/drive</pre>

Implementasikan 6 metode template matching pada OpenCV dengan menggunakan gambar cats\_and\_bunnies.jpg dan cat2\_template.jpg sebagai templatanya. Sehingga menghasilkan keluaran seperti berikut:

```
# No 2. Implementasikan 6 metode template matching pada OpenCV
img = cv.imread('/content/drive/MyDrive/PCVK/Images/cats_and_bunnies.jpg',0)
img = cv.cvtColor(img, cv.COLOR_BGR2RGB)
template = cv.imread('/content/drive/MyDrive/PCVK/Images/cat2_template.jpg',0)
template = cv.cvtColor(template, cv.COLOR_BGR2RGB)

# All the 6 methods for comparison in a list
methods = ['cv.TM_CCOEFF', 'cv.TM_CCOEFF_NORMED', 'cv.TM_CCORR',
           'cv.TM_CCORR_NORMED', 'cv.TM_SQDIFF', 'cv.TM_SQDIFF_NORMED']

for meth in methods:
    img_copy = img.copy()
    method = eval(meth)

    # Apply template Matching
    res = cv.matchTemplate(img_copy, template, method)
    min_val, max_val, min_loc, max_loc = cv.minMaxLoc(res)

    # If the method is TM_SQDIFF or TM_SQDIFF_NORMED, take minimum
    if method in [cv.TM_SQDIFF, cv.TM_SQDIFF_NORMED]:
        top_left = min_loc
    else:
        top_left = max_loc

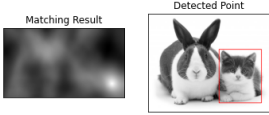
    height, width, channels = template.shape
    bottom_right = (top_left[0]+width, top_left[1]+height)

    bottom_right = (top_left[0]+width, top_left[1]+height)
    cv.rectangle(img_copy, top_left, bottom_right, (255,0,0), 3)

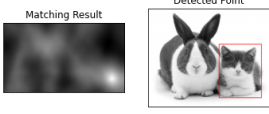
    plt.subplot(121), plt.imshow(res, cmap = 'gray')
    plt.title('Matching Result'), plt.xticks([], plt.yticks([]))
    plt.subplot(122), plt.imshow(img_copy, cmap='gray')
    plt.title('Detected Point'), plt.xticks([], plt.yticks([]))
    plt.suptitle(meth)

    plt.show()
```

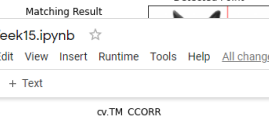
cv.TM\_CCOEFF




cv.TM\_CCOEFF\_NORMED

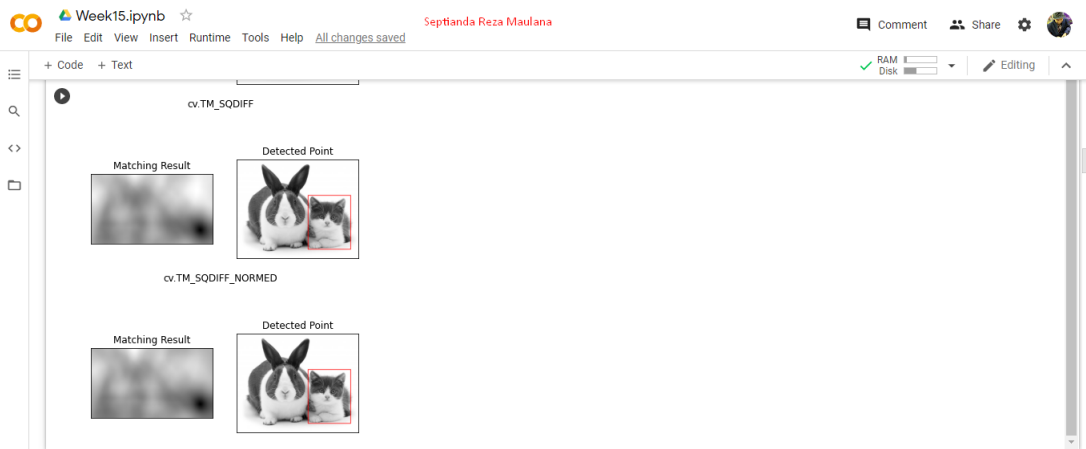


cv.TM\_CCORR



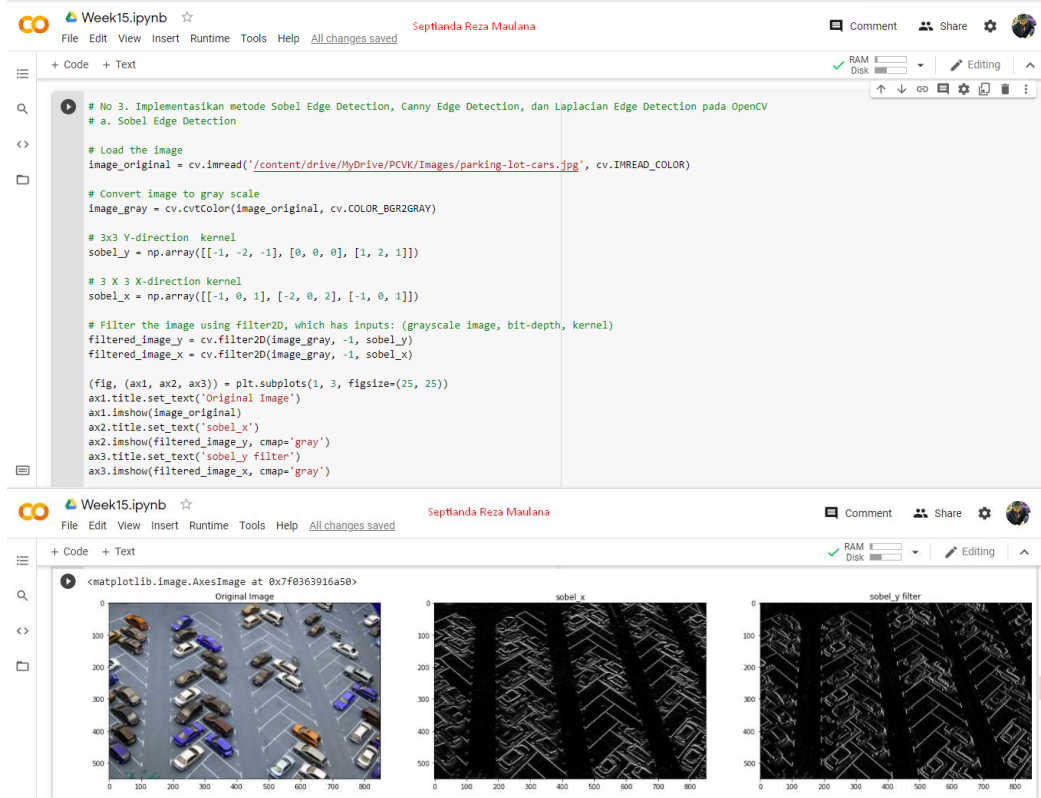
cv.TM\_CCORR\_NORMED



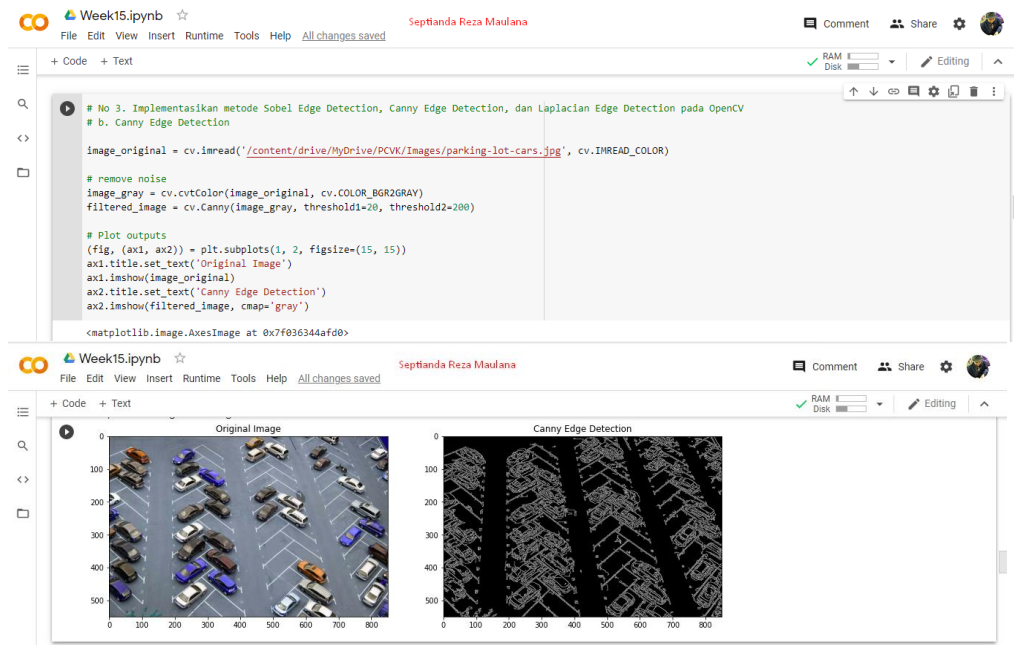


Implementasikan metode Sobel Edge Detection, Canny Edge Detection, dan Laplacian Edge Detection pada OpenCV dengan menggunakan gambar parking-lot-cars.jpg, sehingga menghasilkan luaran sebagai berikut:

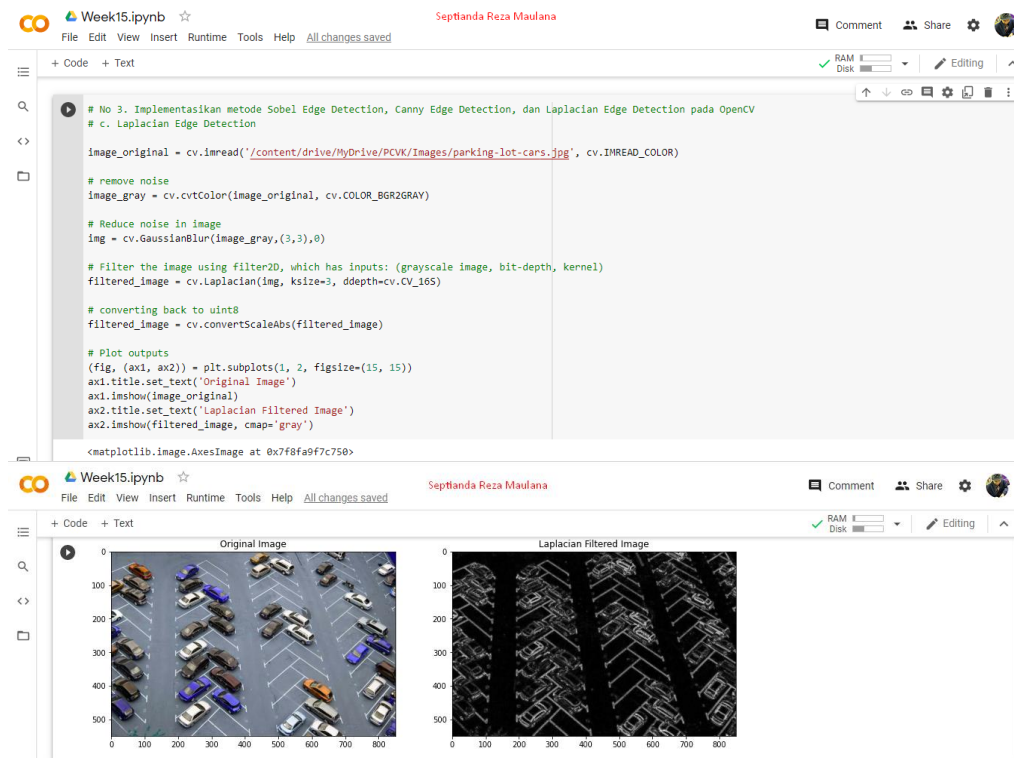
### a. Sobel Edge Detection



## b. Canny Edge Detection

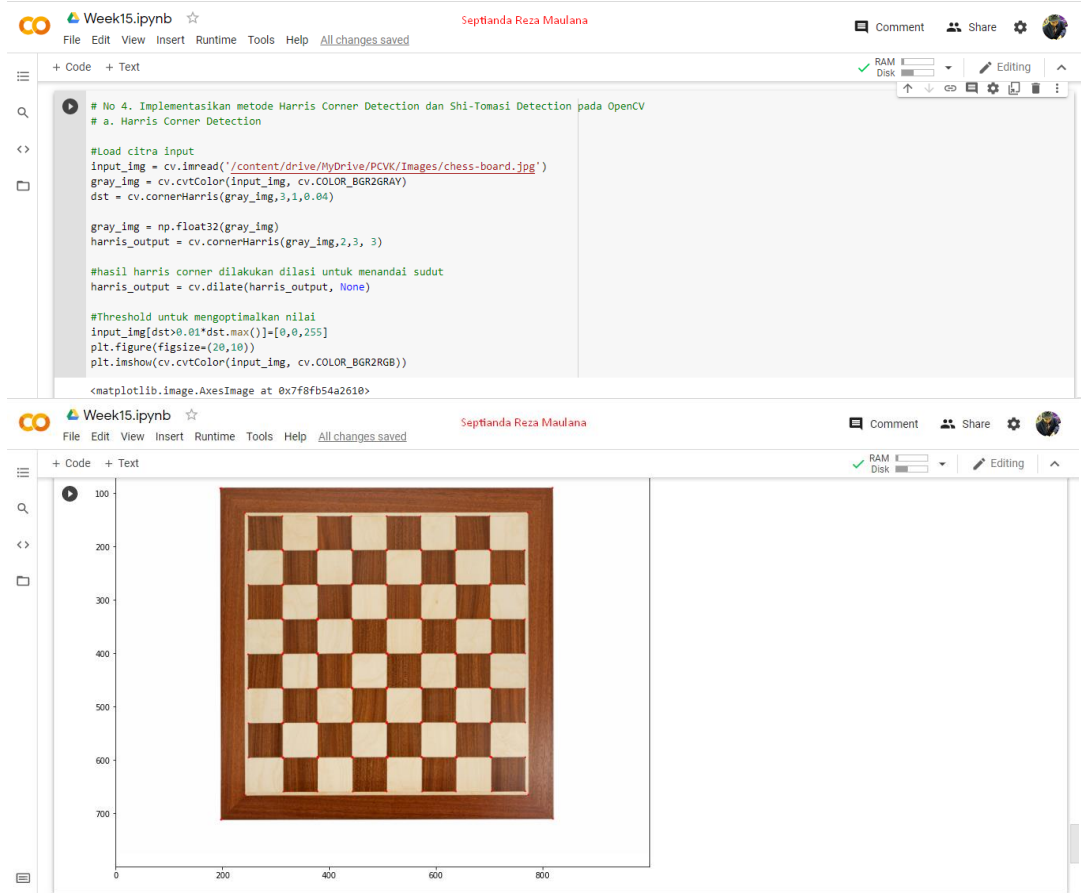


## c. Laplacian Filtered Image



Implementasikan metode Harris Corner Detection dan Shi-Tomasi Detection pada OpenCV dengan menggunakan gambar chess-board.jpg, sehingga menghasilkan luaran sebagai berikut:

a. Harris Corner Detection



b. Shi-Tomasi Detection





Implementasikan metode Hough Transform pada OpenCV dengan menggunakan gambar sudoku.jpg. Tahapan proses grid detection sesuai yang terdapat pada ulasan teori, sehingga menghasilkan luaran sebagai berikut:

```
input_img = cv.imread('/content/drive/MyDrive/PCVK/Images/sudoku.jpg')
img_gray = cv.cvtColor(input_img, cv.COLOR_BGR2GRAY)

edges = cv.Canny (img_gray, 90, 150, apertureSize = 3)

kernel = np.ones((3,3), np.uint8)
edges = cv.dilate(edges,kernel,iterations = 1)

kernel = np.ones((5,5), np.uint8)
edges = cv.erode (edges, kernel, iterations = 1)

lines = cv.HoughLines(edges,1,np.pi/180,150)

if not lines.any():
    print('No lines were found')
    exit()

if filter:
    rho_threshold = 15
    theta_threshold = 0.1

similar_lines = {i : [] for i in range(len(lines))}
for i in range(len(lines)):
    for j in range (len (lines)):
        rho_i, theta_i = lines[i][0]
        rho_j, theta_j = lines[j][0]
        if abs(rho_i - rho_j) < rho_threshold and abs(theta_i - theta_j) < theta_threshold:
            similar_lines[i].append(j)

indices = [i for i in range(len(lines))]
indices.sort(key=lambda x: len(similar_lines[x]))

line_flags = len(lines) * [True]
for i in range(len(lines) - 1):
    if not line_flags [indices[i]]:
        continue

    for j in range(i + 1, len(lines)):
        if not line_flags[indices[j]]:
            continue

        rho_i, theta_i = lines[indices[i]][0]
        rho_j, theta_j = lines[indices[j]][0]
        if abs(rho_i - rho_j) < rho_threshold and abs(theta_i - theta_j) < theta_threshold:
            line_flags [indices[j]] = False

print('number of Hough lines: ', len(lines))
```

```
Week15.ipynb ☆
File Edit View Insert Runtime Tools Help All changes saved
Septianda Reza Maulana
Comment Share
RAM
Disk
Editing

+ Code + Text

line_flags [indices[j]] = False
print('number of Hough lines:', len(lines))

filtered_lines = []

if filter:
    for i in range(len(lines)):
        if line_flags[i]:
            filtered_lines.append(lines[i])
    print('Number of filtered lines:', len(filtered_lines))
else:
    filtered_lines = lines

for line in filtered_lines:
    rho, theta = line[0]
    a = np.cos(theta)
    b = np.sin(theta)
    x0 = a*rho
    y0 = b*rho
    x1 = int(x0 + 1000*(-b))
    y1 = int(y0 + 1000*(a))
    x2 = int(x0 - 1000*(-b))
    y2 = int(y0 - 1000*(a))

cv.line(input_img, (x1, y1), (x2, y2), (0,0,255),2)

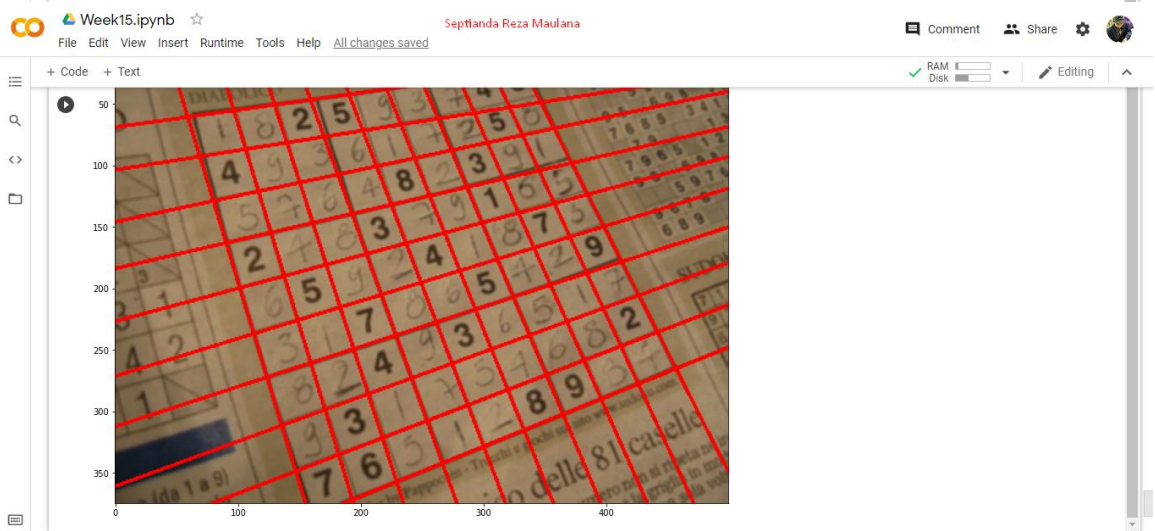
plt.figure(figsize=(20,10))
```

```
Week15.ipynb ☆
File Edit View Insert Runtime Tools Help All changes saved
Septianda Reza Maulana
Comment Share
RAM
Disk
Editing

+ Code + Text

plt.figure(figsize=(20,10))
plt.imshow(cv.cvtColor(input_img, cv.COLOR_BGR2RGB))

Number of filtered lines: 10
Number of filtered lines: 11
Number of filtered lines: 12
Number of filtered lines: 13
Number of filtered lines: 14
Number of filtered lines: 14
Number of filtered lines: 15
Number of filtered lines: 15
Number of filtered lines: 15
Number of filtered lines: 16
Number of filtered lines: 16
Number of filtered lines: 17
Number of filtered lines: 17
Number of filtered lines: 18
Number of filtered lines: 18
Number of filtered lines: 19
Number of filtered lines: 19
Number of filtered lines: 19
Number of filtered lines: 20
Number of filtered lines: 20
Number of filtered lines: 20
Number of filtered lines: 20
Number of filtered lines: 20
```



Implementasikan fungsi `findContours()` pada OpenCV untuk contour detection dengan menggunakan gambar `laptop.jpg`, sehingga menghasilkan luaran sebagai berikut:

6

