# WEEK 11

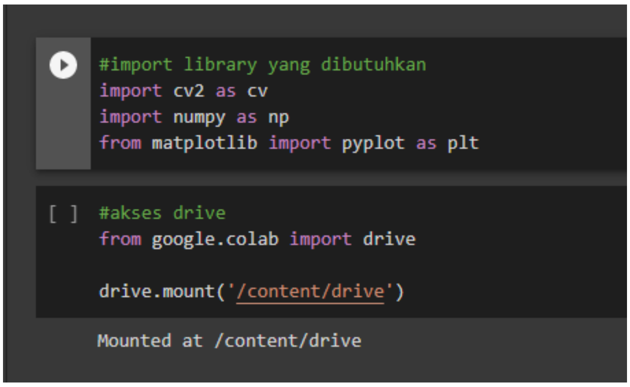
# OBJECT DETECTION METHOD

Arranged By:

Rajendra Rakha Arya Prabaswara
(1941720080/20)

PROGRAM STUDI D-IV TEKNIK INFORMATIKA
JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG

1. Go to https://colab.research.google.com/ . After making sure that Google Colab is connected to your Github, create a new notebook and name it "Week11.ipynb". Then import some libraries and access the folders on your Drive as follows.

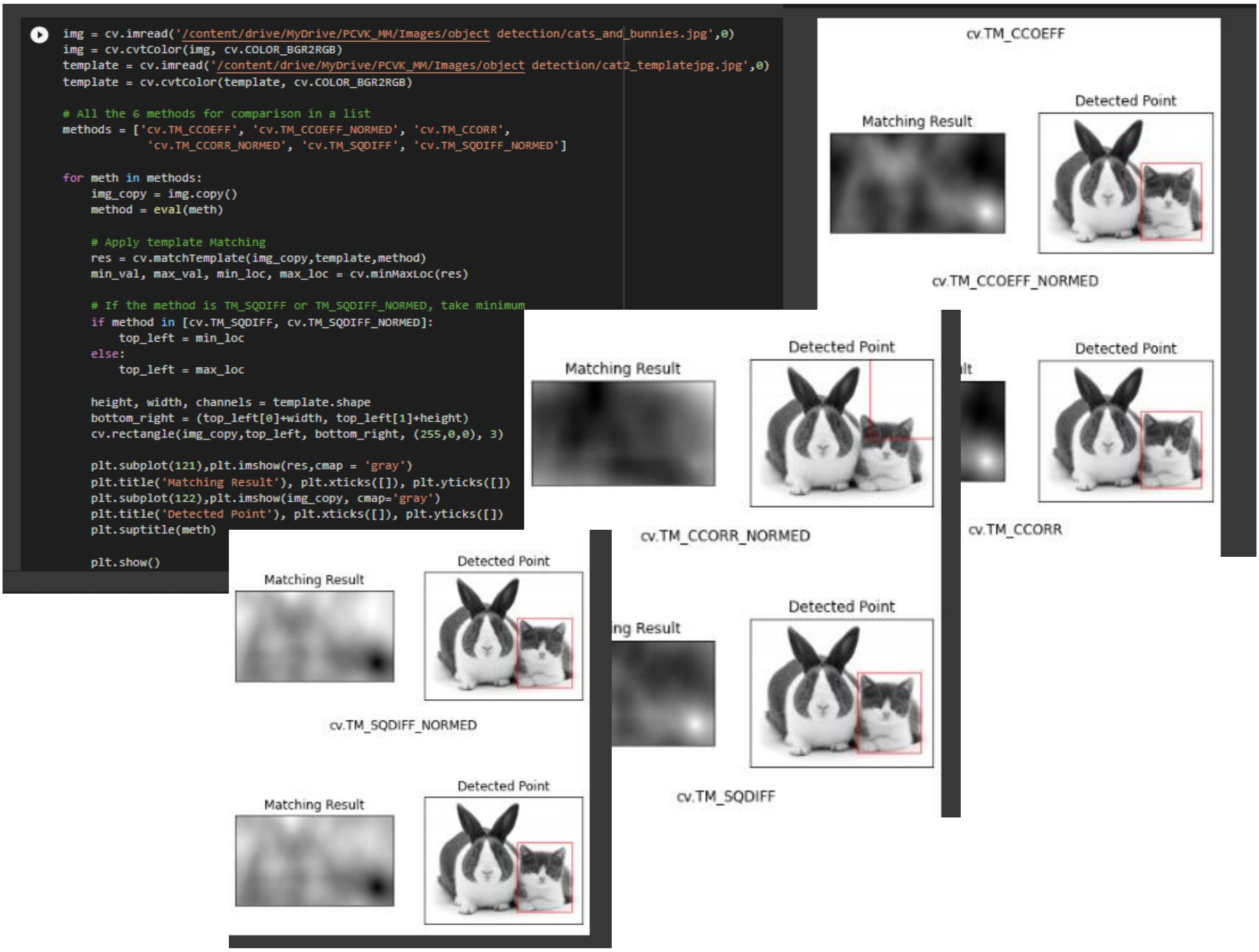


```
#import library yang dibutuhkan
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt
```

```
[ ] #akses drive
from google.colab import drive

drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

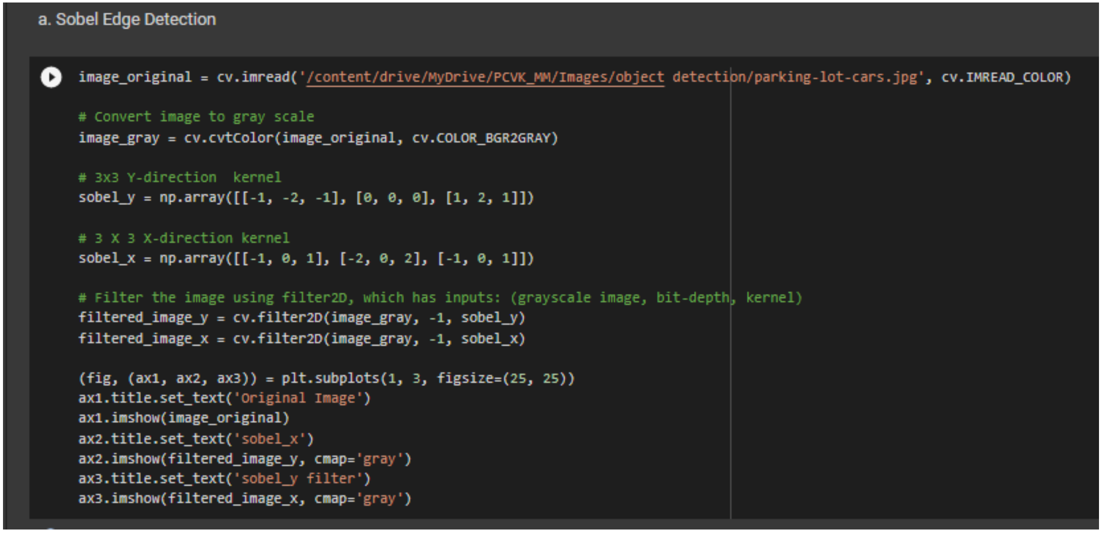
2. Implement 6 template matching methods in OpenCV using the cats_and_bunnies.jpg and cat2_templatejpg.jpg images as templates.



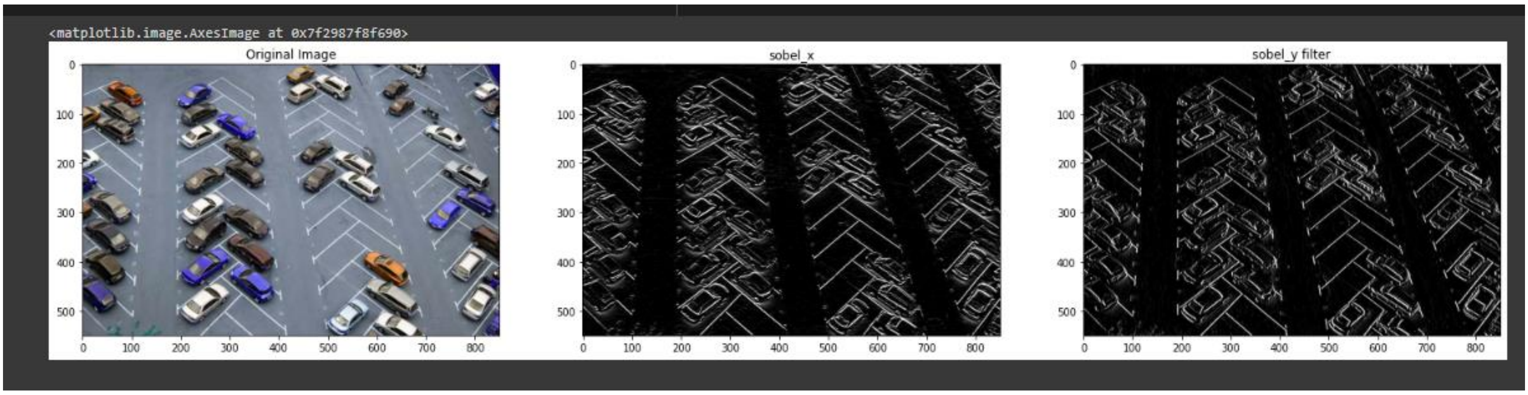
```
img = cv.imread('/content/drive/MyDrive/PCVK_MM/Images/object detection/cats_and_bunnies.jpg',0)
img = cv.cvtColor(img, cv.COLOR_BGR2RGB)
template = cv.imread('/content/drive/MyDrive/PCVK_MM/Images/object detection/cat2_templatejpg.jpg',0)
template = cv.cvtColor(template, cv.COLOR_BGR2RGB)

# 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)
    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()
```

3. Implement the Sobel Edge Detection, Canny Edge Detection, and Laplacian Edge Detection methods in OpenCV using the parking-lot-cars.jpg image, resulting in the following output:

### A. Sobel Edge Detection

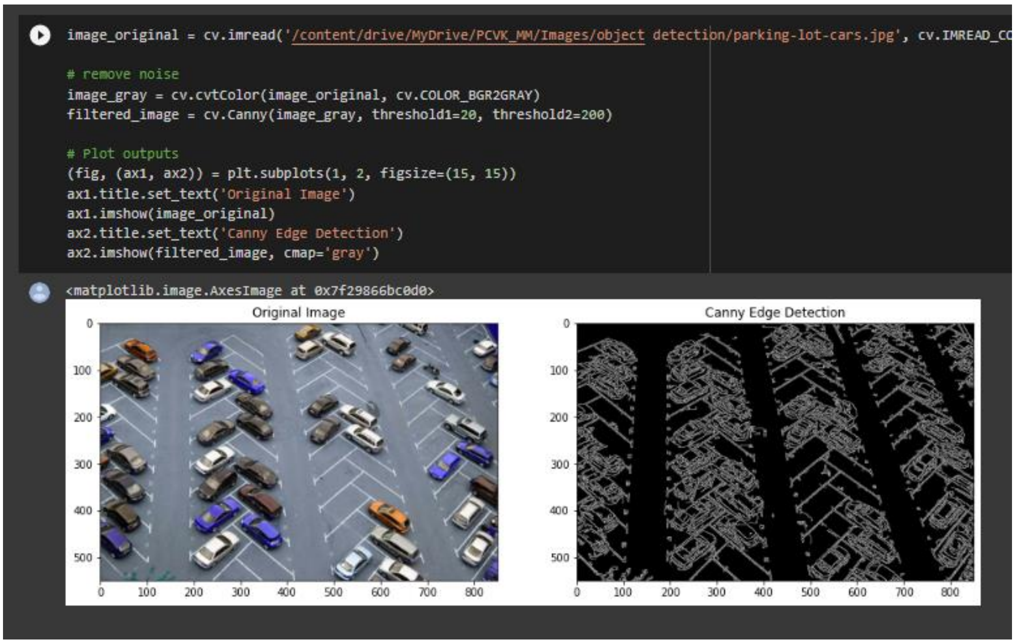

a. Sobel Edge Detection

```
image_original = cv.imread('/content/drive/MyDrive/PCVK_MM/Images/object detection/parking-lot-cars.jpg', cv.IMREAD_COLOR)

# Convert image to gray scale
image_gray = cv.cvtColor(image_original, cv.COLOR_BGR2GRAY)

# 3x3 Y-direction  kernel
sobel_y = np.array([[-1, -2, -1], [0, 0, 0], [1, 2, 1]])

# 3 X 3 X-direction kernel
sobel_x = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]])

# Filter the image using filter2D, which has inputs: (grayscale image, bit-depth, kernel)
filtered_image_y = cv.filter2D(image_gray, -1, sobel_y)
filtered_image_x = cv.filter2D(image_gray, -1, sobel_x)

(fig, (ax1, ax2, ax3)) = plt.subplots(1, 3, figsize=(25, 25))
ax1.title.set_text('Original Image')
ax1.imshow(image_original)
ax2.title.set_text('sobel_x')
ax2.imshow(filtered_image_y, cmap='gray')
ax3.title.set_text('sobel_y filter')
ax3.imshow(filtered_image_x, cmap='gray')
```



```
<matplotlib.image.AxesImage at 0x7f2987f8f690>
```

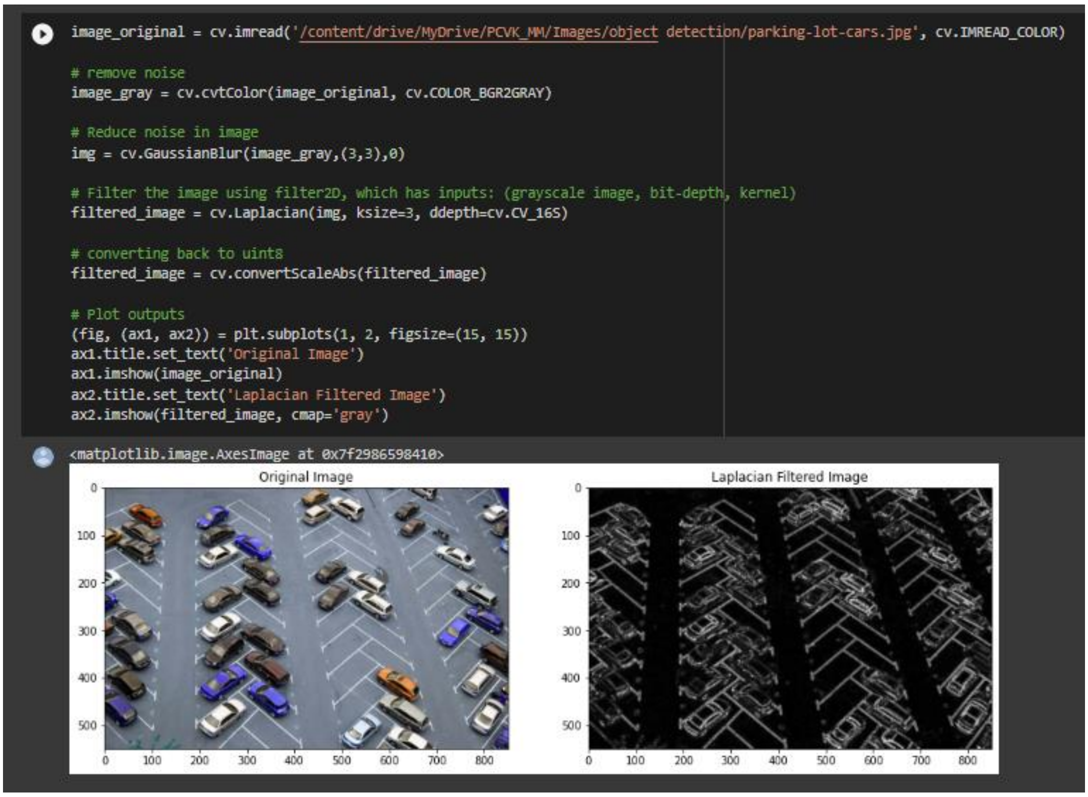
### B. Canny Edge Detection



```
image_original = cv.imread('/content/drive/MyDrive/PCVK_MM/Images/object detection/parking-lot-cars.jpg', cv.IMREAD_CO

# remove noise
image_gray = cv.cvtColor(image_original, cv.COLOR_BGR2GRAY)
filtered_image = cv.Canny(image_gray, threshold1=20, threshold2=200)

# Plot outputs
(fig, (ax1, ax2)) = plt.subplots(1, 2, figsize=(15, 15))
ax1.title.set_text('Original Image')
ax1.imshow(image_original)
ax2.title.set_text('Canny Edge Detection')
ax2.imshow(filtered_image, cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x7f29866bc0d0>
```

### C. Laplacian Edge Detection

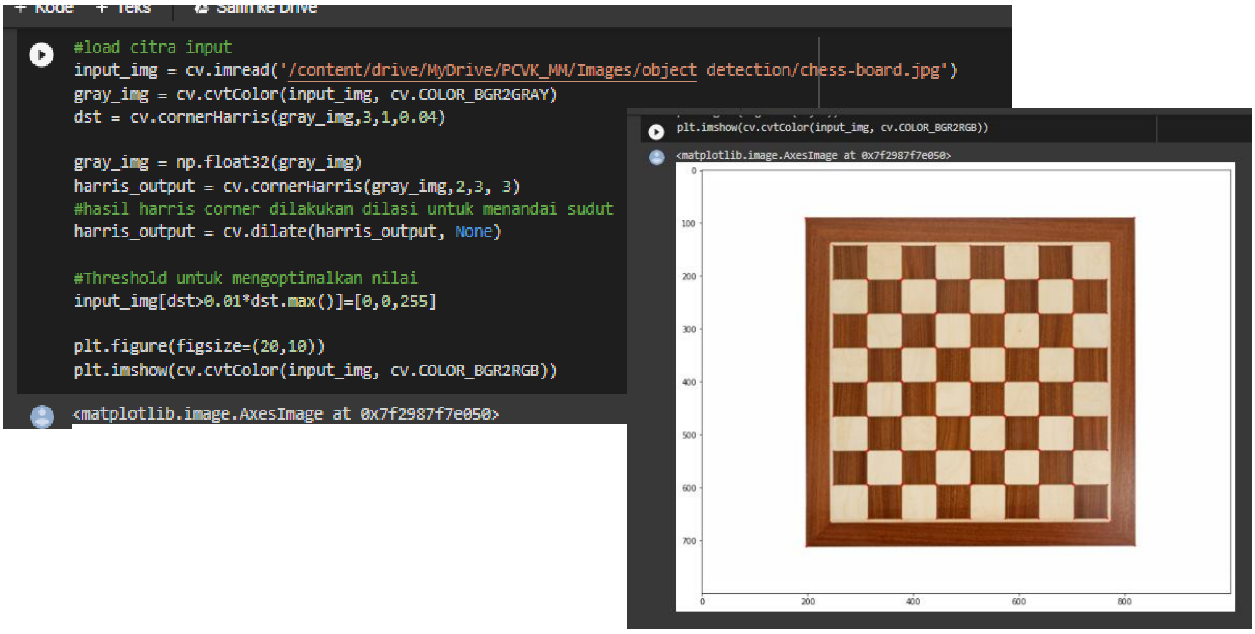


```
image_original = cv.imread('/content/drive/MyDrive/PCVK_MM/Images/object detection/parking-lot-cars.jpg', cv.IMREAD_COLOR)

# remove noise
image_gray = cv.cvtColor(image_original, cv.COLOR_BGR2GRAY)

# Reduce noise in image
img = cv.GaussianBlur(image_gray,(3,3),0)

# Filter the image using filter2D, which has inputs: (grayscale image, bit-depth, kernel)
filtered_image = cv.Laplacian(img, ksize=3, ddepth=cv.CV_16S)

# converting back to uint8
filtered_image = cv.convertScaleAbs(filtered_image)

# Plot outputs
(fig, (ax1, ax2)) = plt.subplots(1, 2, figsize=(15, 15))
ax1.title.set_text('Original Image')
ax1.imshow(image_original)
ax2.title.set_text('Laplacian Filtered Image')
ax2.imshow(filtered_image, cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x7f2986598410>
```

4. Implement the Sobel Edge Detection, Canny Edge Detection, and Laplacian Edge Detection methods in OpenCV using the parking-lot-cars.jpg image, resulting in the following output:

### A. Harris Corner Detection



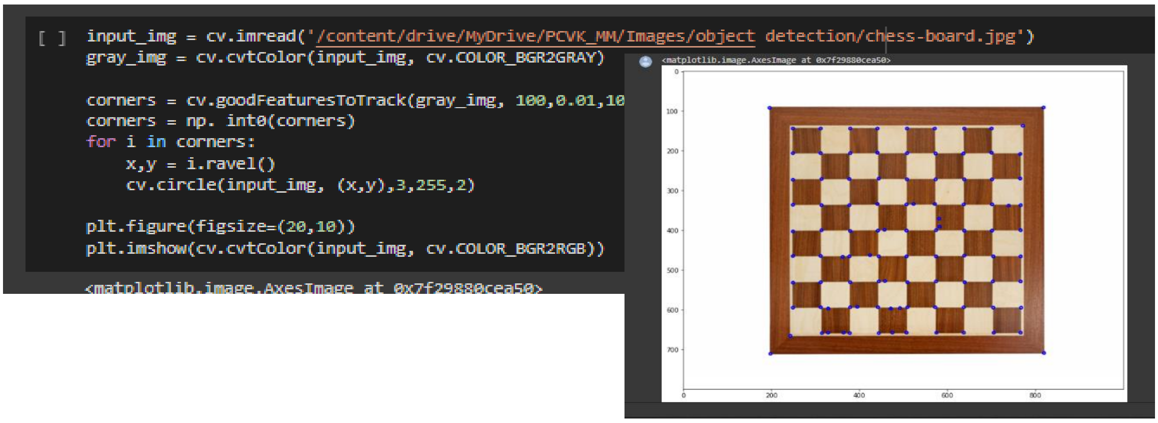
```
#load citra input
input_img = cv.imread('/content/drive/MyDrive/PCVK_MM/Images/object detection/chess-board.jpg')
gray_img = cv.cvtColor(input_img, cv.COLOR_BGR2GRAY)
dst = cv.cornerHarris(gray_img,3,1,0.04)

gray_img = np.float32(gray_img)
harris_output = cv.cornerHarris(gray_img,2,3, 3)
#hasil harris corner dilakukan dilasi untuk menandai sudut
harris_output = cv.dilate(harris_output, None)

#Threshold untuk mengoptimalkan nilai
input_img[dst>0.01*dst.max()]=[0,0,255]

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

```
<matplotlib.image.AxesImage at 0x7f2987f7e050>
```

```
plt.imshow(cv.cvtColor(input_img, cv.COLOR_BGR2RGB))
```

```
<matplotlib.image.AxesImage at 0x7f2987f7e050>
```

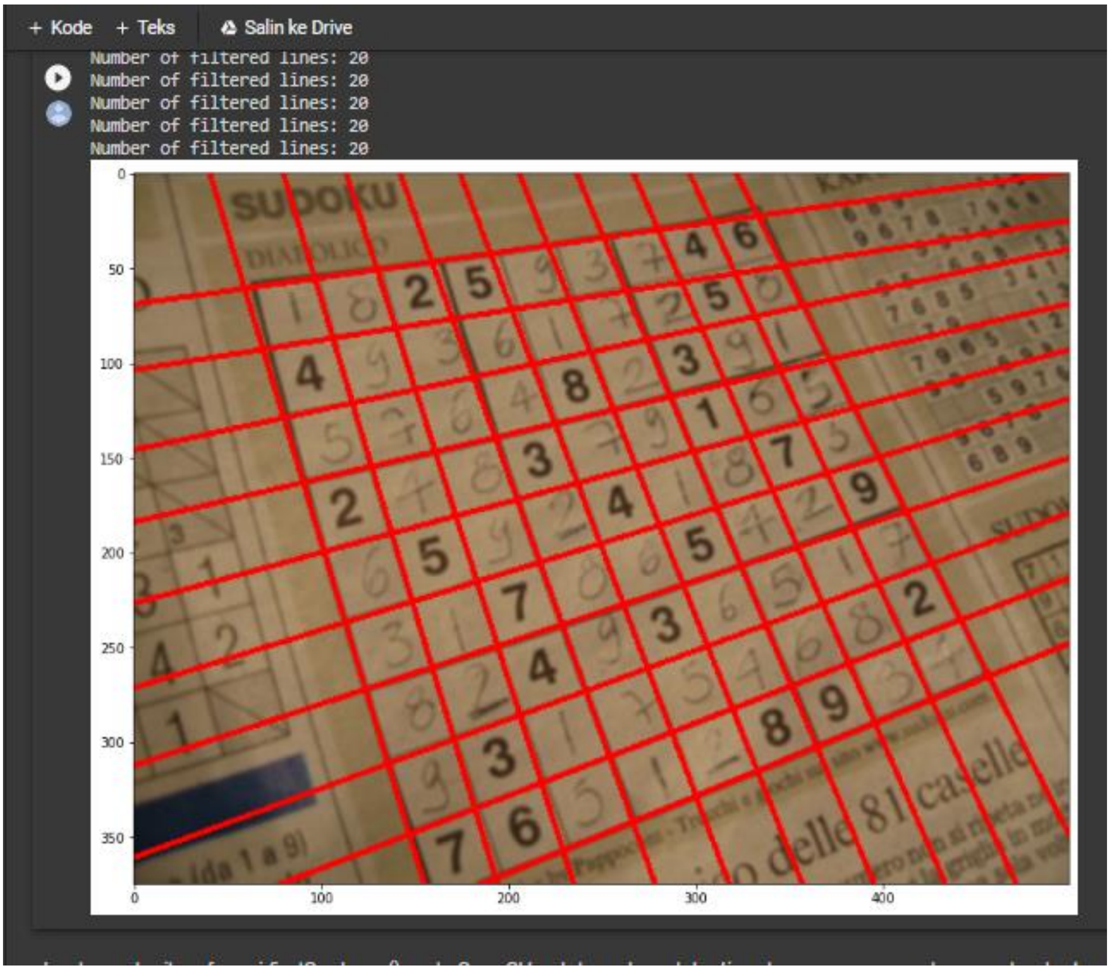
## B. Shi-Tomasi Detection



```
[ ] input_img = cv.imread('/content/drive/MyDrive/PCVK_MM/Images/object detection/chess-board.jpg')
    gray_img = cv.cvtColor(input_img, cv.COLOR_BGR2GRAY)

    corners = cv.goodFeaturesToTrack(gray_img, 100,0.01,10
    corners = np. int0(corners)
    for i in corners:
        x,y = i.ravel()
        cv.circle(input_img, (x,y),3,255,2)

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

    <matplotlib.image.AxesImage at 0x7f29880cea50>
```

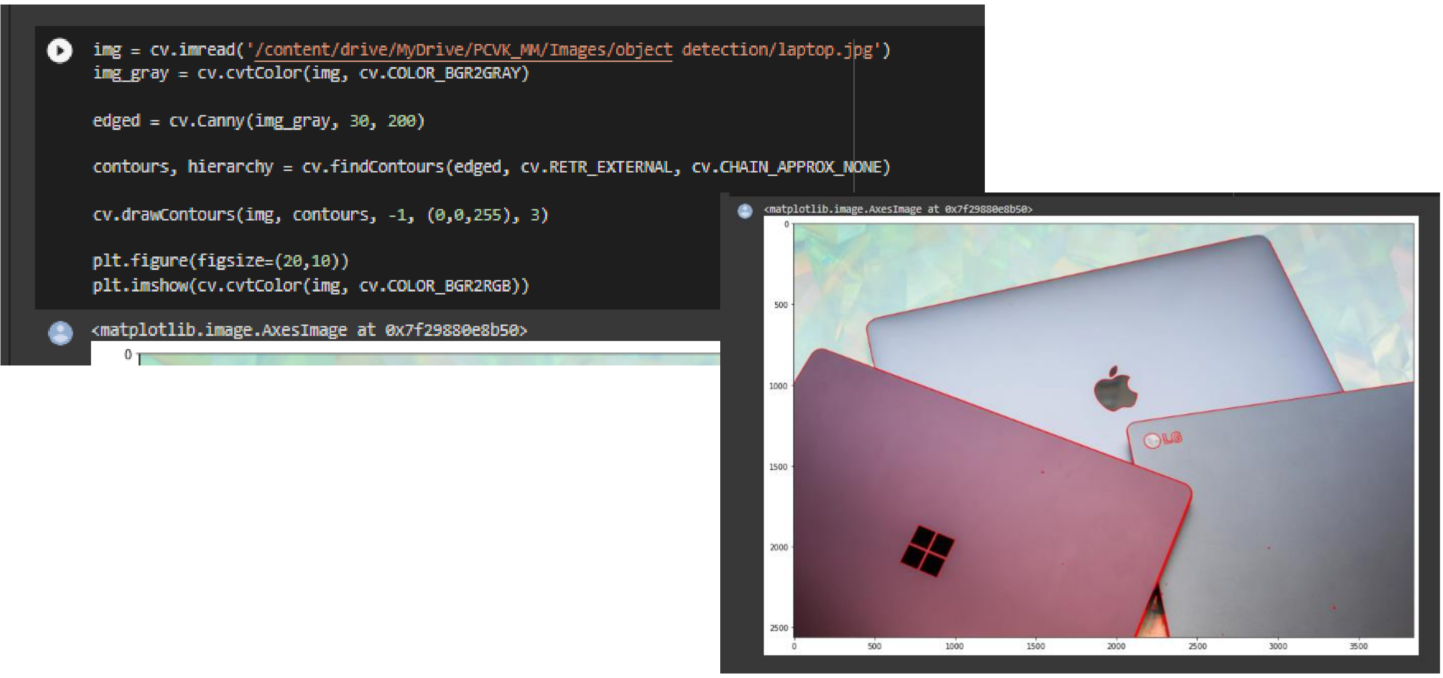
5. Implement the Hough Transform method in OpenCV using the sudoku.jpg image. The stages of the grid detection process are in accordance with those contained in the theoretical review, resulting in the following output:

```
input_img =
cv.imread('/content/drive/MyDrive/PCVK_MM/I
mages/object detection/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)):
        if i == j:
            continue

        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))

    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))
    plt.imshow(cv.cvtColor(input_img,
cv.COLOR_BGR2RGB))
```



```
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
```

6. Implement the findContours () function in OpenCV for contour detection using the laptop.jpg image, resulting in the following output:



```
img = cv.imread('/content/drive/MyDrive/PCVK_MM/Images/object detection/laptop.jpg')
img_gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)

edged = cv.Canny(img_gray, 30, 200)

contours, hierarchy = cv.findContours(edged, cv.RETR_EXTERNAL, cv.CHAIN_APPROX_NONE)

cv.drawContours(img, contours, -1, (0,0,255), 3)

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

```
<matplotlib.image.AxesImage at 0x7f29880e8b50>
```

# FULL CODE

https://github.com/Rjndrkha/PCVK_Genap_2022