

การพัฒนาโปรแกรมประยุกต์และปัญญาประดิษฐ์ เพื่อการมองเห็นของเครื่องจักร  
Computer Programing and Artificial Intelligence in Machine Vision

ชื่อ-สกุล : นางสาวณัฐชยา ผ่องกุล B6226718

8/8 -- คำถามท้ายบทเพื่อทดสอบความเข้าใจ

### Quiz\_401 – กิจกรรมที่ 1/6

จงสร้างแบบจำลองการถดถอยเชิงเส้นอย่างง่ายสำหรับการพยากรณ์ค่าเช่าต่อเดือน (บาท) จากขนาดของพื้นที่ (ตารางเมตร) โดยมีข้อมูลดังต่อไปนี้ไฟล์ Area\_Rental.csv  
โดยอยากทราบว่าพื้นที่ขนาด 50 ตารางเมตร จะต้องจ่ายค่าเช่าประมาณเดือนละเท่าไร?

ทำตามนี้ได้เลย

```
PS C:\Users\asus\OneDrive\งาน\งาน\Week0600_20220112 Machine Learning> conda install scikit-learn
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##
  environment location: C:\Users\asus\miniconda3
  added / updated specs:
    - scikit-learn

The following packages will be downloaded:

```

package	build	size
certifi-2021.10.8	py39haad5532_2	152 KB
joblib-1.1.0	py39h3eb1b0_0	211 KB
scikit-learn-1.0.2	py39h3eb1b0_0	5.0 MB
threadpoolctl-2.2.0	py39h69192_0	16 KB
Total:		5.4 MB

```

The following NEW packages will be INSTALLED:
  joblib          pkgs/main/noarch::joblib-1.1.0-py39h3eb1b0_0
  scikit-learn    pkgs/main/win-64::scikit-learn-1.0.2-py39h3eb1b0_0
  threadpoolctl  pkgs/main/noarch::threadpoolctl-2.2.0-py39h69192_0

The following packages will be UPDATED:
  certifi         2021.10.8 py39haad5532_0 -> 2021.10.8 py39haad5532_2

Proceed ([y]/n)? y

Downloading and Extracting Packages
threadpoolctl-2.2.0 - 16 KB ##### 100%

```

```
PS C:\Users\asus\OneDrive\งาน\งาน\Week0600_20220112 Machine Learning> conda install pandas
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##
  environment location: C:\Users\asus\miniconda3
  added / updated specs:
    - pandas

The following packages will be downloaded:

```

package	build	size
bottleneck-1.3.2	py39h7cc1a96_1	107 KB
numexpr-2.8.1	py39h69192_0	117 KB
pandas-1.3.5	py39h6214cd6_0	8.6 MB
pytz-2021.3	pyh3eb1b0_0	171 KB
Total:		9.0 MB

```

The following NEW packages will be INSTALLED:
  bottleneck      pkgs/main/win-64::bottleneck-1.3.2-py39h7cc1a96_1
  numexpr         pkgs/main/win-64::numexpr-2.8.1-py39h69192_0
  pandas          pkgs/main/win-64::pandas-1.3.5-py39h6214cd6_0
  pytz            pkgs/main/noarch::pytz-2021.3-pyh3eb1b0_0

Proceed ([y]/n)? y

```

- conda install pandas
- conda install scikit-learn

```
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 %config InlineBackend.figure_format = 'retina'
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

%config InlineBackend.figure_format = 'retina'
```

อ่านข้อมูล CSV ไฟล์

```
1 bp = pd.read_csv('./data/Area_Rental.csv')
2 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
```

```
bp = pd.read_csv('./data/Area_Rental.csv')
print(bp)
```

ดูค่าสหสัมพันธ์(Correlation) ระหว่าง Area กับ Rental

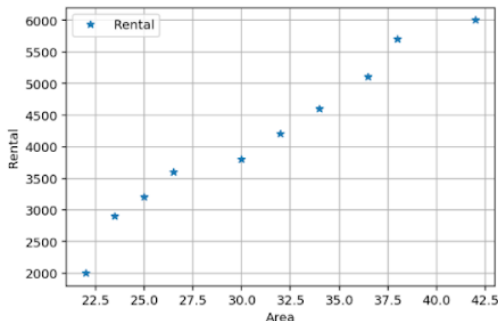
```
In [18]: 1 bp.corr()
```

```
Out[18]:
```

	Area	Rental
Area	1.000000	0.982158
Rental	0.982158	1.000000

สร้างแผนภาพการกระจาย (scatter plot) ระหว่าง Area และ Rental

```
1 bp.plot(x='Area', y='Rental', style='*')
2 plt.xlabel('Area')
3 plt.ylabel('Rental')
4 plt.grid()
5 plt.show()
6
```



```
bp.plot(x='Area', y='Rental', style='*')
plt.xlabel('Area')
plt.ylabel('Rental')
plt.grid()
plt.show()
```

นำเข้า Linear Regression

```
1 from sklearn.linear_model import LinearRegression
```

```
from sklearn.linear_model import LinearRegression
```

กำหนดตัวแปรต้น (X) และตัวแปรตาม (y)

```
In [23]: 1 xx = bp[['Area']]
          2 yy = bp['Rental']
```

```
xx = bp[['Area']]
```

```
yy = bp['Rental']
```

สร้างและฝึกฝนแบบจำลอง

```
In [24]: 1 lrm = LinearRegression()
          2 lrm.fit(xx,yy)
```

```
Out[24]: LinearRegression()
```

```
lrm = LinearRegression()
```

```
lrm.fit(xx,yy)
```

ดูค่าจุดตัดแกน  $y$  (ค่า  $c$ )

```
In [25]: 1 lrm.intercept_
```

```
Out[25]: -1629.5818148125263
```

```
lrm.intercept_
```

ดูค่าสัมประสิทธิ์ (ค่า  $m$ )

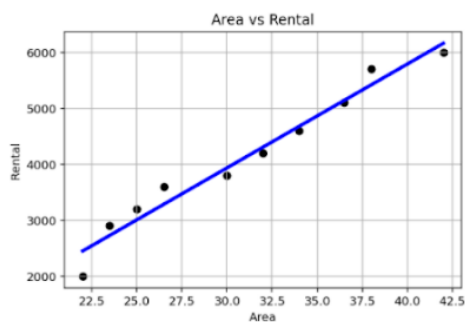
```
In [26]: 1 lrm.coef_
```

```
Out[26]: array([185.44690839])
```

```
lrm.coef
```

การพยากรณ์ด้วยค่า  $X$  และสร้างกราฟผลการพยากรณ์

```
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')
7 plt.grid()
8 plt.show()
9
```



```
predictions = lrm.predict(xx)
plt.scatter(xx, yy, color='black')
plt.plot(xx, predictions, color='blue', linewidth=3)
plt.title('Area vs Rental')
plt.xlabel('Area')
plt.ylabel('Rental')
plt.grid()
plt.show()
```

ทดลองพยากรณ์ ข้อมูลใหม่

จากตัวอย่างการใช้ KNN จงเปลี่ยน dataset เป็นไฟล์จาก digits\_dataset2.zip โดยจะมีข้อมูลตัวเลขเพิ่มขึ้นมาเป็น 0-9 (จำนวนภาพ 500 ภาพต่อ 1 ตัวเลข)

- # ทำตามนี้ได้เลย
- 
- ## Step1/4: จัดเตรียมข้อมูล
- In [4]:

```
1 # ตัวอย่าง ข้อมูลแบบภาพสี  
2 import cv2  
3 import numpy as np  
4 img = cv2.imread('./image/digits_dataset2/0_1.png')  
5 img.shape, img
```
- Out[4]: ((20, 20, 3), array([[0, 0, 0],  
 [0, 0, 0],  
 [0, 0, 0],  
 ...,  
 ... - - ]])
- In [5]:

```
1 # ตัวอย่าง ข้อมูลแบบภาพขาวดำ  
2 import cv2  
3 import numpy as np  
4 img = cv2.imread('./image/digits_dataset2/0_1.png')  
5 gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)  
6 gray.shape, gray
```
- Out[5]: ((20, 20), array([[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0],  
 [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0],  
 [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0])
- # ตัวอย่าง ข้อมูลแบบภาพสี

```

import cv2
import numpy as np
img = cv2.imread('./image/digits_dataset2/0_1.png')
img.shape, img

# ตัวอย่าง ข้อมูลแบบภาพขาวดำ
import cv2
import numpy as np
img = cv2.imread('./image/digits_dataset2/0_1.png')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
gray.shape, gray

```

#### Step2/4: การนำเข้าข้อมูล

```

1 train = np.array([])
2 for i in range(9):
3     for j in range(1, 251):
4         img = cv2.imread('./image/digits_dataset2/' + str(i) + '_' + str(j) + '.png')
5         gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
6         train = np.append(train, gray)
7 train = train.reshape(-1, 400).astype(np. float32)
8 print('Get Data for train - Ok')
9
10 test = np.array([])
11 for i in range(9):
12     for j in range(251, 501):
13         img = cv2.imread('./image/digits_dataset2/' + str(i) + '_' + str(j) + '.png')
14         gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
15         test = np.append(test, gray)
16 test = test.reshape (-1, 400).astype (np. float32)
17 print('Get Data for test - Ok')
18

```

```

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

```

```

#train: ตัวเลข 0-6 (250 ภาพแรก: 1-250) ตัวเลข 0-6 (250 ภาพหลัง: 251-500)
train = np.array([])
for i in range(9):
    for j in range(1, 251):
        img = cv2.imread('./image/digits_dataset2/' + 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(9):
    for j in range(251, 501):
        img = cv2.imread('./image/digits_dataset2/' + 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')

```

## Step3/4: การสร้างแบบจำลอง

```
In [5]: 1 k = np.arange(9)
2 train_labels = np.repeat(k, 250)[:, np.newaxis]
3 test_labels = train_labels.copy ()
4
5 knn = cv2.ml.KNearest_create()
6 knn.train(train, cv2.ml.ROW_SAMPLE, train_labels)
7 ret, result, neighbours, dist = knn.findNearest(test, k=5)
8 result.shape, result
9
```

```
Out[5]: ((2250, 1),
array([[0.],
[0.],
[0.],
...,
[8.],
[4.],
[8.]], dtype=float32))
```

```
k = np.arange(9)
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
```

## Step4/4: การทดสอบแบบจำลอง - ตรวจสอบ Accuracy

```
In [6]: 1 matches = result == test_labels
2 correct = np.count_nonzero(matches)
3 accuracy = correct * 100.0 / result.size
4 print('accuracy = ', accuracy)
5
```

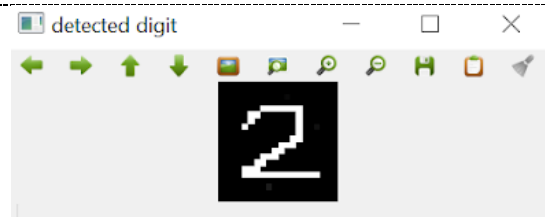
```
accuracy = 93.68888888888888
```

```
matches = result == test_labels
correct = np.count_nonzero(matches)
accuracy = correct * 100.0 / result.size
print('accuracy = ', accuracy)
```

## Step4/4: การทดสอบแบบจำลอง - ตรวจสอบด้วย Unknow X

```
1 mydigit = cv2.imread('./image/unknown_x.png')
2 cv2.imshow('detected digit', mydigit)
3 mydigit_gray = cv2.cvtColor(mydigit, cv2.COLOR_BGR2GRAY)
4 mydigit_test = mydigit_gray.reshape((-1, 400)).astype(np.float32)
5 ret, result, neighbours, dist = knn.findNearest(mydigit_test, k=5)
6 print(ret)
7
8 font = cv2.FONT_HERSHEY_SIMPLEX
9 cv2.putText(mydigit, str(int(ret)), (0, 7), font, 0.3, (255, 0, 0), 1, cv2.LINE_AA)
10
11 cv2.waitKey(0)
12 cv2.destroyAllWindows ()
13
```

2.0



```
mydigit = cv2.imread('./image/unknown_x.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 ()

```

#### Step4/4: การทดสอบแบบจำลอง – ตรวจสอบด้วย Unknow Y

```

1 mydigit = cv2.imread('./image/unknown_y.png')
2 cv2.imshow('detected digit', mydigit)
3 mydigit_gray = cv2.cvtColor(mydigit, cv2.COLOR_BGR2GRAY)
4 mydigit_test = mydigit_gray.reshape((-1,400)).astype(np. float32)
5 ret, result, neighbours, dist = knn. findNearest (mydigit_test, k=5)
6 print(ret)
7
8 font = cv2.FONT_HERSHEY_SIMPLEX
9 cv2.putText(mydigit, str(int(ret)), (0, 7), font, 0.3, (255, 0, 0), 1, cv2.LINE_AA)
10
11 cv2.waitKey (0)
12 cv2.destroyAllWindows ()
13

```

0.0



#### Step4/4: การทดสอบแบบจำลอง – ตรวจสอบด้วย Unknow Z

```

1 mydigit = cv2.imread('./image/unknown_z.png')
2 cv2.imshow('detected digit', mydigit)
3 mydigit_gray = cv2.cvtColor(mydigit, cv2.COLOR_BGR2GRAY)
4 mydigit_test = mydigit_gray.reshape((-1,400)).astype(np. float32)
5 ret, result, neighbours, dist = knn. findNearest (mydigit_test, k=5)
6 print(ret)
7
8 font = cv2.FONT_HERSHEY_SIMPLEX
9 cv2.putText(mydigit, str(int(ret)), (0, 7), font, 0.3, (255, 0, 0), 1, cv2.LINE_AA)
10
11 cv2.waitKey (0)
12 cv2.destroyAllWindows ()
13

```

2.0



#### Step4/4: การทดสอบแบบจำลอง – ตรวจสอบด้วยด้วยการเขียนเอง-X

```

1 mydigit = cv2.imread('./image/myunknown_01.png')
2 cv2.imshow('detected digit', mydigit)
3 mydigit_gray = cv2.cvtColor(mydigit, cv2.COLOR_BGR2GRAY)
4 mydigit_test = mydigit_gray.reshape((-1,400)).astype(np.float32)
5 ret, result, neighbours, dist = knn.findNearest(mydigit_test, k=5)
6 print(ret)
7
8 font = cv2.FONT_HERSHEY_SIMPLEX
9 cv2.putText(mydigit, str(int(ret)), (0, 7), font, 0.3, (255, 0, 0), 1, cv2.LINE_AA)
10
11 cv2.waitKey(0)
12 cv2.destroyAllWindows()
13

```

1.0

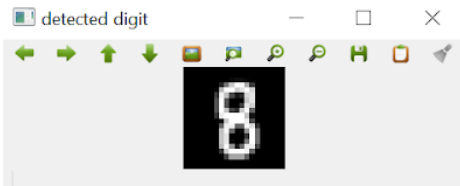


#### Step4/4: การทดสอบแบบจำลอง – ตรวจสอบด้วยการเขียนเอง-Y

```

1 mydigit = cv2.imread('./image/myunknown_02.png')
2 cv2.imshow('detected digit', mydigit)
3 mydigit_gray = cv2.cvtColor(mydigit, cv2.COLOR_BGR2GRAY)
4 mydigit_test = mydigit_gray.reshape((-1,400)).astype(np.float32)
5 ret, result, neighbours, dist = knn.findNearest(mydigit_test, k=5)
6 print(ret)
7
8 font = cv2.FONT_HERSHEY_SIMPLEX
9 cv2.putText(mydigit, str(int(ret)), (0, 7), font, 0.3, (255, 0, 0), 1, cv2.LINE_AA)
10
11 cv2.waitKey(0)
12 cv2.destroyAllWindows()
13

```



#### Step4/4: การทดสอบแบบจำลอง – ตรวจสอบด้วยการเขียนเอง-Z

```

1 mydigit = cv2.imread('./image/myunknown_03.png')
2 cv2.imshow('detected digit', mydigit)
3 mydigit_gray = cv2.cvtColor(mydigit, cv2.COLOR_BGR2GRAY)
4 mydigit_test = mydigit_gray.reshape((-1,400)).astype(np.float32)
5 ret, result, neighbours, dist = knn.findNearest(mydigit_test, k=5)
6 print(ret)
7
8 font = cv2.FONT_HERSHEY_SIMPLEX
9 cv2.putText(mydigit, str(int(ret)), (0, 7), font, 0.3, (255, 0, 0), 1, cv2.LINE_AA)
10
11 cv2.waitKey(0)
12 cv2.destroyAllWindows()
13

```

5.0





```

1  # Edit in paint and save myNbr.jpg
2  # Open and resize to 20x20 pixel
3  # convert image black to white
4
5  import cv2
6  img = cv2.imread('./image/uk_2.png')
7  print("Original > ",img.shape)
8
9  img20x20 = cv2.resize(img, (20,20), interpolation = cv2.INTER_AREA)
10 img20x20 = ~img20x20
11 cv2.imwrite('./image/myunknown_0.png', img20x20)
12 print("rezise > ",img20x20.shape)
13
14 cv2.imshow("myPica", img)
15 cv2.imshow("myPica-20x20", img20x20)
16 cv2.waitKey()
17 cv2.destroyAllWindows()

```

```

# Edit in paint and save myNbr.jpg
# Open and resize to 20x20 pixel
# convert image black to white

```

```

import cv2
img = cv2.imread('./image/uk_2.png')
print("Original > ",img.shape)

img20x20 = cv2.resize(img, (20,20), interpolation = cv2.INTER_AREA)
img20x20 = ~img20x20
cv2.imwrite('./image/myunknown_0.png', img20x20)
print("rezise > ",img20x20.shape)

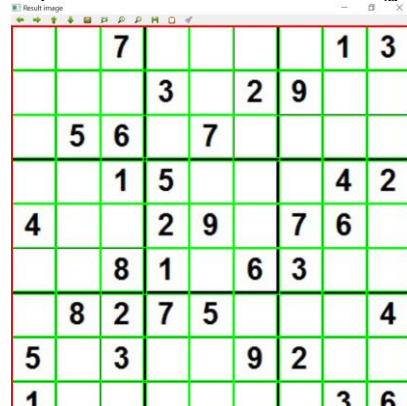
cv2.imshow("myPica", img)
cv2.imshow("myPica-20x20", img20x20)
cv2.waitKey()
cv2.destroyAllWindows()

```

**Quiz\_403 – กิจกรรมที่ 3/6 – Sudoku to Text by Tesseract**

- Capture ผลการทำงานที่ได้ลองปฏิบัติ
- ลองใช้ตารางซูโดกุอื่น ในการทดสอบ
- อภิปรายผล
- คำถามที่อยากถาม
- บอกแนวการใช้งาน กับงานที่รับผิดชอบ

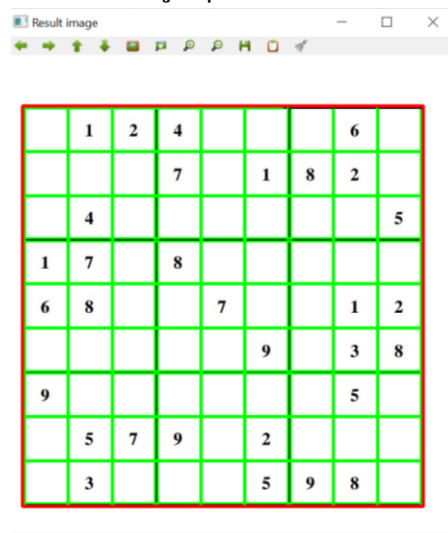
Capture ผลการทำงานที่ได้ลองปฏิบัติ



		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-----13
---3-29--
-56-7-----
--15----42
4--29-76-
--81-63--
-8275----4
5-3--92--
1-----36
```

ลองใช้ตารางซูโดกุอื่น ในการทดสอบ



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

```
- 124 - - 6-
- - - 7- 182-
- 4 - - - - 5
17- 8- - - -
68- - 7- - 12
- - - - - 9- 38
9- - - - - 5-
- 579- 2- - -
- 3- - - 598-
```

```
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/Sudoku_01.jpg")
#imageC = cv2.imread("./image/Sudoku_02.jpg")
imageC = cv2.imread("./image/Sudoku_03.jpg")
#imageC = cv2.imread("./image/Sudoku_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))

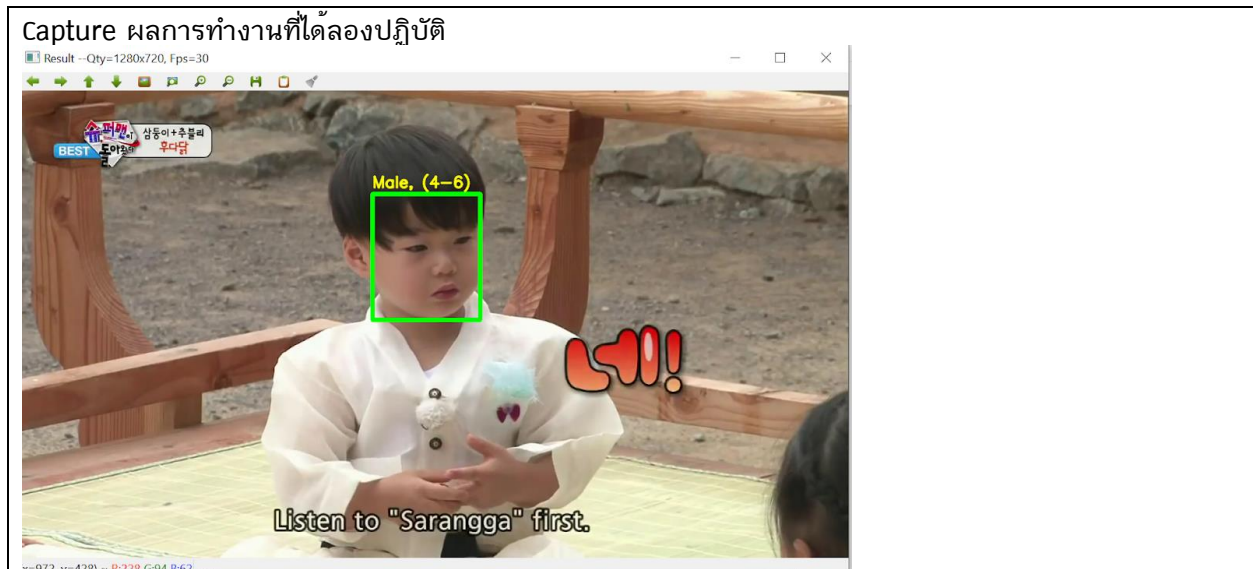
        ROI = imageC[yStart:yStop, xStart:xStop]
        ret,imageB = cv2.threshold(ROI,127,255,cv2.THRESH_BINARY)
        imageX = cv2.cvtColor(imageB, cv2.COLOR_BGR2RGB)
        #cv2.imshow('R-'+str(iRow)+str(jCol),imageX)
        imageP = Image.fromarray(imageX)
        text_from_image = pytesseract.image_to_string(imageP,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

        if len(text_from_image)==0:

```

<pre> 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() </pre>	<p><b>อภิปรายผล</b>  Tesseract คือแปลงรูปเป็นข้อความ กรณีแปลงเลขจากตารางชุดก็จะติดตาราง ทำให้เลขไม่ออกมา จึงมีการจับช่องว่างในตารางแต่ละช่องแทน จากนั้นส่งภาพที่ได้ไปแปลงเป็นข้อความ เอาเข้าเงื่อนไข for เพื่อวนหลักวนแถวให้ครบ ใช้เงื่อนไข if else เพื่อจัดเรียงข้อความออกมาให้ตรงตามตาราง</p> <p><b>คำถามที่อยากถาม</b>  โค้ดสุดยอมากเลยคะ ลองหาเน็ตดูไม่มีเลย นอกจากในเอกสารแล้วยังมีหนังสือหรือเว็บให้อ่านเพิ่มมั้ยคะ</p> <p><b>บอกแนวการใช้งาน</b> กับงานที่รับผิดชอบ  ใช้แปลงข้อความในรูปภาพเป็นข้อความตัวอักษร สามารถนำไปประยุกต์ใช้ได้หลากหลาย เช่น การพิมพ์เอกสารจากไฟล์ pdf เป็นต้น</p>
---	---

### Quiz\_404 – กิจกรรมที่ 4/6 – Gender and Age Detection



```

67 blob=cv2.dnn.blobFromImage(face, 1.0, (227,227), MODEL_MEAN_VALUES, swapRB=False)
68 genderNet.setInput(blob)
69 genderPreds=genderNet.forward()
70 gender=genderList[genderPreds[0].argmax()]
71 print(f'Gender: {gender} - ', end = " ")
72
73 ageNet.setInput(blob)
74 agePreds=ageNet.forward()
75 age=ageList[agePreds[0].argmax()]
76 print(f'Age: {age[1:-1]} years >> ', end = " ")
77
78 cv2.putText(resultImg, f'{gender}, {age}', (faceBox[0], faceBox[1]-10),
79 cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0,255,255), 2, cv2.LINE_AA)
80
81 cv2.imshow(myName + " >> Detecting age and gender - " + str(i), resultImg)
82 print(f"")
83
84 cv2.waitKey(0)
85 cv2.destroyAllWindows()

```

```

Picture: 0 >> Gender: Male - Age: 8-12 years >>
Picture: 1 >> Gender: Male - Age: 0-2 years >>
Picture: 2 >> Gender: Male - Age: 8-12 years >> Gender: Male - Age: 8-12 years >>
Picture: 3 >> Gender: Female - Age: 15-20 years >>
Picture: 4 >> Gender: Male - Age: 15-20 years >>
Picture: 5 >> Gender: Female - Age: 25-32 years >>
Picture: 6 >> Gender: Female - Age: 0-2 years >>
Picture: 7 >> Gender: Male - Age: 38-43 years >>
Picture: 8 >> Gender: Female - Age: 15-20 years >>
Picture: 9 >> Gender: Male - Age: 25-32 years >>
Picture: 10 >> Gender: Male - Age: 8-12 years >>

```

```

import cv2
import math
import argparse
import os

```

```

myName = 'B6226718-Miss Natchaya Phongkuson'
imageC = cv2.imread("./img/1.jpg")

```

```

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

```

```

def load_images_from_folder(folder):
    images = []
    for filename in os.listdir(folder):
        img = cv2.imread(os.path.join(folder,filename))
        if img is not None:
            images.append(img)
    return images

```

```

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)

padding=20
images = load_images_from_folder("./img")

for i in range(len(images)):
    frame = images[i].copy()
    print(f'Picture: {i} >> ', end = "")

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

    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} - ', end = "")

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

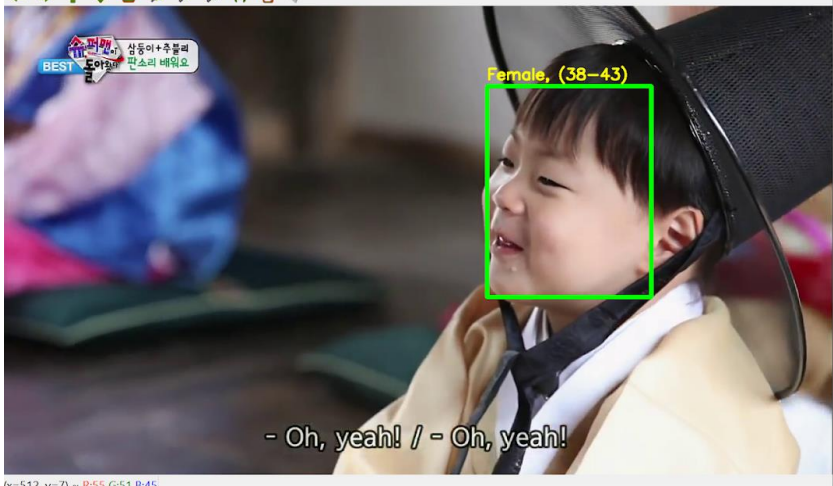
        cv2.putText(resultImg, f'{gender}, {age}', (faceBox[0], faceBox[1]-10),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0,255,255), 2, cv2.LINE_AA)

    cv2.imshow(myName + " >> Detecting age and gender - " + str(i), resultImg)
    print(f'')

cv2.waitKey(0)
cv2.destroyAllWindows()

```

ลองใช้รูปภาพอื่นในการทดสอบ

	
<p>อภิปรายผล ประเด็นความถูกต้องของการระบุเพศ ของ DNN(Deep Neural Network) โมเดลนี้ จะเห็นว่ามีรูปที่ใช้ทดสอบทั้งสองรูปเป็นเด็กคนเดียวกัน แต่มีตำแหน่งและรูปแบบต่างกัน โมเดลนี้ยังระบุเพศได้ไม่แน่นอน มีความคลาดเคลื่อนเนื่องจากการเทรนที่ยังไม่ฉลาดพอ</p>	
<p>อภิปรายผล ประเด็นความถูกต้องของการระบุช่วงอายุ ของ DNN โมเดลนี้ อายุที่ได้มีหลายช่วง เช่น 4-6 และ 38 – 43 พบว่าช่วงค่อนข้างห่างกันมาก มีทั้งช่วงที่ถูกและผิดพลาดมาก ๆ จึงสรุปได้ว่าโมเดลนี้ยังฉลาดไม่มากพอ หากใส่ข้อมูลและได้รับการเทรนเพิ่มจะสามารถประมวลผลได้ดีขึ้น</p>	
<p>คำถามที่อยากถาม ไม่มี</p>	
<p>บอกแนวการใช้งาน กับงานที่รับผิดชอบ</p> <p>สามารถนำไปใช้จับภาพลูกค้าที่เข้ามาใช้บริการร้านได้ เพื่อนำไปทำแผนการตลาดว่าสินค้าตนเหมาะกับกลุ่มลูกค้าประเภทใด</p>	

## Quiz\_405 – กิจกรรมที่ 5/6 – Object Detection and Tracking

### Capture ผลการทำงานที่ได้ลองปฏิบัติ (A) Coin Segmentation

```

1 import cv2
2 import numpy as np
3
4 cap=cv2.VideoCapture("../image/coins.jpg") # Test Coin Picture-1
5 #cap=cv2.VideoCapture("../image/koruny_111.jpg") # Test Coin Picture-2
6 #cap=cv2.VideoCapture("../image/koruny_112.jpg") # Test Coin Picture-3
7 #cap=cv2.VideoCapture("../image/koruny_199.jpg") # Test Coin Picture-3
8 #cap=cv2.VideoCapture("../image/koruny_110.jpg") # Test Coin Picture-4
9 #cap=cv2.VideoCapture("../image/PkCoin.jpg") # Test Coin Picture-5
10
11
12 refFrame = cap.read()
13 roi = frame[1080,0:1920]
14
15 gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
16 gray_blur = cv2.GaussianBlur(gray,(15,15),0)
17 thresh = cv2.adaptiveThreshold(gray,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BINARY_INV,11,1)
18 kernel = np.ones((3,3),np.uint8)
19 closing = cv2.morphologyEx(thresh,cv2.MORPH_CLOSE,kernel,iterations=4)
20
21 result_img = closing.copy()
22 contours,hierarchy = cv2.findContours(result_img,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
23
24 counter = 0
25 for cnt in contours:
26     area = cv2.contourArea(cnt)
27     if area<5000 or area > 35000:
28         continue
29     ellipse = cv2.FitEllipse(cnt)
30     cv2.ellipse(roi,ellipse,(0,255,0),2)
31     counter += 1
32
33 cv2.putText(str(counter),(10,100),cv2.FONT_HERSHEY_SIMPLEX,4,(255,0,0),2,cv2.LINE_AA)
34 cv2.imshow("Show",roi)
35
36 cv2.waitKey()
37 cap.release()
38 cv2.destroyAllWindows()

```

```

import cv2
import numpy as np

cap=cv2.VideoCapture("./image/coins.jpg")    # Test Coin Picture-1
#cap=cv2.VideoCapture("./image/koruny_r11.jpg") # Test Coin Picture-2
#cap=cv2.VideoCapture("./image/koruny_r12.jpg") # Test Coin Picture-3
#cap=cv2.VideoCapture("./image/koruny_r99.jpg") # Test Coin Picture-3
#cap=cv2.VideoCapture("./image/koruny_t10.jpg") # Test Coin Picture-4
#cap=cv2.VideoCapture("./image/PkCoin.jpg")    # Test Coin Picture-5

ref,frame = cap.read()
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)

cv2.waitKey()
cap.release()
cv2.destroyAllWindows()

```

Capture ผลการทำงานที่ได้ลองปฏิบัติ (B) Coin Amouth



```

156 if __name__ == '__main__':
157     picTest = cv2.imread("./image/koruny_r11.jpg") # Picture-1
158     # picTest = cv2.imread("./image/koruny_t10.jpg") # Picture-2
159     # picTest = cv2.imread("./image/koruny_t20.jpg") # Picture-3
160     # picTest = cv2.imread("./image/koruny_2.jpg") # Picture-4
161
162     cv2.imwrite("./image/Show1_Gamma_Input.jpg", picTest)
163     gamma_correction()
164     calculate_amount()
165     coins1 = cv2.imread("./image/Show1_Gamma_Input.jpg")
166     coins2 = cv2.imread("./image/Show2_Gamma_Output.jpg")
167     coins3 = cv2.imread("./image/Show3_Hough_Output.jpg")
168     coins4 = cv2.imread("./image/Show4_Count_Output.jpg")
169     cv2.imshow("Show1_Coin_Input", coins1)
170     cv2.imshow("Show2_Gamma_Output", coins2)
171     cv2.imshow("Show3_Hough_Output", coins3)
172     cv2.imshow("Show4_Count_Output", coins4)
173     cv2.waitKey()
174     cv2.destroyAllWindows()
175
On-Run: Gamma_correction
On-Run: Linear_stretching
Finish: Linear_stretching
Finish: Gamma_correction
On-Run: Calculate_amount
On-Run: Detect_coins
Finish: Detect_coins
The total amount is 88 CZK
1 CZK = 1x
2 CZK = 1x
5 CZK = 1x
10 CZK = 1x
20 CZK = 1x
50 CZK = 1x
Finish: Calculate_amount

```

# 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": 1.05,
            "count": 0,
        },
        "5 CZK": {
            "value": 5,
            "radius": 23,
            "ratio": 1.10,
            "count": 0,
        },
        "10 CZK": {
            "value": 10,
            "radius": 24.5,

```

```

        "ratio": 1.220,
        "count": 0,
    },
    "20 CZK": {
        "value": 20,
        "radius": 26,
        "ratio": 1.29,
        "count": 0,
    },
    "50 CZK": {
        "value": 50,
        "radius": 27.5,
        "ratio": 1.35,
        "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)

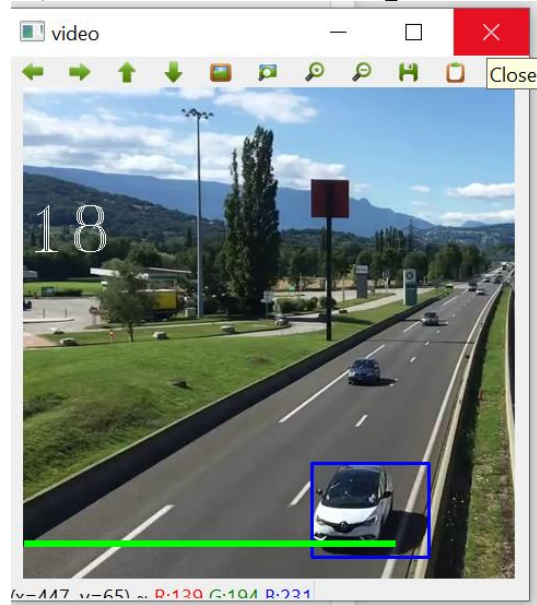
```

```

cv2.imshow("Show3_Hough_Output", coins3)
cv2.imshow("Show4_Count_Output", coins4)
cv2.waitKey()
cv2.destroyAllWindows()

```

### Capture ผลการทำงานที่ได้ลองปฏิบัติ (C) Car Counting



```

105 video.release()
106 cv2.destroyAllWindows()
107
108
109

```

```

result : 2.725663716814159
result : 2.353982300884956
result : 2.47787610619469
result : 2.584070796460177
result : 2.725663716814159
result : 2.601769911504425
result : 2.7079646017699117
result : 1.9469026548672566
result : 2.0353982300884956
result : 2.15929203539823
result : 2.2831858407079646
result : 2.1976401179941
result : 2.725663716814159
result : 2.1946902654867255
result : 0.6666666666666666
result : 2.088495575221239
result : 2.1769911504424777
result : 2.2831858407079646
result : 2.3893805309734515
result : 0.6489675516224189
result : 2.017699115044248
result : 2.088495575221239
result : 2.1946902654867255
result : 2.3185840707964602
result : 0.5014749262536873
result : 1.8407079646017699
result : 1.7345132743362832
result : 2.1769911504424777
result : 2.336283185840708
result : 1.7168141592920354
result : 1.7168141592920354
result : 1.7168141592920354
result : 1.0147492625368733
result : 1.0923451322433628

```

```

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

```

```

video=cv2.VideoCapture("./image/CarVideo_01.mp4")
ret,frame=video.read()
cropped_image= frame[0:450, 0:450]
fgbg=cv2.createBackgroundSubtractorMOG2()
Sensor1 = Sensor(
    Kordinat(1, cropped_image.shape[1] - 35),
    Kordinat(340, cropped_image.shape[1] - 30),
    cropped_image.shape[0],
    cropped_image.shape[1])

kernel=np.ones((5,5),np.uint8)
font=cv2.FONT_HERSHEY_TRIPLEX
while (1):
    ret,frame=video.read()

```

```

# resize frame
cropped_image= frame[0:450, 0:450]
# make morphology for frame
deleted_background=fgbg.apply(cropped_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=cropped_image.copy()

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

# detect moving anything with loop
for cnt in contours:
    x,y,w,h=cv2.boundingRect(cnt)
    if (w>75 and h>75 and w<160 and h<160 ):
        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("result : ",sensor_rate)
# if car is passing under the red line sensor . red line sensor is yellow color.

if (sensor_rate>=0.9 and sensor_rate<3.1 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<0.9 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,150),font,2,(255,255,255))

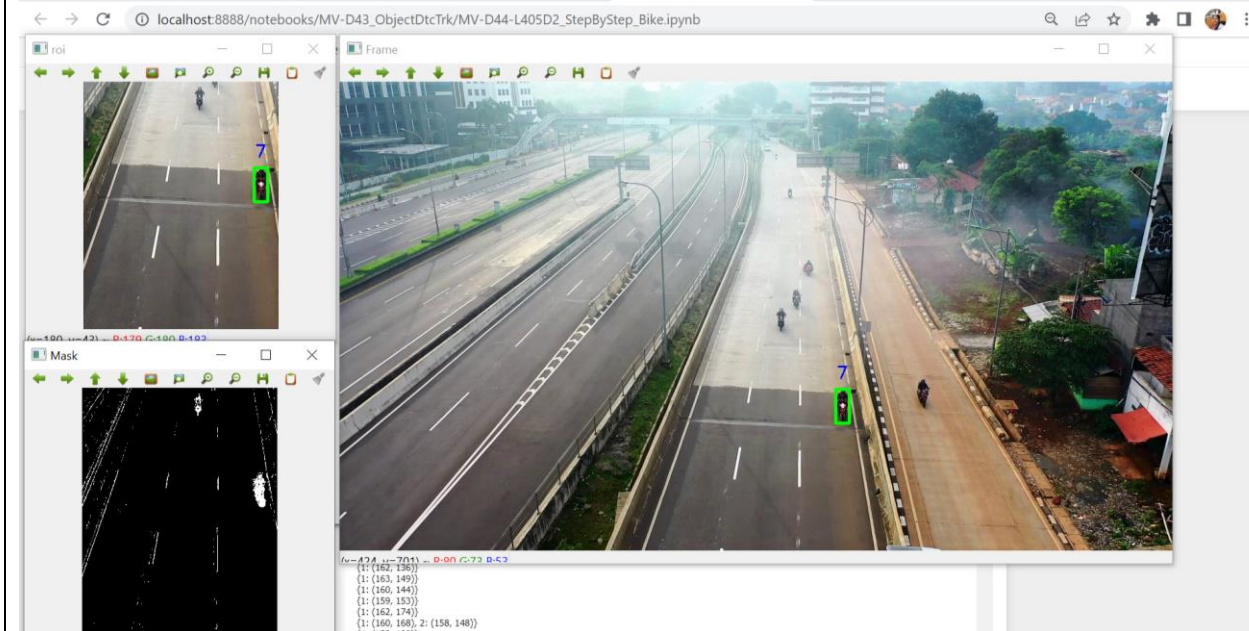
cv2.imshow("video", result)
#cv2.imshow("mask_result", mask_result)
#cv2.imshow("zeros_image", zeros_image)
#cv2.imshow("opening_image", opening_image)

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

video.release()
cv2.destroyAllWindows()

```

## Capture ผลการทำงานที่ได้ลองปฏิบัติ (D) Car Detection and Tracking



```

28 #cv2.drawContours(roi, [cnt], -1, (0, 255, 0), 2)
29 x, y, w, h = cv2.boundingRect(cnt)
30 detections.append([x, y, w, h])
31
32 # 2. Object Tracking
33 boxes_ids = tracker.update(detections)
34 for box_id in boxes_ids:
35     x, y, w, h, id = box_id
36     cv2.putText(roi, str(id), (x, y - 15), cv2.FONT_HERSHEY_PLAIN, 2, (255, 0, 0), 2)
37     cv2.rectangle(roi, (x, y), (x + w, y + h), (0, 255, 0), 3)
38
39 cv2.imshow("roi", roi)
40 cv2.imshow("Frame", frame)
41 cv2.imshow("Mask", mask)
42
43 key = cv2.waitKey(30)
44 if key == 27:
45     break
46
47 cap.release()
48 cv2.destroyAllWindows()
49

```

```

{0: (163, 79)}
{1: (161, 122)}
{1: (163, 127)}
{1: (162, 136)}
{1: (163, 149)}
{1: (160, 144)}
{1: (159, 153)}
{1: (162, 174)}
{1: (160, 168), 2: (158, 148)}
{1: (159, 180)}
{1: (158, 190)}
{1: (158, 198)}
{1: (158, 210)}
{1: (157, 220)}
{1: (156, 232)}
{1: (156, 244)}
{1: (156, 256)}
{1: (155, 269)}
{1: (154, 284)}

```

```

#Step-05,06
import cv2
from moving_tracker import *

# Create tracker object
tracker = EuclideanDistTracker()
cap = cv2.VideoCapture("./image/highway.mp4")

# Object detection from Stable camera
object_detector = cv2.createBackgroundSubtractorMOG2(history=100, varThreshold=40)

while True:
    ret, frame = cap.read()
    height, width, _ = frame.shape

    # Extract Region of interest
    roi = frame[340: 720, 500: 800]

```

```

# 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 > 100:
        #cv2.drawContours(roi, [cnt], -1, (0, 255, 0), 2)
        x, y, w, h = cv2.boundingRect(cnt)
        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("Frame", frame)
cv2.imshow("Mask", mask)

key = cv2.waitKey(30)
if key == 27:
    break

cap.release()
cv2.destroyAllWindows()

```

#### อภิปรายผลการทดสอบ แต่ละหัวข้อ (A, B, C, D)

A คือการตีกรอบให้เหรียญ ทำได้โดยการจับความต่างของสีจากภาพแล้วกำหนดเงื่อนไขให้จับในพื้นที่ที่เป็นวงกลม สามารถกำหนดขนาดของรัศมีได้

B คือการนับเหรียญในรูปว่ามีมูลค่าเท่าไร สามารถทำได้โดยการตั้งเงื่อนไขว่า วงกลมที่มีขนาดอยู่ในช่วงนี้มีราคาเท่าไร ทำให้โปรแกรมสามารถแยกมูลค่าของเหรียญได้

C คือการนับรถ ใช้วิธีนับรถที่ผ่านในขอบเขตที่กำหนดไว้ วิธีการนับจะมีทั้งแบบขับรถเข้าเส้นและขับรถออกจากเส้น

D คือการจับภาพรถและนับ โดยสามารถกำหนดเงื่อนไขของรถที่ต้องการนับได้ เช่น นับรถยนต์ไม่นับรถจักรยานยนต์ วิธีนี้จะคล้ายๆกับ C แต่มีส่วนที่ต่างกันคือเงื่อนไขการตีกรอบ ให้กำหนดขนาดของความต่างของสีว่าต้องอยู่ในวงไหน เพื่อให้มันตรงเงื่อนไขของสิ่งที่ต้องการนับ

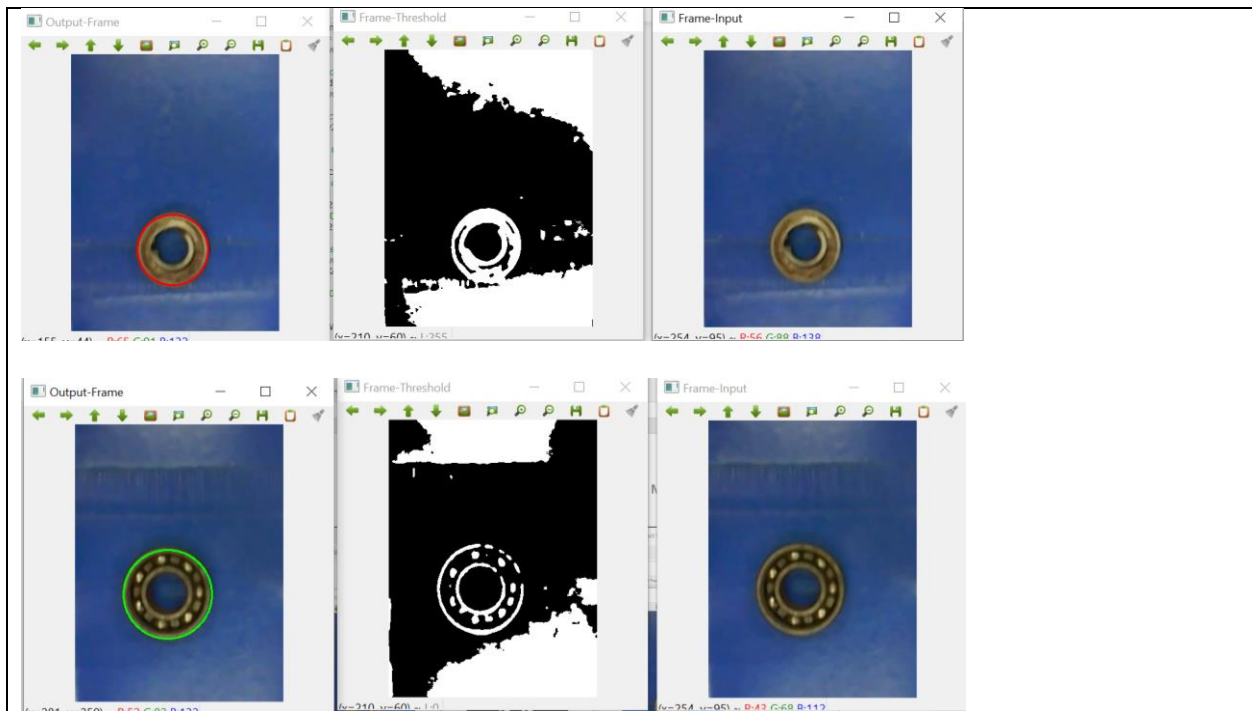
#### บอกแนวการใช้งาน กับงานที่รับผิดชอบ

สามารถนำไปประยุกต์ใช้ได้หลายหลาย ทางด้านการนับจำนวนต่างๆและอื่นๆตามที่ต้องการใช้ เช่น การนับคนถืออาวุธก็สามารถใช้เทคนิคนี้จับภาพอาวุธได้ สามารถระบุชนิดอาวุธได้ ตามเงื่อนไขและการเทรนที่กำหนด

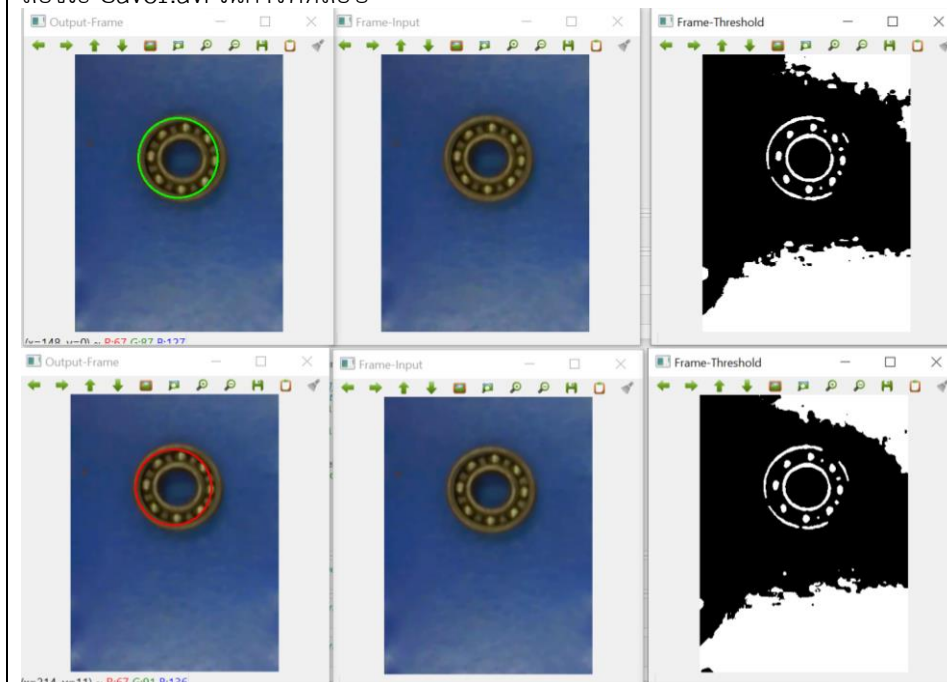
### Quiz\_406 – กิจกรรมที่ 6/6 – Visual Inspection

Capture ผลการทำงานที่ได้ลองปฏิบัติ





ลองใช้ข้อมูลอื่นในการทดสอบ  
ลองใช้ Save1.avi ในการทดสอบ



#### อภิปรายผล

จากการทดสอบจับขนาดของเหรียญที่ผ่านสายพาน โดยให้เป็นวงกลมสีเขียวเมื่อมีขนาดใหญ่ และสีแดงเมื่อมีขนาดเล็กพบว่า โค้ดที่เขียนสามารถใช้งานได้ แต่ประสิทธิภาพจะขึ้นอยู่กับความเร็วด้วย กรณีที่สายพานเดินไวมากการจับภาพก็จะแปรปรวนดังข้อที่ใช้ข้อมูลอื่นทดสอบ

#### คำถามที่อยากถาม

เหรียญที่ใช้คือเหรียญอะไรคะ ไม่เคยเห็นเลย หรือมันคือนอต



บอกแนวการใช้งาน กับงานที่รับผิดชอบ  
สามารถใช้นับจำนวนวัตถุที่มีขนาดกลมได้ กำหนดเงื่อนไขความกลมเพื่อใช้แยกความแตกต่าง