การพัฒนาโปรแกรมประยุกต์และปัญญาประดิษฐ์ เพื่อการมองเห็นของเครื่องจักร
Computer Programing and Artificial Intelligence in Machine Vision

ชื่อ-สกุล : วราสิริ ลิ้มประเสริฐ B6214005

8/8 -- คำถามท้ายบทเพื่อทดสอบความเข้าใจ

Quiz_401 – กิจกรรมที่ 1/6

จงสร้างแบบจำลองการถดถอยเชิงเส้นอย่างง่ายสำหรับใช้ในการพยากรณ์ค่าเช่าต่อเดือน (บาท) จากขนาดของพื้นที่ (ตารางเมตร) โดยมีข้อมูลดังต่อไปไฟล์ Area_Rental.csv

โดยอยากทราบว่าพื้นที่ขนาด 50 ตารางเมตร จะต้องจ่ายค่าเช่าประมานเดือนล่ะเท่าไหร่?

In [1]:
```
# anaconda promt --> run as admin
# conda install pandas
# conda install scikit-learn

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%config InlineBackend.figure_format = 'retina'
```

In [2]:
```
bp = pd.read_csv('./data/Area_Rental.csv')
print(bp)
```

```
   Area  Rental
0  22.0    2000
1  23.5    2900
2  25.0    3200
3  26.5    3600
4  30.0    3800
5  32.0    4200
6  34.0    4600
7  36.5    5100
8  38.0    5700
9  42.0    6000
```

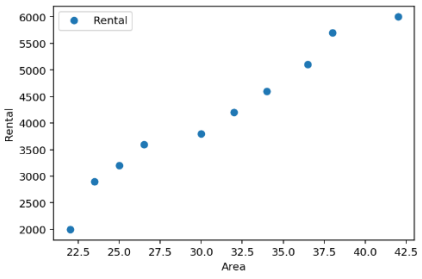
In [3]:
```
bp.corr()
```

Out[3]:

	Area	Rental
Area	1.000000	0.982158
Rental	0.982158	1.000000

In [4]:
```
bp.plot(x='Area', y='Rental',style='o')
plt.xlabel('Area')
plt.ylabel('Rental')
```

Out[4]: Text(0, 0.5, 'Rental')



In [5]:
```
from sklearn.linear_model import LinearRegression
```

In [6]:
```
xx = bp[['Area']]
yy = bp['Rental']
```

In [7]:
```
lrm = LinearRegression()
lrm.fit(xx,yy)
```

Out[7]: LinearRegression()

```
In [8]:
1 lrm.intercept_
```

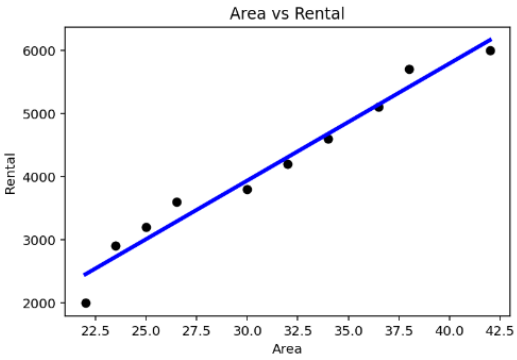
```
Out[8]: -1629.5818148125263
```

```
In [9]:
1 lrm.coef_
```

```
Out[9]: array([185.44690839])
```

```
In [10]:
1 predictions = lrm.predict(xx)
2 plt.scatter(xx, yy, color='black')
3 plt.plot(xx, predictions, color='blue', linewidth=3)
4 plt.title('Area vs Rental')
5 plt.xlabel('Area')
6 plt.ylabel('Rental')
```

```
Out[10]: Text(0, 0.5, 'Rental')
```



ดังนั้น ที่ Area = 50 เราจะต้องจ่ายค่าเช่า = 7,642.76 บาท

```
In [11]:
1 lrm.predict([[50]])
```

```
D:\angrybird\other\miniConda\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have
rRegression was fitted with feature names
  warnings.warn(
```

```
Out[11]: array([7642.76360492])
```

Quiz_402 – กิจกรรมที่ 2/6

จากตัวอย่างการใช้ KNN จงเปลี่ยน dataset เป็นไฟล์จาก digits_dataset2.zip โดยจะมีข้อมูลตัวเลขเพิ่มขึ้นมาเป็น 0-9 (จำนวนภาพ 500 ภาพต่อ 1 ตัวเลข)

- ทำการสร้างแบบจำลองด้วย KNN และทดสอบแบบจำลองด้วยการหาค่า accuracy
- สร้าง dataset ที่เป็น unknown ขึ้นมาอย่างน้อย 1 ตัวเลขด้วยการเขียนเอง จากนั้นทดสอบด้วยแบบจำลองที่สร้างขึ้น

In [1]:
```
# ตัวอย่าง ข้อมูลแบบภาพสี
import cv2
import numpy as np
img = cv2.imread('./image/digits_dataset/0_1.png')
img.shape, img
```

ขั้นตอนการ train model

In [3]:
```
train = np.array([])
for i in range(6):
    for j in range(1, 251):
        img   = cv2.imread('./image/digits_dataset/'+str(i)+'_'+str(j)+'.png')
        gray  = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        train = np.append(train, gray)
train = train.reshape(-1, 400).astype(np. float32)
print('Get Data for train - Ok')

test = np.array([])
for i in range(6):
    for j in range(251, 501):
        img = cv2.imread('./image/digits_dataset/' +str(i)+'_'+str(j)+'.png')
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        test = np.append(test, gray)
test = test.reshape (-1, 400).astype (np. float32)
print('Get Data for test - Ok')

```

```
Get Data for train - Ok
Get Data for test - Ok
```

In [4]:
```
k = np.arange(6)
train_labels = np.repeat(k, 250)[:,np.newaxis]
test_labels  = train_labels.copy ()

knn = cv2.ml.KNearest_create()
knn.train(train, cv2.ml.ROW_SAMPLE, train_labels)
ret, result, neighbours, dist = knn.findNearest(test, k=5)
result.shape, result

```

Out[4]:
```
((1500, 1),
 array([[0.],
        [0.],
        [0.],
        ...,
        [5.],
        [5.],
        [5.]], dtype=float32))
```

In [5]:
```
import sys
np.set_printoptions(threshold=sys.maxsize)

k = np.arange(6)
train_labels = np.repeat(k, 250)[:,np.newaxis]
test_labels  = train_labels.copy ()

knn = cv2.ml.KNearest_create()
knn.train(train, cv2.ml.ROW_SAMPLE, train_labels)
ret, result, neighbours, dist = knn.findNearest(test, k=5)
result.shape, result

```

In [6]:
```
matches  = result == test_labels
correct  = np.count_nonzero(matches)
accuracy = correct * 100.0 / result.size
print('accuracy = ', accuracy)

```

```
accuracy =  95.86666666666666
```

In [ ]:
```
#mydigit = cv2.imread('./image/unknown_x.png')
#mydigit = cv2.imread('./image/unknown_y.png')
#mydigit = cv2.imread('./image/unknown_z.png')
mydigit = cv2.imread('./image/unknown_aaa.png')

cv2.imshow('detected digit', mydigit)
mydigit_gray = cv2.cvtColor(mydigit, cv2.COLOR_BGR2GRAY)
mydigit_test = mydigit_gray.reshape (-1,400).astype(np. float32)
ret, result, neighbours, dist = knn. findNearest (mydigit_test, k=5)
print(ret)

font = cv2.FONT_HERSHEY_SIMPLEX
cv2.putText(mydigit, str(int(ret)), (0, 7), font, 0.3, (255, 0, 0), 1, cv2.LINE_AA)

cv2.waitKey (0)
cv2.destroyAllWindows ()

```

รูปที่ใช้ทดสอบ

ผลลัพธ์

In [*]:
```
#mydigit = cv2.imread('./image/unknown_x.png')
#mydigit = cv2.imread('./image/unknown_y.png')
#mydigit = cv2.imread('./image/unknown_z.png')
mydigit = cv2.imread('./image/unknown_aaa.png')

cv2.imshow('detected digit', mydigit)
mydigit_gray = cv2.cvtColor(mydigit, cv2.COLOR_BGR2GRAY)
mydigit_test = mydigit_gray.reshape (-1,400).astype(np. float32)
ret, result, neighbours, dist = knn. findNearest (mydigit_test, k=5)
print(ret)

font = cv2.FONT_HERSHEY_SIMPLEX
cv2.putText(mydigit, str(int(ret)), (0, 7), font, 0.3, (255, 0, 0), 1, cv2.LINE_AA)

cv2.waitKey (0)
cv2.destroyAllWindows ()

```

```
3.0
```

Quiz_403 – กิจกรรมที่ 3/6 – Sudoku to Text by Tesseract

Source code:

```
import cv2
import numpy as np
from PIL import Image
import pytesseract
pytesseract.pytesseract.tesseract_cmd = r'C:\\Program Files\\Tesseract-OCR\\tesseract.exe'

# Step1 - Open
#imageC = cv2.imread("./image/Sodoku_01.jpg")
#imageC = cv2.imread("./image/Sodoku_02.jpg")
#imageC = cv2.imread("./image/Sodoku_03.jpg")
imageC = cv2.imread("./image/Sodoku_04.jpg")


#Step2 - Week07.Lab02.Hough Lines
imageG = cv2.cvtColor(imageC, cv2.COLOR_BGR2GRAY)
imageR = imageC.copy()
edges = cv2.Canny (imageG,50,150)
lines = cv2.HoughLinesP(edges, 1, np.pi/180, 100,minLineLength=100,maxLineGap=10)
min_X,min_Y,max_X,max_Y = 9999,9999,-9999,-9999
for line in lines:
    x1,y1,x2,y2 = line[0]
    min_X = min(min_X,x1,x2)
    min_Y = min(min_Y,y1,y2)
    max_X = max(max_X,x1,x2)
    max_Y = max(max_Y,y1,y2)
    cv2.line(imageR, (x1,y1), (x2,y2), (0,255,0),2)

cv2.line(imageR, (min_X,min_Y), (min_X,max_Y), (0, 0,255),3)
cv2.line(imageR, (min_X,min_Y), (max_X,min_Y), (0, 0,255),3)
cv2.line(imageR, (max_X,max_Y), (max_X,min_Y), (0, 0,255),3)
cv2.line(imageR, (max_X,max_Y), (min_X,max_Y), (0, 0,255),3)
cv2.imshow('Result image',imageR)
cv2.waitKey(250)

#Step3 - Cut Image
edgeCutPercent = 10 / 100
x0, y0 = min_X, min_Y
xStep = (max_X - min_X) / 9.0
yStep = (max_Y - min_Y) / 9.0

for iRow in range(9):
    for jCol in range(9):
        xStart = x0 + int((jCol)  *xStep + (edgeCutPercent*xStep))
        xStop  = x0 + int((jCol+1)*xStep - (edgeCutPercent*xStep))
        yStart = y0 + int((iRow)  *xStep + (edgeCutPercent*yStep))
        yStop  = y0 + int((iRow+1)*xStep - (edgeCutPercent*yStep))

        ROI = imageC[yStart:yStop, xStart:xStop]
        ret,imageB = cv2.threshold(ROI,127,255,cv2.THRESH_BINARY)
        imageX = cv2.cvtColor(imageB, cv2.COLOR_BGR2RGB)
        #cv2.imshow('R-'+str(iRow)+str(jCol),imageX)
        imageP = Image.fromarray(imageX)
        text_from_image = pytesseract.image_to_string(imageP, lang='eng', config='--psm 10 --oem 3 -c tessedit_char_whitelist=0123456789')
```

```
        if len(text_from_image)==1:
            print("- ", end='')
        else:
            print(text_from_image[0] + " ", end='')

    print()

#cv2.imshow('Test image',imageC)
#cv2.imshow('Result image',imageR)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

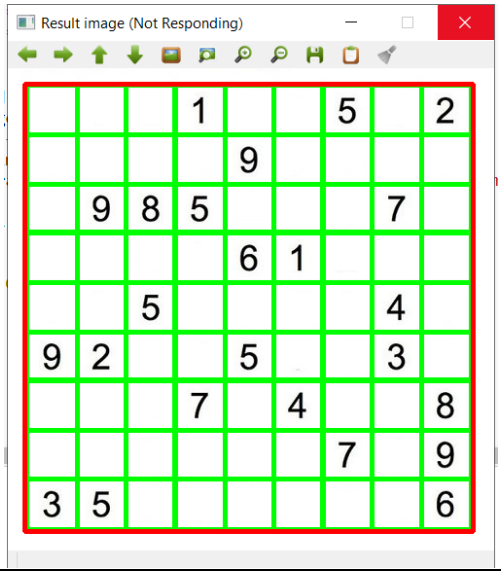
Capture ผลการทำงานที่ได้ลองปฏิบัติ

```
- - 7 - - - - 1 3
- - - 3 - 2 9 - -
- 5 6 - 7 - - - -
- - 1 5 - - - 4 2
4 - - 2 9 - 7 6 -
- - 8 1 - 6 3 - -
- 8 2 7 5 - - - 4
5 - 3 - - 9 2 - -
1 - - - - - - 3 6
```

		7					1	3
			3		2	9		
	5	6		7				
		1	5				4	2
4			2	9		7	6	
		8	1		6	3		
	8	2	7	5				4
5		3			9	2		
1							3	6

ลองใช้ตารางซูโดกุอื่น ในการทดสอบ

```
- - - 1 - - 5 - 2
- - - - - - - - -
- 9 8 5 - - - 7 -
- - - - 6 1 - - -
- - 5 - - - - 4 -
9 2 - - 5 - - 3 -
- - - 7 - 4 - - 8
- - - - - - 7 - -
3 5 - - - - - - 6
```



			1			5		2
				9				
	9	8	5				7	
				6	1			
		5					4	
9	2			5			3	
			7		4			8
						7		9
3	5							6

อภิปรายผล

หลังจากที่เราได้ลองใช้ source code ตัวโค้ดจะทำการตีตารางของ sudoku ออกมาก่อน(ในโค้ดตัวอย่างหากลองรันด้วยรูป Sodoku_02x ซึ่งไม่มีเส้นขอบ จะไม่สามารถรันต่อได้) แล้วจากนั้นจะ capture ช่อแต่ละช่องเพื่อเอารูปไปให้ tesseract อ่านและแปลงออกมาเป็น text กล่าวคือ tesseract ทำหน้าที่เป็น model ในการคำนวณว่ารูปนั้นเป้นตัวเลขอะไรแล้วแปลงออกมาเป็น text

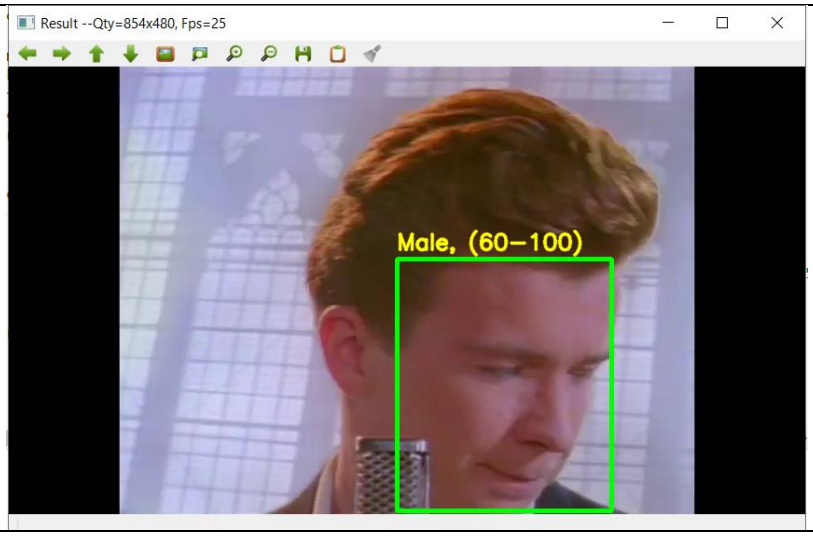
บอกแนวการใช้งาน กับงานที่รับผิดชอบ

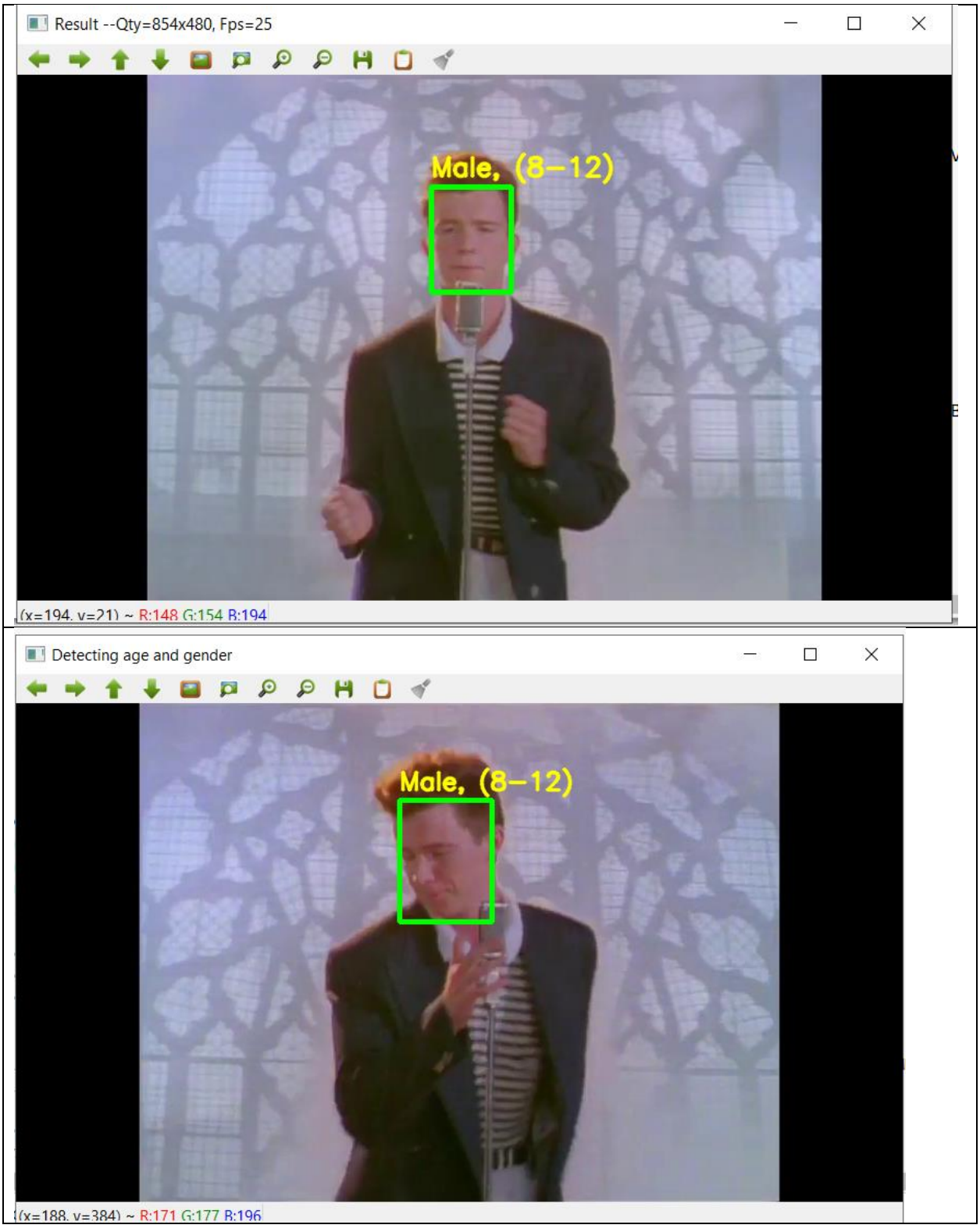
นอกจาก Sudoku แล้ว Tesseract จะช่วยในเรื่องการอ่านข้อมูลเลขที่มีรูปภาพให้ชัดเจน เช่น การอ่านพัสดุ การแปลงข้อความจากรูปถ่ายป้ายต่างๆ เป็นต้น

Quiz_404 – กิจกรรมที่ 4/6 – Gender and Age Detection

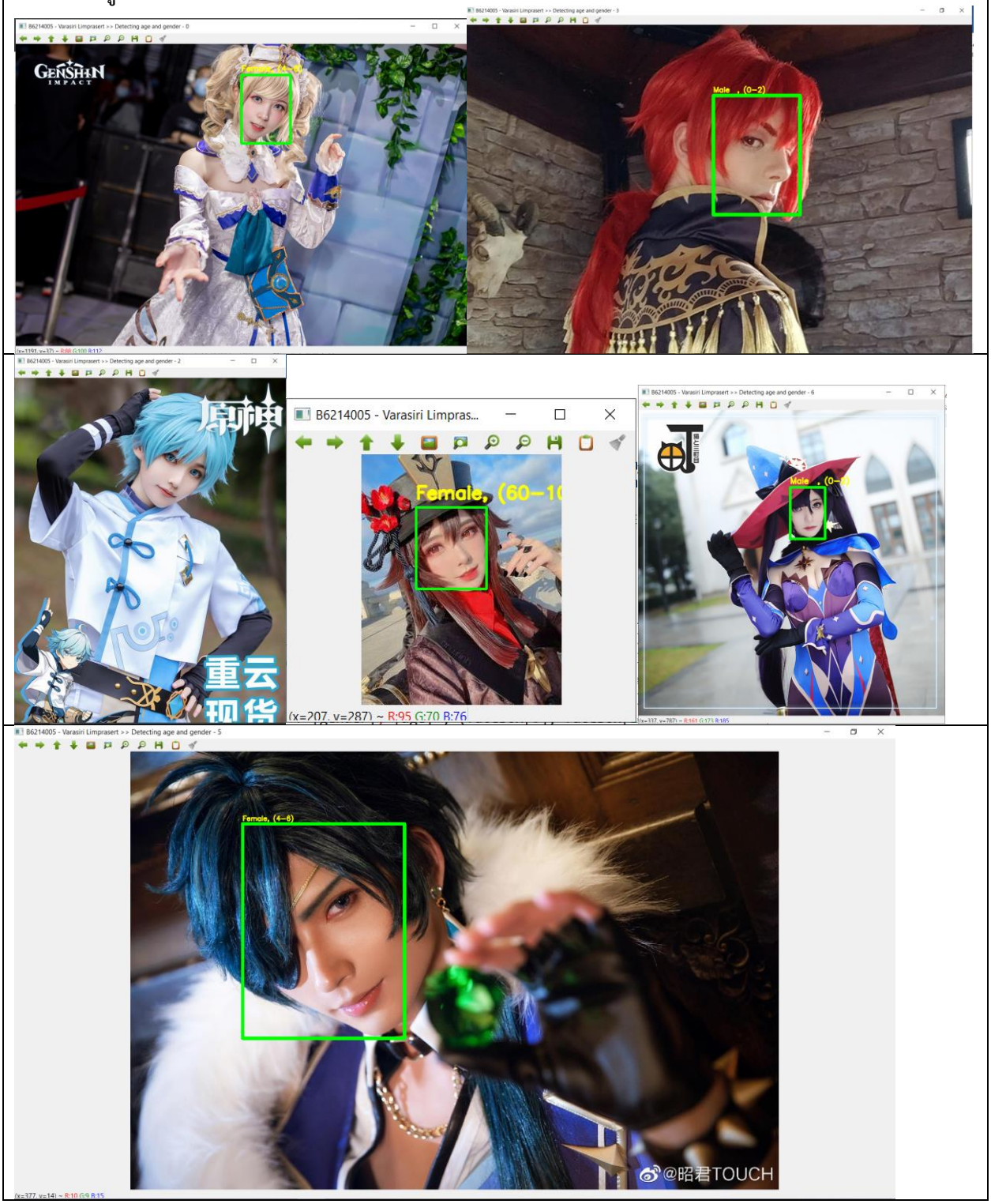
```
import cv2
import math
import argparse

fileName = './video/nevergonnagiveyouup.mp4'

def highlightFace(net, frame, conf_threshold=0.7):
    frameOpencvDnn=frame.copy()
    frameHeight=frameOpencvDnn.shape[0]
    frameWidth=frameOpencvDnn.shape[1]
    blob=cv2.dnn.blobFromImage(frameOpencvDnn, 1.0, (300, 300), [104, 117, 123], True, False)

    net.setInput(blob)
    detections=net.forward()
    faceBoxes=[]
    for i in range(detections.shape[2]):
        confidence=detections[0,0,i,2]
        if confidence>conf_threshold:
            x1=int(detections[0,0,i,3]*frameWidth)
            y1=int(detections[0,0,i,4]*frameHeight)
            x2=int(detections[0,0,i,5]*frameWidth)
            y2=int(detections[0,0,i,6]*frameHeight)
            faceBoxes.append([x1,y1,x2,y2])
            cv2.rectangle(frameOpencvDnn, (x1,y1), (x2,y2), (0,255,0), int(round(frameHeight/150)), 8)
    return frameOpencvDnn,faceBoxes

faceProto   = "./data/opencv_face_detector.pbtxt"
faceModel   = "./data/opencv_face_detector_uint8.pb"
ageProto    = "./data/age_deploy.prototxt"
ageModel    = "./data/age_net.caffemodel"
genderProto = "./data/gender_deploy.prototxt"
genderModel = "./data/gender_net.caffemodel"

MODEL_MEAN_VALUES = (78.4263377603, 87.7689143744, 114.895847746)
ageList    = ['(0-2)', '(4-6)', '(8-12)', '(15-20)', '(25-32)', '(38-43)', '(48-53)', '(60-100)']
genderList = ['Male','Female']

faceNet   = cv2.dnn.readNet(faceModel,faceProto)
ageNet    = cv2.dnn.readNet(ageModel,ageProto)
genderNet = cv2.dnn.readNet(genderModel,genderProto)
```

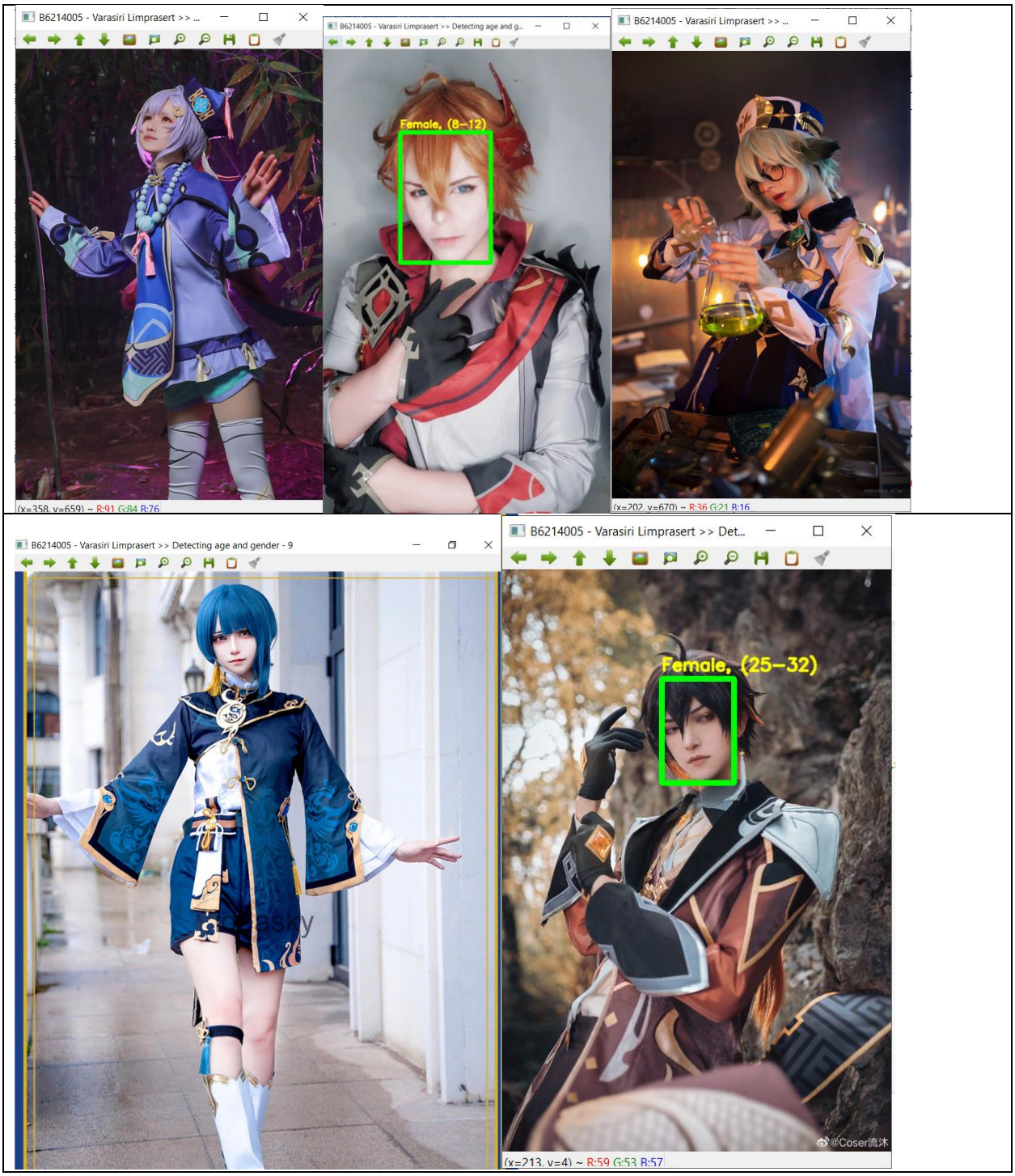
```
cap = cv2.VideoCapture(fileName)
if (cap.isOpened()== False):
    print("Error opening video stream or file")

padding=20
while(cap.isOpened()):
    hasFrame, frame = cap.read()
    if not hasFrame:
        cv2.waitKey()
        break

    if cv2.waitKey(25) & 0xFF == ord('q'):
        break

    resultImg,faceBoxes=highlightFace(faceNet,frame)
    if not faceBoxes:
        print("No face detected")

    for faceBox in faceBoxes:
        face=frame[max(0,faceBox[1]-padding):min(faceBox[3]+padding, frame.shape[0]-1),
                   max(0,faceBox[0]-padding):min(faceBox[2]+padding, frame.shape[1]-1)]

        blob=cv2.dnn.blobFromImage(face, 1.0, (227,227), MODEL_MEAN_VALUES, swapRB=False)
        genderNet.setInput(blob)
        genderPreds=genderNet.forward()
        gender=genderList[genderPreds[0].argmax()]
        print(f'Gender: {gender}')

        ageNet.setInput(blob)
        agePreds=ageNet.forward()
        age=ageList[agePreds[0].argmax()]
        print(f'Age: {age[1:-1]} years')

        cv2.putText(resultImg, f'{gender}, {age}', (faceBox[0], faceBox[1]-10), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0,255,255), 2, cv2.LINE_AA)
        cv2.imshow("Detecting age and gender", resultImg)

cap.release()
cv2.destroyAllWindows()
```




Result --Qty=854x480, Fps=25
Male, (8–12)
(x=194, y=21) ~ R:148 G:154 B:194
Detecting age and gender
Male, (8–12)
(x=188, y=384) ~ R:171 G:177 B:196

ทดสอบการรูปภาพคนคอสเพลย์




B6214005 - Varasiri Limprasert >> ...
B6214005 - Varasiri Limprasert >> Detecting age and g...
Female, (8–12)
B6214005 - Varasiri Limprasert >> ...
B6214005 - Varasiri Limprasert >> Detecting age and gender - 9
B6214005 - Varasiri Limprasert >> Det...
Female, (25–32)
@Coser流沐

อภิปรายผล ประเด็นความถูกต้องของการระบุเพศ ของ DNN(Deep Neural Network) โมเดลนี้

จากการจับภาพวิดีโอ ตัวโมเดลทายเพศได้ถูกต้อง แต่เมื่อลองใช้กับรูปภาพนักคอสเพลย์กลับทายผิดพลาดค่อนข้างมาก ซึ่งอาจเกิดจาก

- ภาพนั้นต้องเห็นองค์ประกอบใบหน้าครบ (ต้องมีตาครบ 2 ข้าง)
- หน้าต้องไม่แต่งมากจนเกินไป (สังเกตจากรูปตัวอย่างที่ใช้คนคอสเพลย์มาทาย)
- มุมใบหน้าต้องไม่เอียงจนเกินไป
- แสงต้องมากพอที่จะเห็นใบหน้าชัดเจน
- ดวงตาน่าจะต้องเป็นสีตามธรรมชาติ (กรณีคอสเพลย์อาจจะทำให้ตรวจจับไม่ได้)

อภิปรายผล ประเด็นความถูกต้องของการระบุช่วงอายุ ของ DNN โมเดลนี้

ช่วงอายุทั้งแบบวิดีโอและทายภาพนักคอสเพลย์ โมเดลต่างก็ทายผิดพลาดค่อนข้างมาก คิดว่าอาจมาจากโมเดลที่ยังไม่แม่นยำมากนัก หรือการจัดวางมุมกล้องทำให้ใบหน้าดูไม่สมบูรณ์จนพยากรได้ผิดพลาด

คำถามที่อยากถาม

-

บอกแนวการใช้งาน กับงานที่รับผิดชอบ

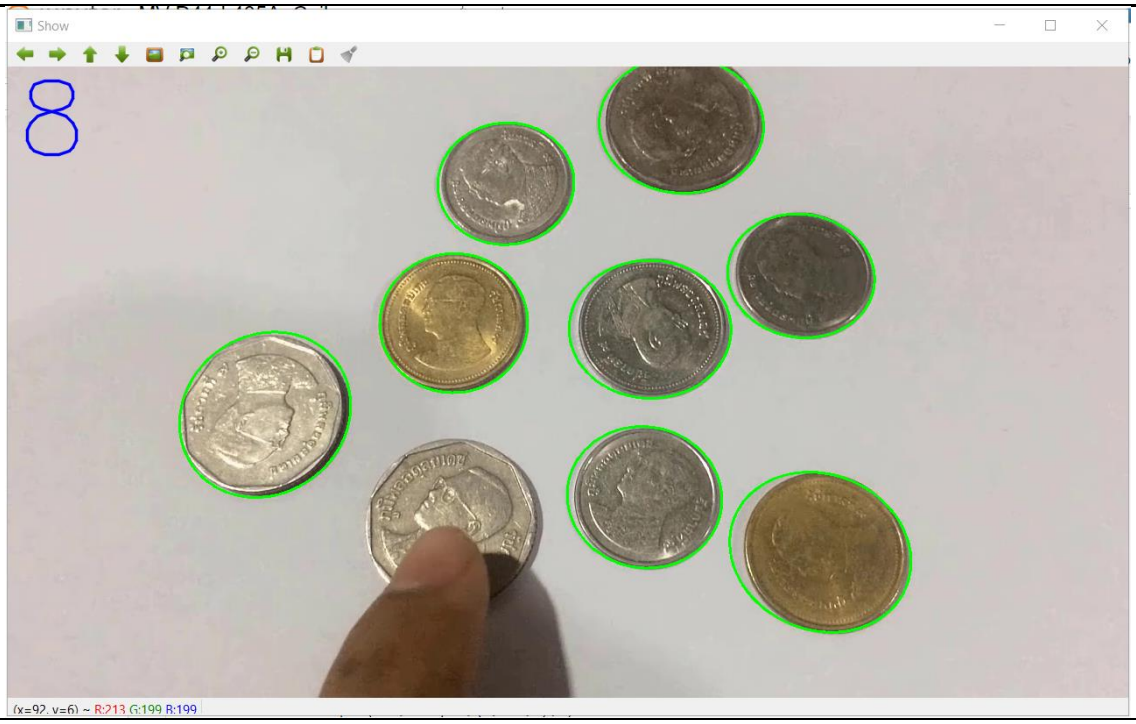
เพียงแค่เราจัดเตรียมรูปที่มีใบหน้าคนชัดเจน ไม่แต่งหน้า หรือแสงมากจนเกินไป โมเดลนี้ก็จะพยากรได้แม่นยำมากขึ้น

Quiz_405 – กิจกรรมที่ 5/6 – Object Detection and Tracking

Capture ผลการทำงานที่ได้ลองปฏิบัติทั้ง 4 กรณี A, B, C, D

กรณี A

```
import cv2
import numpy as np
cap=cv2.VideoCapture("./image/Coin2.mp4")

while(True):
    ref,frame = cap.read()
    if frame is None:
        break
    else:
        roi = frame[:1080,0:1920]

        gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
        gray_blur = cv2.GaussianBlur(gray,(15,15),0)
        thresh = cv2.adaptiveThreshold(gray_blur,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BINARY_INV,11,1)
        kernel = np.ones((3,3),np.uint8)
        closing = cv2.morphologyEx(thresh,cv2.MORPH_CLOSE,kernel,iterations=4)

        result_img = closing.copy()
        contours,hierachy = cv2.findContours(result_img,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
        counter = 0
        for cnt in contours:
            area = cv2.contourArea(cnt)
            if area<5000 or area > 35000:
                continue
            ellipse = cv2.fitEllipse(cnt)
            cv2.ellipse(roi,ellipse,(0,255,0),2)
            counter += 1
        cv2.putText(roi,str(counter),(10,100),cv2.FONT_HERSHEY_SIMPLEX,4,(255,0,0),2,cv2.LINE_AA)
        cv2.imshow("Show",roi)

    if cv2.waitKey(1) & 0xFF==ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```



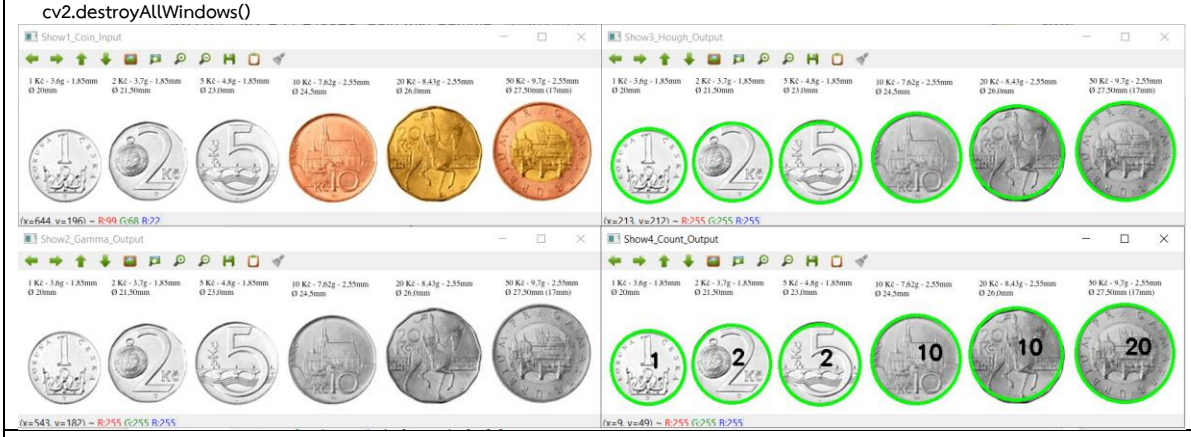
กรณี B

```
# Coin recognition, real life application
# task: calculate the value of coins on picture

import cv2
import numpy as np

#=============================================================================================
def linear_stretching(input, lower_stretch_from, upper_stretch_from):
    """
    Linear stretching of input pixels
    :param input: integer, the input value of pixel that needs to be stretched
    :param lower_stretch_from: lower value of stretch from range - input
    :param upper_stretch_from: upper value of stretch from range - input
    :return: integer, integer, the final stretched value
    """
    lower_stretch_to = 0  # lower value of the range to stretch to - output
    upper_stretch_to = 255  # upper value of the range to stretch to - output
    output = (input - lower_stretch_from) * ((upper_stretch_to - lower_stretch_to)
        / (upper_stretch_from - lower_stretch_from)) + lower_stretch_to
    return output

#=============================================================================================
def gamma_correction():
    """
    Restore the contrast in the faded image using linear stretching.
    """
    # imports the image of the moon
    #moon = cv2.imread('./image/moon.jpg', 0)
    print('On-Run: Gamma_correction')
    moon = cv2.imread('./image/Show1_Gamma_Input.jpg', 0)
    # assign variable to max and min value of image pixels
```

```
    max_value = np.max(moon)
    min_value = np.min(moon)
    # cycle to apply linear stretching formula on each pixel
    print('On-Run: Linear_stretching')
    for y in range(len(moon)):
        for x in range(len(moon[y])):
            moon[y][x] = linear_stretching(moon[y][x], min_value, max_value)
    # writes out the resulting restored picture
    cv2.imwrite("./image/Show2_Gamma_Output.jpg", moon)
    print('Finish: Linear_stretching')
    print('Finish: Gamma_correction')

#================================================================================
def detect_coins():
    print('On-Run: Detect_coins')
    coins = cv2.imread('./image/Show2_Gamma_Output.jpg', 1)
    gray = cv2.cvtColor(coins, cv2.COLOR_BGR2GRAY)
    img = cv2.medianBlur(gray, 7)
    circles = cv2.HoughCircles(
        img,  # source image
        cv2.HOUGH_GRADIENT,  # type of detection
        1,
        50,
        param1 = 100,
        param2 = 50,
        minRadius = 10,  # minimal radius
        maxRadius = 80,  # max radius
    )

    coins_copy = coins.copy()
    for detected_circle in circles[0]:
        x_coor, y_coor, detected_radius = detected_circle
        coins_detected = cv2.circle(
            coins_copy,
            (int(x_coor), int(y_coor)),
            int(detected_radius),
            (0, 255, 0),
            4,
        )

    cv2.imwrite("./image/Show3_Hough_Output.jpg", coins_detected)
    print('Finish: Detect_coins')
    return circles

#================================================================================
def calculate_amount():
    koruny = {
        "1 CZK": {
            "value": 1,
            "radius": 20,
            "ratio": 1,
            "count": 0,
        },
        "2 CZK": {
            "value": 2,
            "radius": 21.5,
            "ratio": 1.075,
```

```
        "count": 0,
    },
    "5 CZK": {
        "value": 5,
        "radius": 23,
        "ratio": 1.15,
        "count": 0,
    },
    "10 CZK": {
        "value": 10,
        "radius": 24.5,
        "ratio": 1.225,
        "count": 0,
    },
    "20 CZK": {
        "value": 20,
        "radius": 26,
        "ratio": 1.3,
        "count": 0,
    },
    "50 CZK": {
        "value": 50,
        "radius": 27.5,
        "ratio": 1.375,
        "count": 0,
    },
}

print('On-Run: Calculate_amount')
circles = detect_coins()
radius = []
coordinates = []

for detected_circle in circles[0]:
    x_coor, y_coor, detected_radius = detected_circle
    radius.append(detected_radius)
    coordinates.append([x_coor, y_coor])

smallest = min(radius)
tolerance = 0.0375
total_amount = 0

coins_circled = cv2.imread("./image/Show3_Hough_Output.jpg", 1)
font = cv2.FONT_HERSHEY_SIMPLEX

for coin in circles[0]:
    ratio_to_check = coin[2] / smallest
    coor_x = coin[0]
    coor_y = coin[1]
    for koruna in koruny:
        value = koruny[koruna]['value']
        if abs(ratio_to_check - koruny[koruna]['ratio']) <= tolerance:
            koruny[koruna]['count'] += 1
            total_amount += koruny[koruna]['value']
            cv2.putText(coins_circled, str(value), (int(coor_x),
                int(coor_y)), font, 1, (0, 0, 0), 4)
```

```
    print(f"The total amount is {total_amount} CZK")
    for koruna in koruny:
        pieces = koruny[koruna]['count']
        print(f"{koruna} = {pieces}x")

    cv2.imwrite("./image/Show4_Count_Output.jpg", coins_circled)
    print('Finish: Calculate_amount')

#================================================================================================
if __name__ == "__main__":
    picTest = cv2.imread("./image/koruny_r11.jpg") # Picture-1
    # picTest = cv2.imread("./image/koruny_t10.jpg") # Picture-2
    # picTest = cv2.imread("./image/koruny_t20.jpg") # Picture-3
    # picTest = cv2.imread("./image/koruny_2.jpg")   # Picture-4

    cv2.imwrite("./image/Show1_Gamma_Input.jpg", picTest)
    gamma_correction()
    calculate_amount()
    coins1 = cv2.imread("./image/Show1_Gamma_Input.jpg")
    coins2 = cv2.imread("./image/Show2_Gamma_Output.jpg")
    coins3 = cv2.imread("./image/Show3_Hough_Output.jpg")
    coins4 = cv2.imread("./image/Show4_Count_Output.jpg")
    cv2.imshow("Show1_Coin_Input", coins1)
    cv2.imshow("Show2_Gamma_Output", coins2)
    cv2.imshow("Show3_Hough_Output", coins3)
    cv2.imshow("Show4_Count_Output", coins4)
    cv2.waitKey()
    cv2.destroyAllWindows()
```

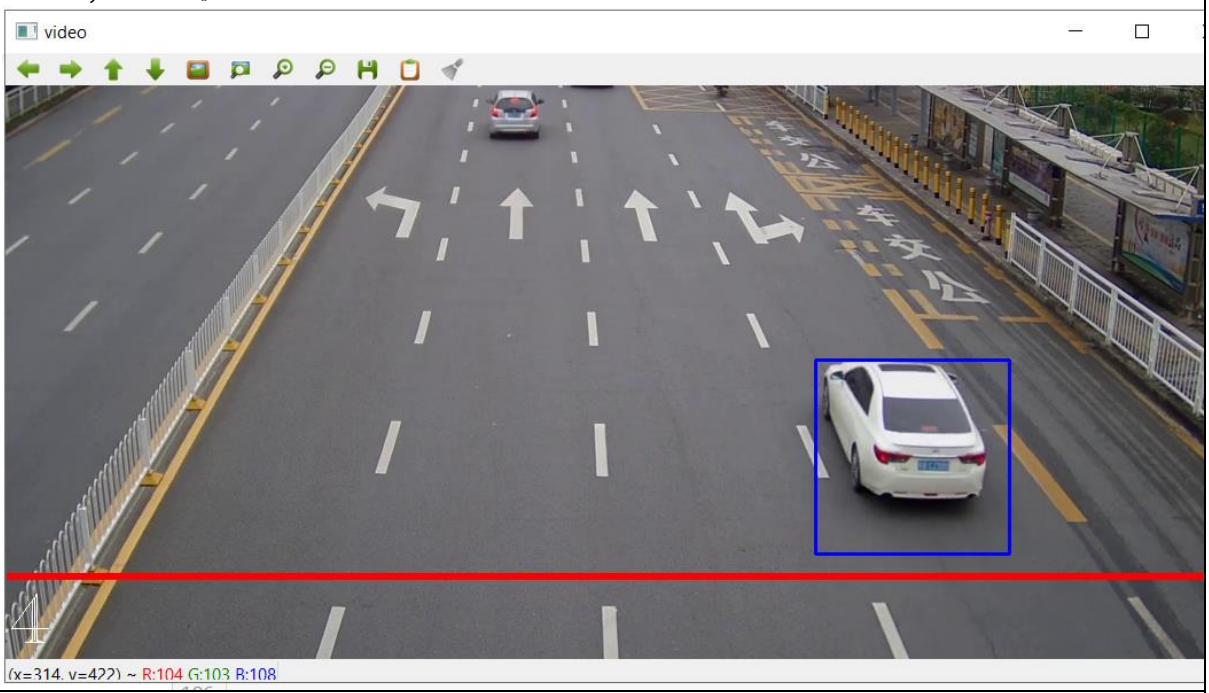


กรณี C

```
import cv2
import numpy as np

class Kordinat:
    def __init__(self,x,y):
        self.x=x
        self.y=y

class Sensor:
    def __init__(self,kordinat1,kordinat2,frame_weight,frame_lenght):
        self.kordinat1=kordinat1
        self.kordinat2=kordinat2
        self.frame_weight=frame_weight
        self.frame_lenght =frame_lenght
```

```
        self.mask=np.zeros((frame_weight,frame_lenght,1),np.uint8)*abs(
            self.kordinat2.y-self.kordinat1.y)
        self.full_mask_area=abs(self.kordinat2.x-self.kordinat1.x)
        cv2.rectangle(self.mask,(self.kordinat1.x,self.kordinat1.y),
                    (self.kordinat2.x,self.kordinat2.y),(255),thickness=cv2.FILLED)
        self.stuation=False
        self.car_number_detected=0

Sensor1 = Sensor(Kordinat(1, 425), Kordinat(1080, 430), 500, 1080)
video=cv2.VideoCapture("./image/CarVideo_02.mp4")
fgbg=cv2.createBackgroundSubtractorMOG2()
#fgbg=cv2.createBackgroundSubtractorMOG2()
kernel=np.ones((5,5),np.uint8)
font=cv2.FONT_HERSHEY_TRIPLEX
while (1):
    ret,frame=video.read()
    # resize frame
    cut_image=frame[100:600,100:1180]
    # make morphology for frame
    deleted_background=fgbg.apply(cut_image)
    opening_image=cv2.morphologyEx(deleted_background,cv2.MORPH_OPEN,kernel)
    ret,opening_image=cv2.threshold(opening_image,125,255,cv2.THRESH_BINARY)

    # detect moving anything
    contours, hierarchy  =cv2.findContours(opening_image,
                                    cv2.RETR_TREE,cv2.CHAIN_APPROX_NONE)
    result=cut_image.copy()

    zeros_image=np.zeros((cut_image.shape[0],cut_image.shape[1],1),np.uint8)

    # detect moving anything with loop
    for cnt in contours:
        x,y,w,h=cv2.boundingRect(cnt)
        if (w>100 and h>100 ):
            cv2.rectangle(result,(x,y),(x+w,y+h),(255,0,0),thickness=2)
            cv2.rectangle(zeros_image,(x,y),(x+w,y+h),(255),thickness=cv2.FILLED)

    # detect whether there is car via bitwise_and
    mask1=np.zeros((zeros_image.shape[0],zeros_image.shape[1],1),np.uint8)
    mask_result=cv2.bitwise_or(zeros_image,zeros_image,mask=Sensor1.mask)
    white_cell_number=np.sum(mask_result==255)

    # detect to control whether car is passing under the red line sensor
    sensor_rate=white_cell_number/Sensor1.full_mask_area
    if sensor_rate>0:
        print(sensor_rate)

    # if car is passing under the red line sensor . red line sensor is yellow color.
    if (sensor_rate >= 1.8 and sensor_rate<2.9 and Sensor1.stuation == False):
        # draw the red line
        cv2.rectangle(result, (Sensor1.kordinat1.x, Sensor1.kordinat1.y),
                    (Sensor1.kordinat2.x, Sensor1.kordinat2.y),
                    (0, 0, 255), thickness=cv2.FILLED)
        Sensor1.stuation = False
        Sensor1.car_number_detected += 2
    if (sensor_rate>=0.6 and  sensor_rate<1.8 and Sensor1.stuation==False):
        # draw the red line
```

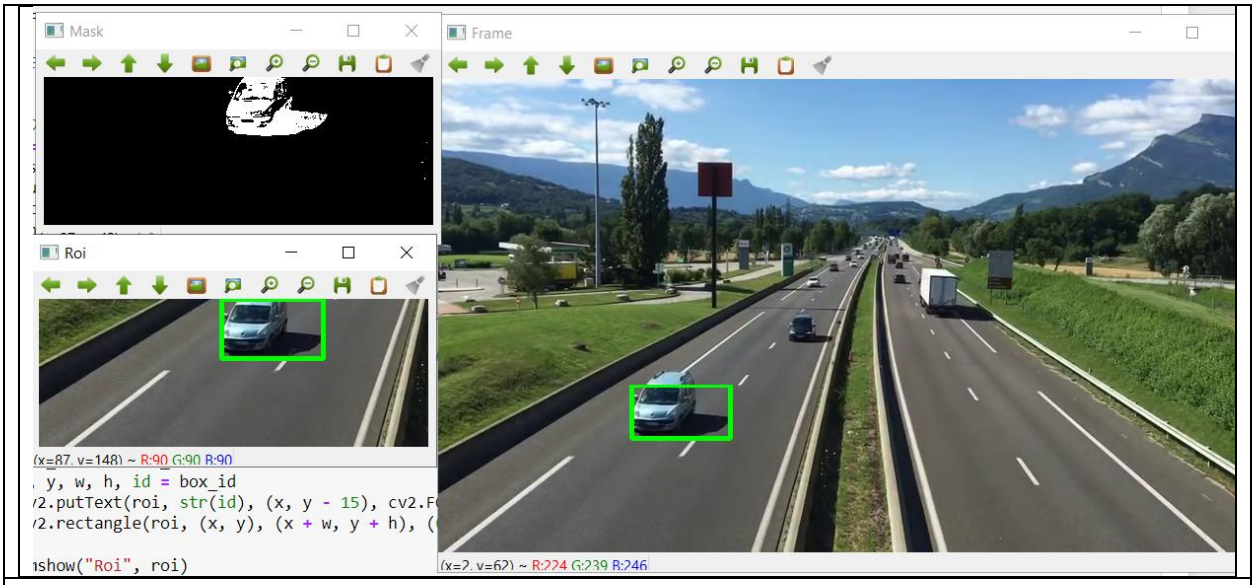
```
        cv2.rectangle(result, (Sensor1.kordinat1.x, Sensor1.kordinat1.y),
                      (Sensor1.kordinat2.x, Sensor1.kordinat2.y),
                      (0,255, 0,), thickness=cv2.FILLED)
        Sensor1.stuation = True
    elif (sensor_rate<0.6 and Sensor1.stuation==True) :
        # draw the red line
        cv2.rectangle(result, (Sensor1.kordinat1.x, Sensor1.kordinat1.y),
                      (Sensor1.kordinat2.x, Sensor1.kordinat2.y),
                      (0, 0,255), thickness=cv2.FILLED)
        Sensor1.stuation = False
        Sensor1.car_number_detected+=1
    else :
        # draw the red line
        cv2.rectangle(result, (Sensor1.kordinat1.x, Sensor1.kordinat1.y),
                      (Sensor1.kordinat2.x, Sensor1.kordinat2.y),
                      (0, 0, 255), thickness=cv2.FILLED)

    cv2.putText(result,str(Sensor1.car_number_detected),
                (Sensor1.kordinat1.x,Sensor1.kordinat1.y+60),font,2,(255,255,255))

    cv2.imshow("video", result)
    #cv2.imshow("mask_result", mask_result)
    #cv2.imshow("zeros_image", zeros_image)
    #cv2.imshow("opening_image", opening_image)

    k=cv2.waitKey(30) & 0xff
    if k == 27 :  # ESC Key
        break

video.release()
cv2.destroyAllWindows()
```



กรณี D

```
import cv2
```

```
from moving_tracker import *

tracker = EuclideanDistTracker()
cap = cv2.VideoCapture(".\image\CarVideo_01.mp4")
object_detector = cv2.createBackgroundSubtractorMOG2(history=100, varThreshold=40)

while True:
    ret, frame = cap.read()

    # 0. Extract Region of interest
    roi = frame[310: 460, 5: 400]

    # 1. Object Detection
    mask = object_detector.apply(roi)
    _, mask = cv2.threshold(mask, 254, 255, cv2.THRESH_BINARY)
    contours, _ = cv2.findContours(mask, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
    detections = []
    for cnt in contours:
        # Calculate area and remove small elements
        area = cv2.contourArea(cnt)
        if area > 900:
            #cv2.drawContours(roi, [cnt], -1, (0, 255, 0), 2)
            x, y, w, h = cv2.boundingRect(cnt)
            #cv2.rectangle(roi, (x, y), (x + w, y + h), (0, 255, 0), 3)
            detections.append([x, y, w, h])

    # 2. Object Tracking
    boxes_ids = tracker.update(detections)
    for box_id in boxes_ids:
        x, y, w, h, id = box_id
        cv2.putText(roi, str(id), (x, y - 15), cv2.FONT_HERSHEY_PLAIN, 2, (255, 0, 0), 2)
        cv2.rectangle(roi, (x, y), (x + w, y + h), (0, 255, 0), 3)

    cv2.imshow("Roi", roi)
    cv2.imshow("Mask", mask)
    cv2.imshow("Frame", frame)
    k=cv2.waitKey(30)
    if k == 27 :  # ESC Key
        break
cap.release()
cv2.destroyAllWindows()
```



อภิปรายผลการทดสอบ แต่ละหัวข้อ (A, B, C, D)

A -> เป็นการตีกรอบวงกลมให้กับรูป โดยถ้าเหรียญนั้นตีวงกลมได้พอดีก็จะถูกวาดรูปวงกลมลงไปแล้วนับ 1

B -> เพิ่มเติมจากกรณี A ตรงที่มีการเพิ่มเงื่อนไขในการตรวจจับว่า เหรียญขนาดเท่าไหร่ มีค่าเท่ากับเท่าไหน แล้วใส่ Text กำกับมูลค่าของเหรียญไว้บนภาพ

C -> เป็นการตรวจจับวัตถุที่ผ่านเส้นตรวจจับ(ในผลลัพธ์คือเส้นสีแดง) ขั้นแรกจะทำการหา Mark ของมุมกล้องนั้นก่อน แล้วจากนั้นจึงเอา Mask มาหาวัตถุที่อยู่นอกเหนือจาก Mark นั้นๆ ซึ่งก็คือรถ หรือต้นไม้ที่เคลื่อนไหวไปมา เมื่อวัตถุที่ไม่ตรงกับ Mark วิ่งผ่านเส้นสีแดงก็จะถูกนับเป็น 1

D -> กรณีนี้คล้ายกับกรณี C เพียงแต่เส้นที่ใช้ตรวจจับวัตถุถูกเปลี่ยนเป็นพื้นที่เฉพาะส่วนในภาพทั้งหมด โดย Roi จะทำหน้าที่ตรวจจับทุกวัตถุที่เข้ามาใน frame แล้วไม่ตรงกับ Mark จะนับเป็น 1 คัน

คำถามที่อยากถาม

-

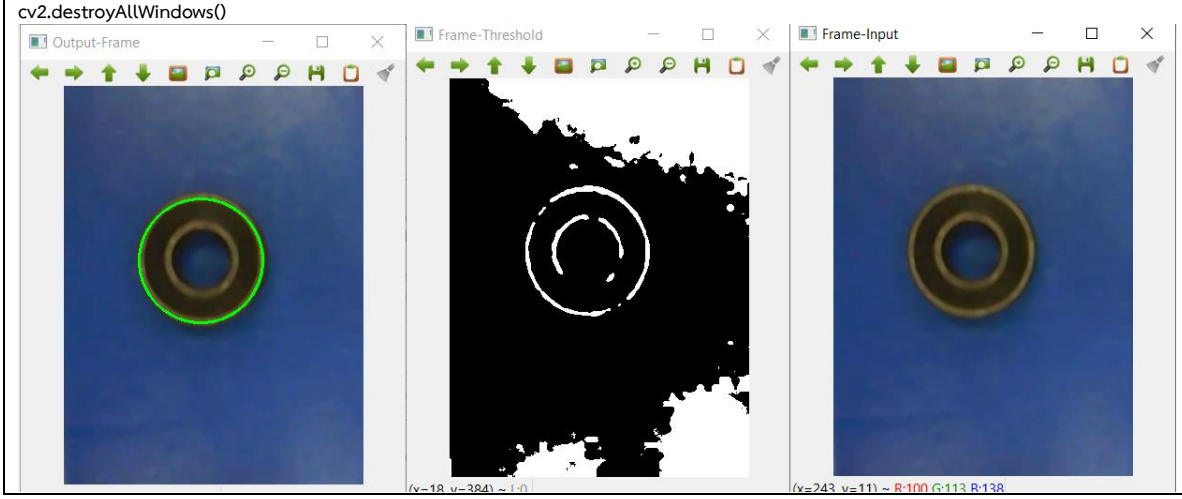
บอกแนวการใช้งาน กับงานที่รับผิดชอบ

สามารถนำไปใช้กับการตรวจจับวัตถุที่ผ่านเข้าออกพื้นที่นั้นๆ เช่น การตรวจจำนวนคนเข้าร้าน หรือ การนับวัตถุบนสายพานที่มีความเรียบเสมอกัน

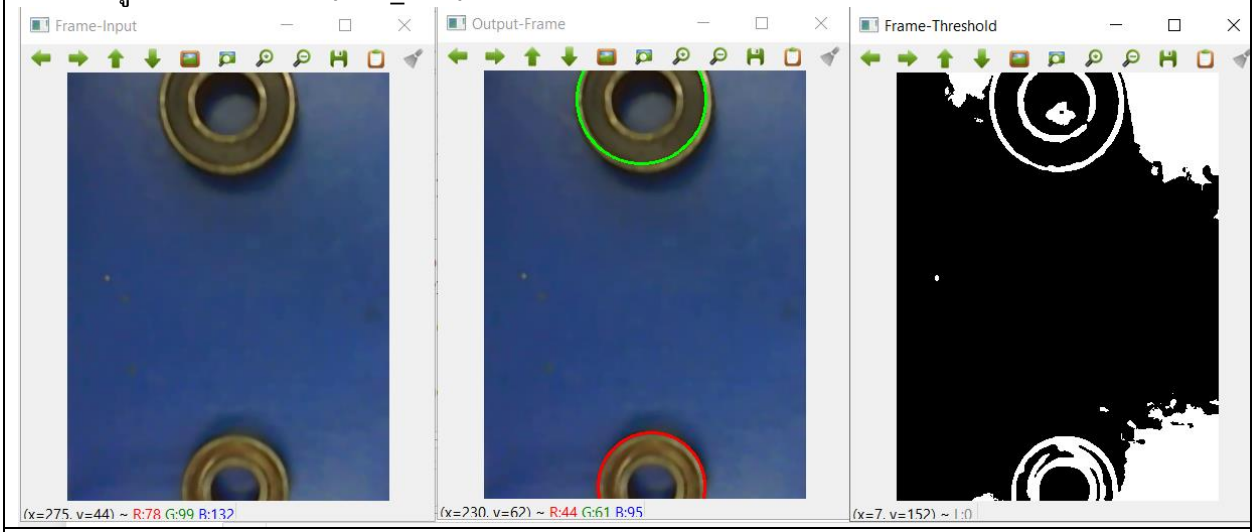
Quiz_406 – กิจกรรมที่ 6/6 – Visual Inspection

Capture ผลการทำงานที่ได้ลองปฏิบัติ

```
import cv2
import time
import numpy as np

cap = cv2.VideoCapture('./image/save2.avi')
while(True):
    ret, frame = cap.read()

    # Cut the Area, Blur, Chage to Gray
    belt = frame[50: 475, 80: 400]
    belt = cv2.medianBlur(belt,5)
    gray_belt = cv2.cvtColor(belt, cv2.COLOR_BGR2GRAY)
    cv2.imshow('Frame-Input',belt)

    # Threshold Operation
    _, threshold = cv2.threshold(gray_belt,80,255,cv2.THRESH_BINARY)
    cv2.imshow('Frame-Threshold',threshold)

    # Mask Circle and Draw
    circles = cv2.HoughCircles(threshold,cv2.HOUGH_GRADIENT,1,20,
                        param1=50,param2=30,minRadius=30,maxRadius=80)
    if circles is not None:
        circles = np.uint16(np.around(circles))
        for i in circles[0,:]:
            #cv2.circle(belt,(i[0],i[1]),i[2],(0,255,0),2)
            if i[2] >= 60:  # big nut
                cv2.circle(belt,(i[0],i[1]),i[2],(0,255,0),2)
            elif 40 < i[2] < 60:
                cv2.circle(belt,(i[0],i[1]),i[2],(0,0,255),2)

    # Show Result and Wait Exit
    cv2.imshow('Output-Frame',belt)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
    time.sleep(0.01)

cap.release()
cv2.destroyAllWindows()
```



ลองใช้ข้อมูลอื่นในการทดสอบ (save_3.avi)



อภิปรายผล

จากการทดสอบ ขั้นแรกเราจะใช้การตรวจสอบว่าพื้นที่ไหนมีความต่างของสีมาก เราจะทำ Mask เอาไว้ แล้วจากนั้น หาก Mask มีลักษณะเป็นวงกลมตามรัศมีที่เรากำหนด ให้วาดวงกลมทับลงบนเฟลม Output โดยวงเล็กถูกวาดด้วยเส้นสีแดง ส่วนสีเขียวจะเป็นวงกลมใหญ่ เพื่อใช้จำแนกขนาดของลูกปืนบนสายพาน

คำถามที่อยากถาม

-

บอกแนวการใช้งาน กับงานที่รับผิดชอบ

นอกจากลูกปืนแล้ว เรายังสามารถใช้โค้ดนี้ในการตรวจหาวัตถุที่มีขนาดต่างกัน เช่น น็อต เหรียญ หรือฝาขวด โดยพื้นหลังของเราจำเป็นจะต้องมีสีที่เสมอกัน และเป็นไปได้ให้จัดแสงให้สว่างเท่ากันทั้งเฟลมจะให้ผลลัพธ์แม่นยำที่สุด (อย่าใช้สายพานที่มีลวดลาย หรือผิวขรุขระเกินไป)