การพัฒนาโปรแกรมประยุกต์และปัญญาประดิษฐ์ เพื่อการมองเห็นของเครื่องจักร
Computer Programing and Artificial Intelligence in Machine Vision

4/4 – Machine Learning + Case Study

- Artificial Intelligence, Machine Learning and Deep Learning
- การเรียนรู้ของเครื่องจักร (Machine Learning)
- 10-Basic Machine Learning Algorithm
- Case Study 1 -- Sudoku to Text
- Case Study 2 -- Gender and Age Detection
- Case Study 3 -- Object Detection and Tracking
- Case Study 4 -- Visual Inspection
- คำถามท้ายบทเพื่อทดสอบความเข้าใจ

6/8 -- Case Study 3 -- Object Detection and Tracking

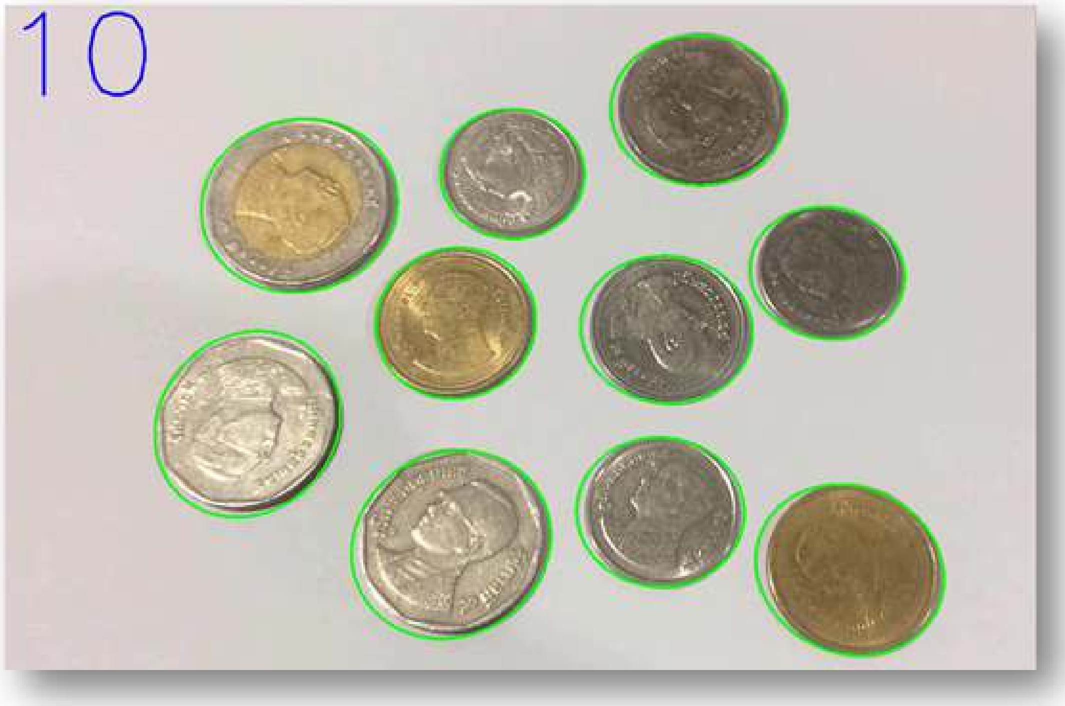
Lab405a: Coil Segmentation

https://www.youtube.com/watch?v=KRZIV1RBHSI
https://kongruksiamza.medium.com/coin-secmentation-python-opencv-9c7a9537002c

1. ตรวจจับและค้นหาเหรียญจากภาพ วิดีโอ กล้องแบบเรียลไทม์ด้วย Python + OpenCV สำหรับท่านที่สนใจนำไปพัฒนาต่อได้

2. ที่ต้องใช้ ไลบราลี่

   - pip install opencv-python
   - pip install numpy



3. หลักการและทฤษฎี

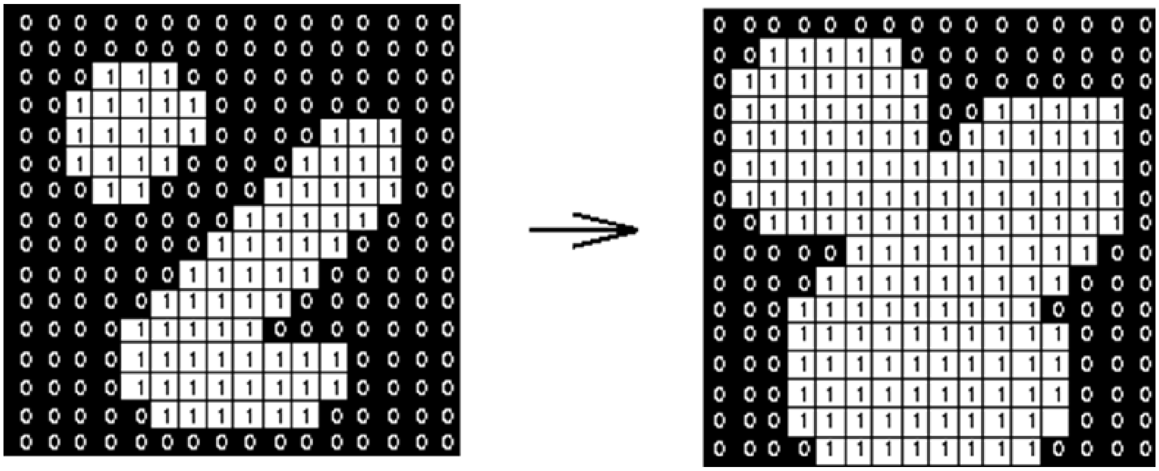
อาศัยหลักการประมวลผลภาพกับรูปร่างและโครงสร้างของภาพโดยนำคณิตสัณฐานวิทยา (MM : Mathematical Morphology) เพื่อประมวลผลภาพตามหลักทฤษฎีแลตติช เป็นเทคนิคสำหรับการวิเคราะห์และประมวลผลโครงสร้างทางเรขาคณิตบนพื้นฐานของทฤษฎีเซต ทฤษฎีตาข่าย โครงสร้างของเครือข่ายและฟังก์ชั่นแบบสุ่ม นิยมนำมาใช้งานกับภาพดิจิตอลเพื่อวิเคราะห์พื้นผิว ขนาด รูปร่าง พื้นที่นูน การเชื่อมต่อโดยอาศัยตัวดำเนินการ 4 ลักษณะ

การประมวลผลภาพกับรูปร่างและโครงสร้างภาพ เป็นเทคนิคของการประมวลผลภาพที่ขึ้นอยู่กับรูปร่าง ค่าของ Pixels ในภาพ Output ขึ้นอยู่กับการเปรียบเทียบของ Pixels ที่สอดคล้องกันในภาพ Input กับพื้นที่ใกล้เคียง โดยเลือกขนาดและรูปร่างของพื้นที่ใกล้เคียงมาสร้างการดำเนินการทางรูปร่างและโครงสร้างของภาพต่อไป

ซึ่งประกอบไปด้วย

- การขยายภาพ (Dilation)
- การกร่อนภาพ (Erosion)
- การเปิดภาพ (Opening)
- การปิดภาพ (Closing)

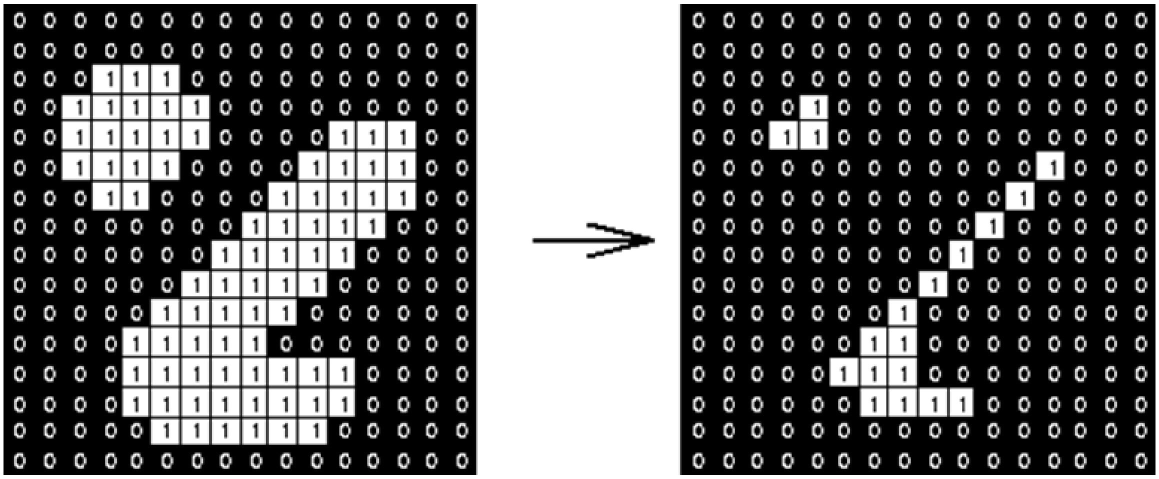
4. **การขยายภาพ (Dilation)** สำหรับตรวจสอบและขยายรูปทรงที่มีอยู่ในภาพ Input



```
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 1 1 1 1 1 0 0 0 0 0 0 0 0 0
0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0        0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0        0 1 1 1 1 1 1 1 0 0 1 1 1 1 1 0
0 0 1 1 1 1 1 0 0 0 0 1 1 1 0 0        0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0
0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0        0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
0 0 0 1 1 0 0 0 0 1 1 1 1 1 0 0        0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0   →    0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0
0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0        0 0 0 0 0 1 1 1 1 1 1 1 1 1 0 0
0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0        0 0 0 0 1 1 1 1 1 1 1 1 1 0 0 0
0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0        0 0 0 1 1 1 1 1 1 1 1 1 0 0 0 0
0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0        0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0
0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0        0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0
0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0        0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0
0 0 0 0 0 1 1 1 1 1 1 0 0 0 0 0        0 0 0 1 1 1 1 1 1 1 1 1   0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0        0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0
```

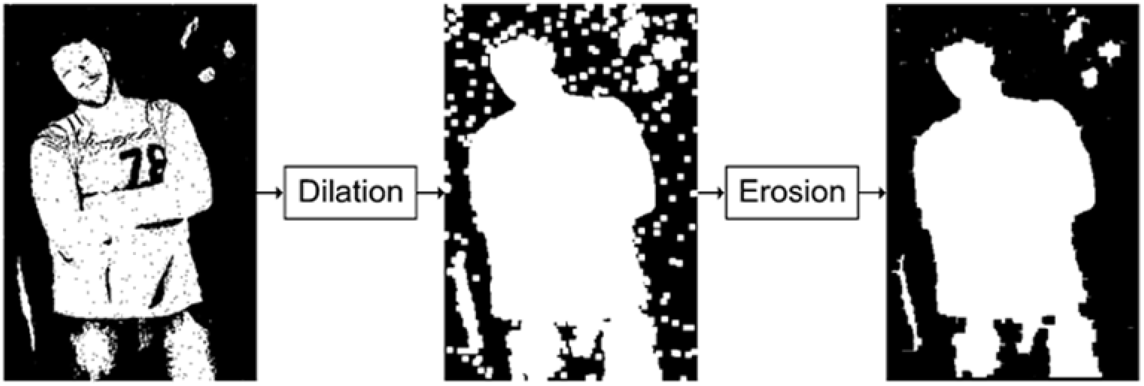
- ถ้าเป็นภาพสีเทา จะขยายภาพและเพิ่มความสว่างของวัตถุโดยใช้พื้นที่ใกล้เคียงสูงสุด
- ถ้าเป็นภาพไบนารี จะขยายภาพและเชื่อมต่อพื้นที่ที่แยกออกจากกันด้วยช่องว่างที่มีขนาดเล็กกว่าองค์ประกอบโครงสร้างและเพิ่ม Pixels เข้าไปที่ขอบด้านนอกของแต่ละวัตถุในภาพ

5. **การกร่อนภาพ (Erosion)** => สำหรับลดขนาดของวัตถุและความผิดปกติเล็กๆโดยลบวัตถุที่มีรัศมีเล็กกว่าองค์ประกอบของโครงสร้าง

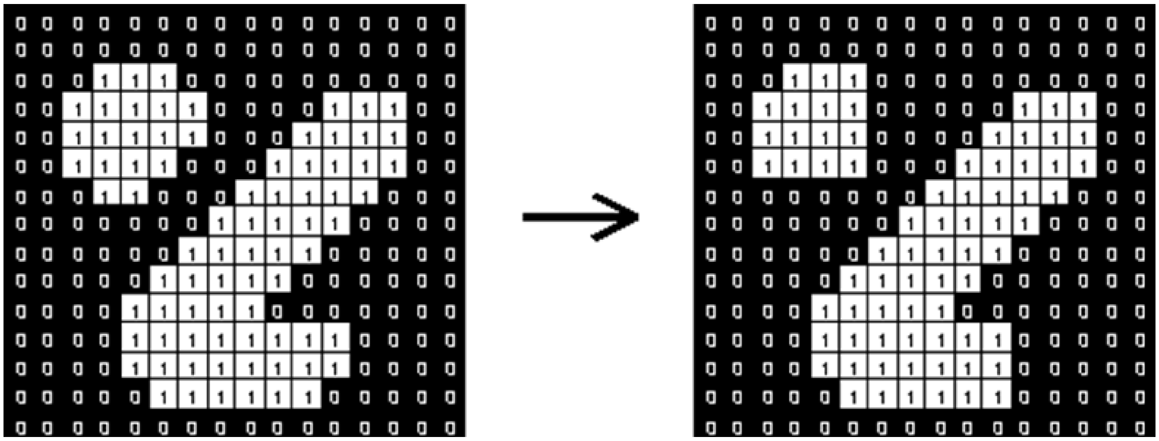


```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0        0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
0 0 1 1 1 1 1 0 0 0 0 1 1 1 0 0        0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0
0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0        0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 1 1 0 0 0 0 1 1 1 1 1 0 0        0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0   →    0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0        0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0        0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0        0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0        0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0
0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0        0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0
0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0        0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0
0 0 0 0 0 1 1 1 1 1 1 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
```

- ถ้าเป็นภาพสีเทา จะลดขนาดวัตถุและลดความสว่างของวัตถุบนพื้นหลังสีเข้มโดยใช้พื้นที่ใกล้เคียงต่ำสุด
- ถ้าเป็นภาพไบนารี ลบวัตถุที่มีขนาดเล็กกว่าองค์ประกอบของโครงสร้างออกและลบ Pixels ที่ขอบด้านนอกออกจากวัตถุในภาพ



6. **การเปิดภาพ (Opening)** ใช้สำหรับกำจัดสัญญานรบกวนในรูปร่างและโครงสร้างของภาพกำจัดวัตถุขนาดเล็กออกไปจากพื้นที่มืดของภาพและนำไปวางไว้ในพื้นหลังของภาพ ทำให้ Pixels ภาพถูกเปิดกว้างมากขึ้น การเปิดภาพนิยมใช้ในการค้นหาองค์ประกอบของโครงสร้างเช่น ขอบภาพและมุมภาพ

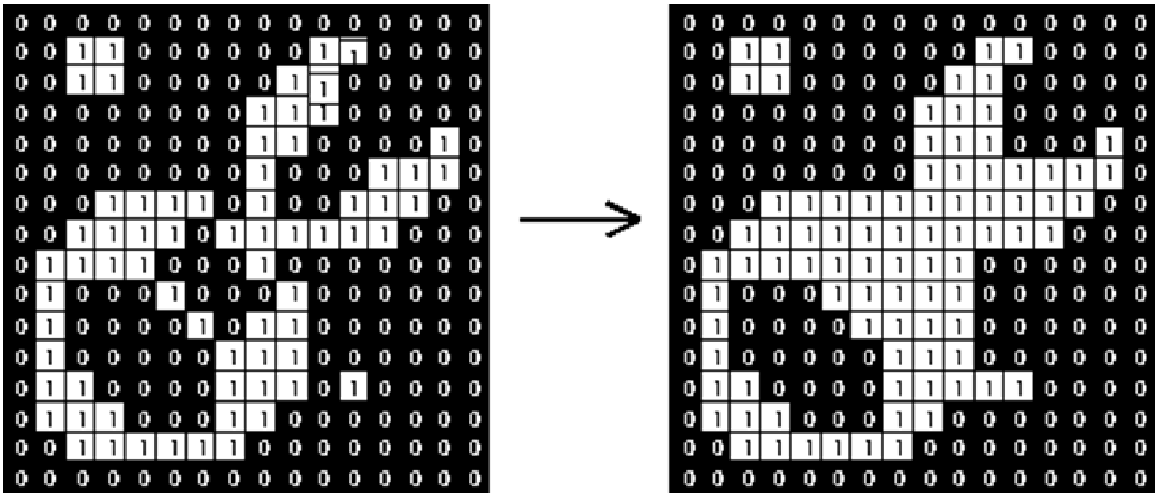


0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	0	0	0	0	1	1	1	0	0
0	0	1	1	1	1	1	0	0	0	1	1	1	1	0	0
0	0	1	1	1	1	0	0	0	1	1	1	1	1	0	0
0	0	0	1	1	0	0	0	1	1	1	1	1	0	0	0
0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0
0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0
0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0
0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
0	0	0	0	0	1	1	1	1	1	1	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	1	1	1	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	0	0	0	0	0	1	1	1	0	0
0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0
0	0	1	1	1	1	0	0	0	1	1	1	1	1	0	0
0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0
0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0
0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0
0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0
0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0
0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0
0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

7. **การปิดภาพ (Closing)** => กระทำในทางตรงกันข้ามกับ Opening โดยเป็นการทำให้ Pixels ของภาพเชื่อมต่อกันมากขึ้น



0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0	1	1	0	0	0	0
0	0	1	1	0	0	0	0	0	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	0
0	0	0	1	1	1	1	0	1	0	0	1	1	1	0	0
0	0	1	1	1	1	0	1	1	1	1	1	1	0	0	0
0	1	1	1	1	0	0	0	1	0	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0
0	1	0	0	0	0	1	0	1	1	0	0	0	0	0	0
0	1	0	0	0	0	0	1	1	1	0	0	0	0	0	0
0	1	1	0	0	0	0	1	1	1	0	1	0	0	0	0
0	1	1	1	0	0	0	1	1	0	0	0	0	0	0	0
0	0	1	1	1	1	1	1	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	1	1	0	0	0	0	0	0	1	1	0	0	0	0
0	0	1	1	0	0	0	0	0	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1	1	0	0	0	1	0
0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0
0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0
0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0
0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0
0	1	0	0	0	1	1	1	1	1	0	0	0	0	0	0
0	1	0	0	0	0	1	1	1	1	0	0	0	0	0	0
0	1	0	0	0	0	0	1	1	1	0	0	0	0	0	0
0	1	1	0	0	0	0	1	1	1	1	1	0	0	0	0
0	1	1	1	0	0	0	1	1	0	0	0	0	0	0	0
0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

การเปิดและปิดภาพเป็นการกำจัดสัญญานรบกวนในรูปร่างและโครงสร้างของภาพ

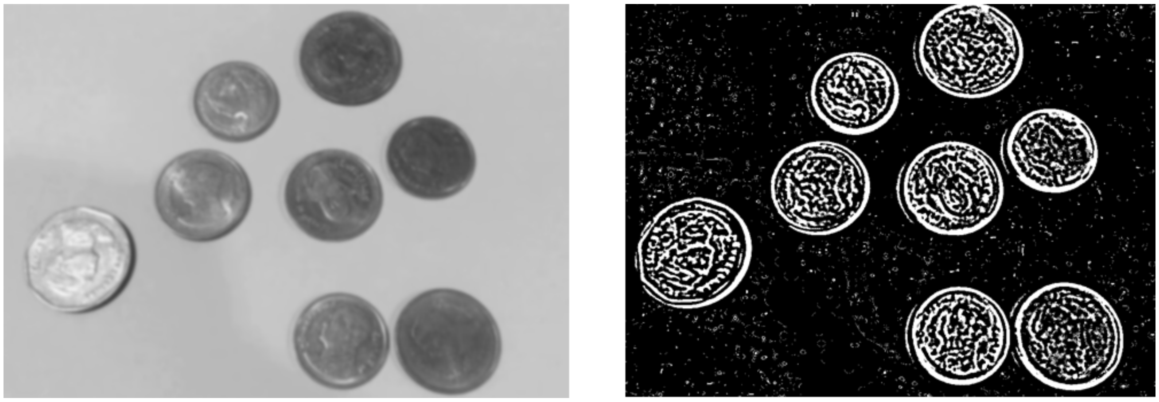
- การเปิด = > กำจัดวัตถุขนาดเล็ก
- การปิด => กำจัดช่องโหว่เล็กๆและเชื่อมต่อ

8. **เริ่มต้นลุยโปรเจค**

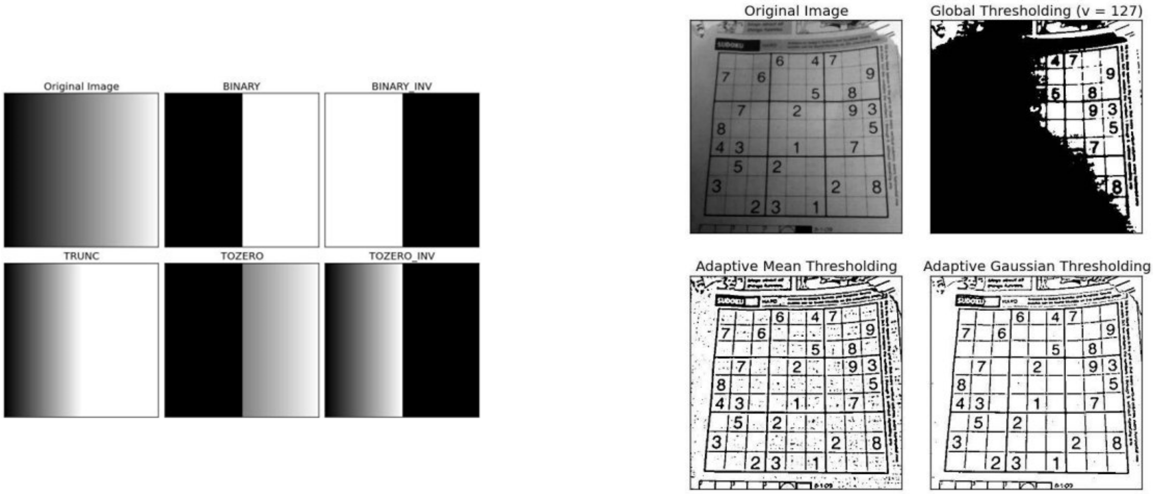
- **NumPy (Numeric Python)** เป็นโมดูลส่วนเสริมของ Python มีฟังก์ชั่นเกี่ยวกับคณิตศาสตร์และการคำนวณต่างๆ มาให้ใช้งาน ใช้การจัดการข้อมูลชุด (Array) ขนาดใหญ่และเมทริกซ์ (สำหรับสร้างตาราง Matrix ในการประมวลผลภาพ)
- **Arrays ของ NumPy** มีลักษณะคล้ายกับ list แต่สมาชิกทุกตัวใน array จะต้องเป็นข้อมูลชนิดเดียวกัน โดยทั่วไปแล้วข้อมูลที่เก็บจะเป็นตัวเลขเช่น int หรือ float
    - คล้าย List แต่ทำไมใช้ Array ???
    - Arrays มีความสามารถในการดำเนินการเกี่ยวกับข้อมูลที่เป็นตัวเลขจำนวนมากๆ ได้อย่างรวดเร็วและมีประสิทธิภาพมากกว่า list

8.1 อ่านภาพจากวิดีโอ & กล้อง

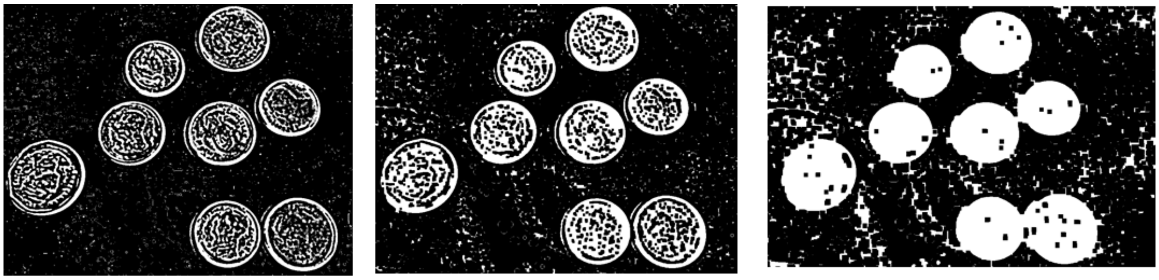
8.2 Gaussian Blur & Threshold



8.3 Adaptive Thresholding



8.4 Morphological Closing

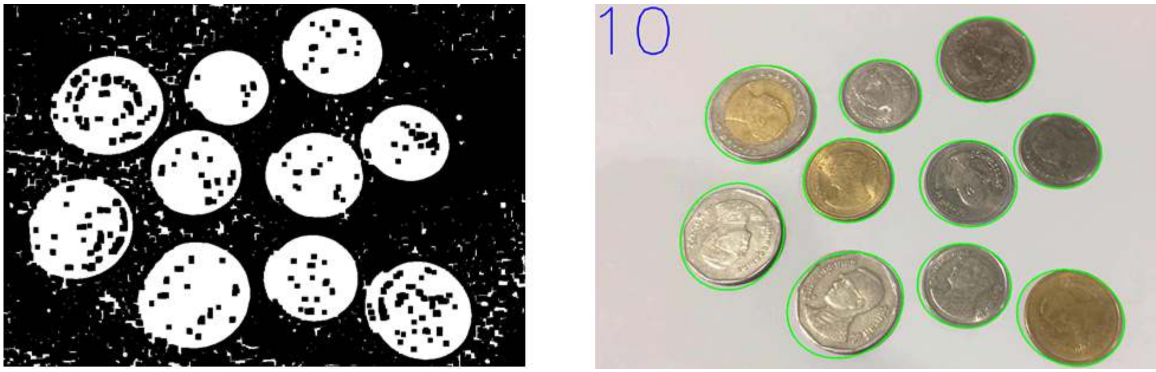


1 x 1

2 x 2

3 x 3

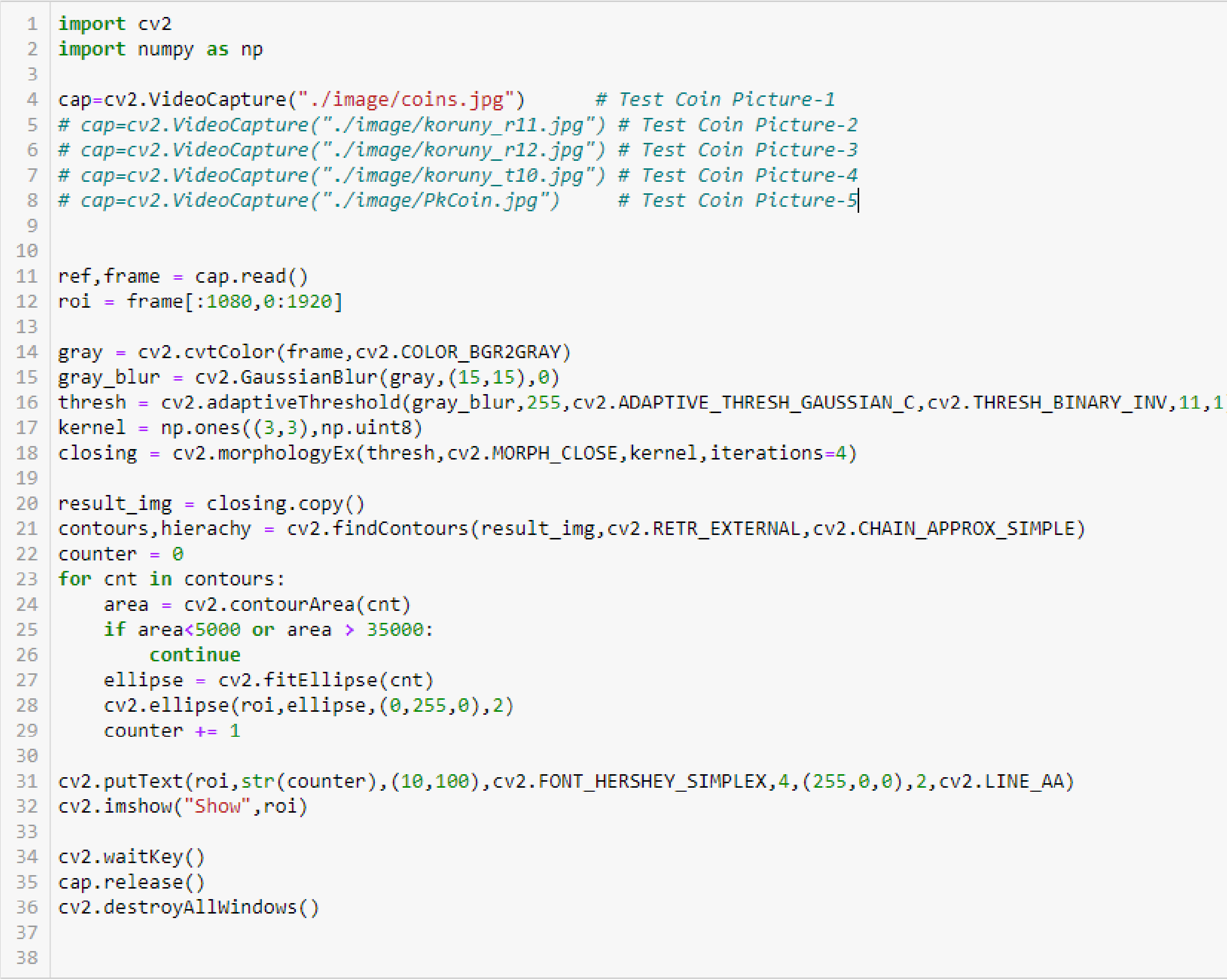
8.5 แสดงรูปร่างและกรองพื้นที่เหรียญ



9. Github Project : https://github.com/kongruksiamza/Coin-Segmentation

10. ทดสอบการทำงานด้วยภาพนิ่ง(Prog_L405A1)

Test01: Prog_L405A1_Coil Segmentation



```python
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_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()
```

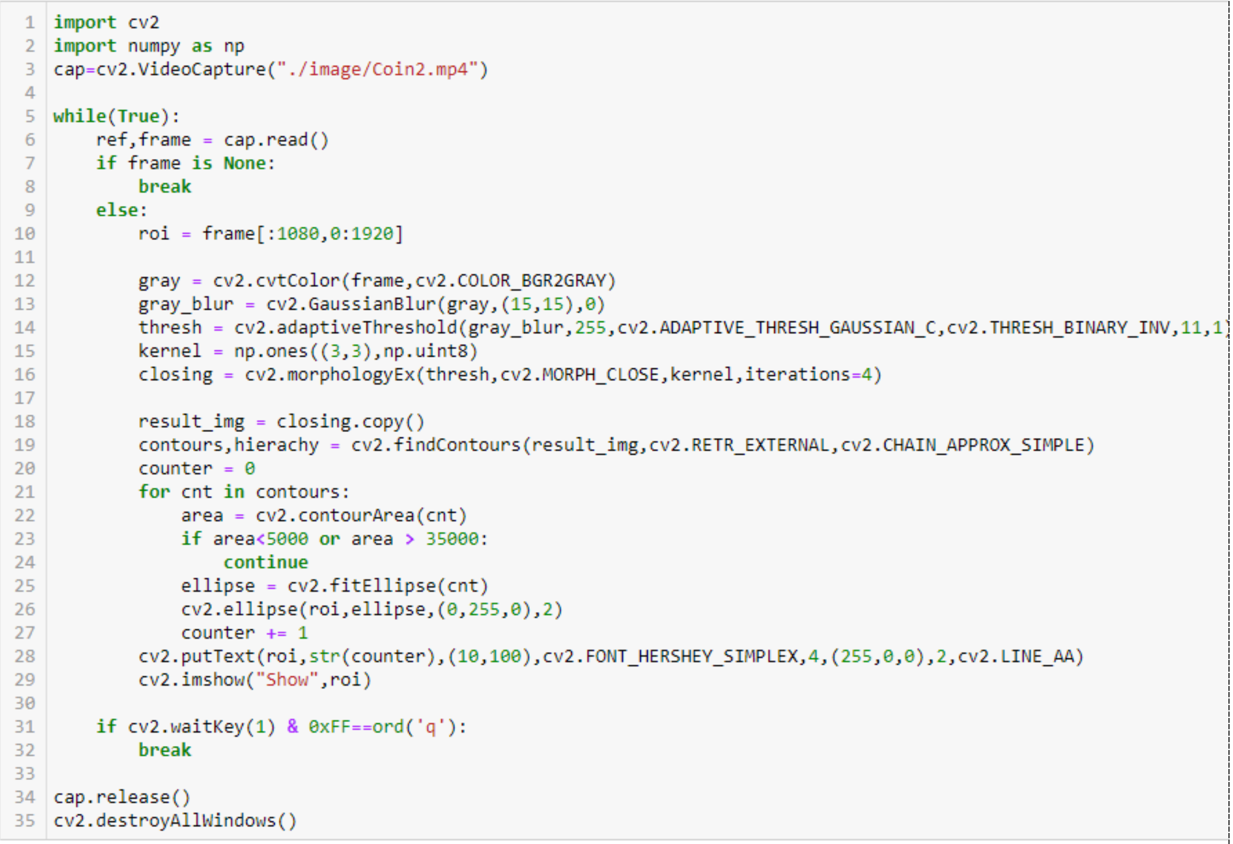
```
import cv2
import numpy as np

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_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()
```

11. คำถามท้ายการทดสอบ

- 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
    - จุดแตกต่างของ P1(koruny_r11.jpg) และ P2(koruny_r12.jpg) คือ อะไร
    - ปรับพารามิเตอร์ให้ถูกต้อง ปรับอะไรบ้าง
    - พารามิเตอร์ P1 กับ P2 ทำงานด้วยกันได้หรือไม่
- cap=cv2.VideoCapture("./image/koruny_t10.jpg")   # Test Coin Picture-4
    - ปรับพารามิเตอร์ให้ถูกต้อง ปรับอะไรบ้าง
- cap=cv2.VideoCapture("./image/PkCoin.jpg")   # Test Coin Picture-5
    - ปรับพารามิเตอร์ให้ถูกต้อง ปรับอะไรบ้าง

12. ทดสอบการทำงานด้วยภาพเคลื่อนไหว(Prog_L405A)

Test02: Prog_L405A1_Coil Segmentation

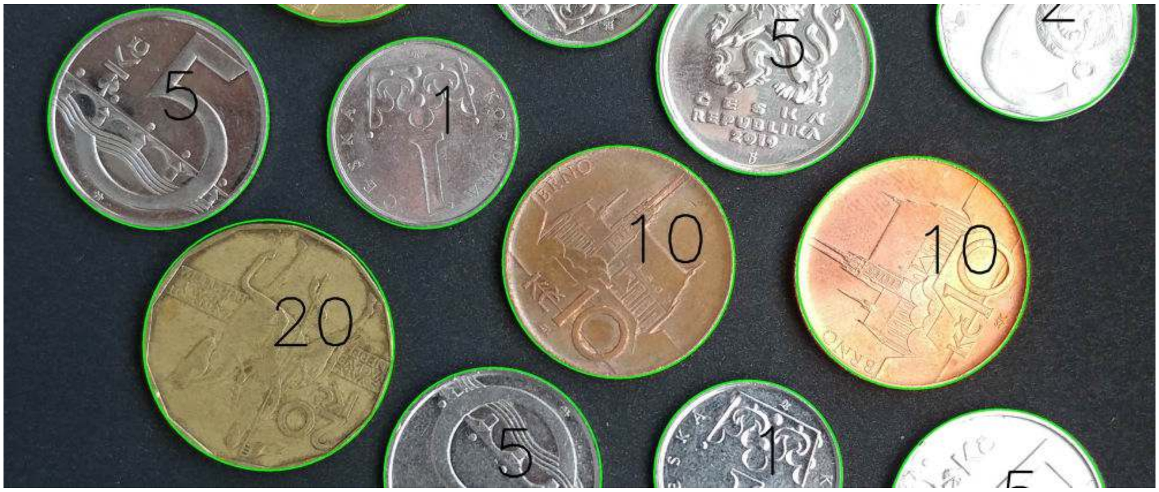


```python
import cv2
import numpy as np
cap=cv2.VideoCapture("./image/Coin2.mp4")

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()
```

```python
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()
```

Lab405b: Coin Amount Calculation

https://dev.to/tinazhouhui/coin-amount-calculation-discovering-opencv-with-python-52gn
https://tremaineconsultinggroup.com/opencv-coin-detection-project/
https://blog.christianperone.com/2014/06/simple-and-effective-coin-segmentation-using-python-and-opencv/



Since I already explored coin detection, I decided to take the real-life application of OpenCV one step further. Now that I can find the coins, naturally, the next step would be to correctly identify the coins and subsequently calculate their amount.

Detect coins

As previously explored in a separate article here, detecting the coins is the first step. I used the Hough Circle Transformation to find these and therefore had the radius of each coin and the coordinates of the center. The logic behind the value identification lies in the radii of each coin as we have only visual information, therefore the precision of the circles drawn around the coins needed to be high.

Identify coins

Since each picture can be taken from a different height, we cannot directly translate the number of pixels to millimetres. Therefore the identification of coins had to be relative based on their radii.

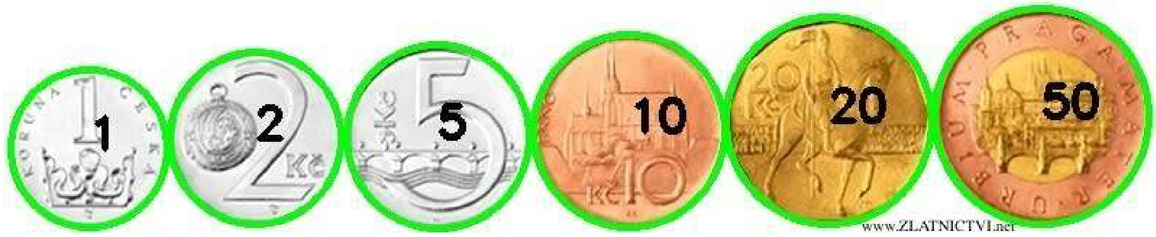
For example, The smallest coin (1 CZK) has a radius of 20 mm and the second to smallest coin (2 CZK) has a radius of 21.5 mm. Therefore, the 2 CZK coin’s radius is 1.075 times larger than the radius of the 1 CZK coin. Amount of pixels representing the radii in the picture must, therefore, also follow the same ratio. This logic works with any coin, but you have to let the program know, which coin is the base coin that you derive the ratios from. In my case, it was easiest to say that the smallest coin on the picture represents the smallest coin in real life so each analysed image had to have at least one 1 CZK coin.

From there, I just created a dictionary of all important information related to each coin - name, value, count, radius, ratio-to-smallest-coin and ran a for cycle for each of the coins, or to be precise, circles found by the Hough circle Transformation.

Now, all that was left was that each time a coin is identified, the value is written to the center of the coin and added to the total value variable. After running through all the coins, the total amount is calculated. As usual, all work can be found here, on GitHub.

Let’s test it!

First I tested the program on the most nicely scanned coins that the internet could offer and found this picture. After some tweaking with the parameters in Hough Transformation, I got this result:



```
The total amount is 88 CZK
1 CZK = 1x
2 CZK = 1x
5 CZK = 1x
10 CZK = 1x
20 CZK = 1x
50 CZK = 1x
```

This gave me the confidence to try to test it on a picture of coins that I took at home. I thought, just as with the picture above, that few tries with the parameters would result in the correct amount.

Unfortunately, that was not the case. See, the biggest problem was that Hough could not detect the circles well enough to fit the ratio. It either made the circle too small and hence thought it was smaller value or the opposite or did not detect them at all. I tried changing the background from white to black, different distance to take the picture, different lighting and still nothing. It took me days of trying almost every combination of parameters to realise that this was not the way. The secret lies in preprocessing the image.

The most important realization was that the output could be only as good as the input data that we are providing.

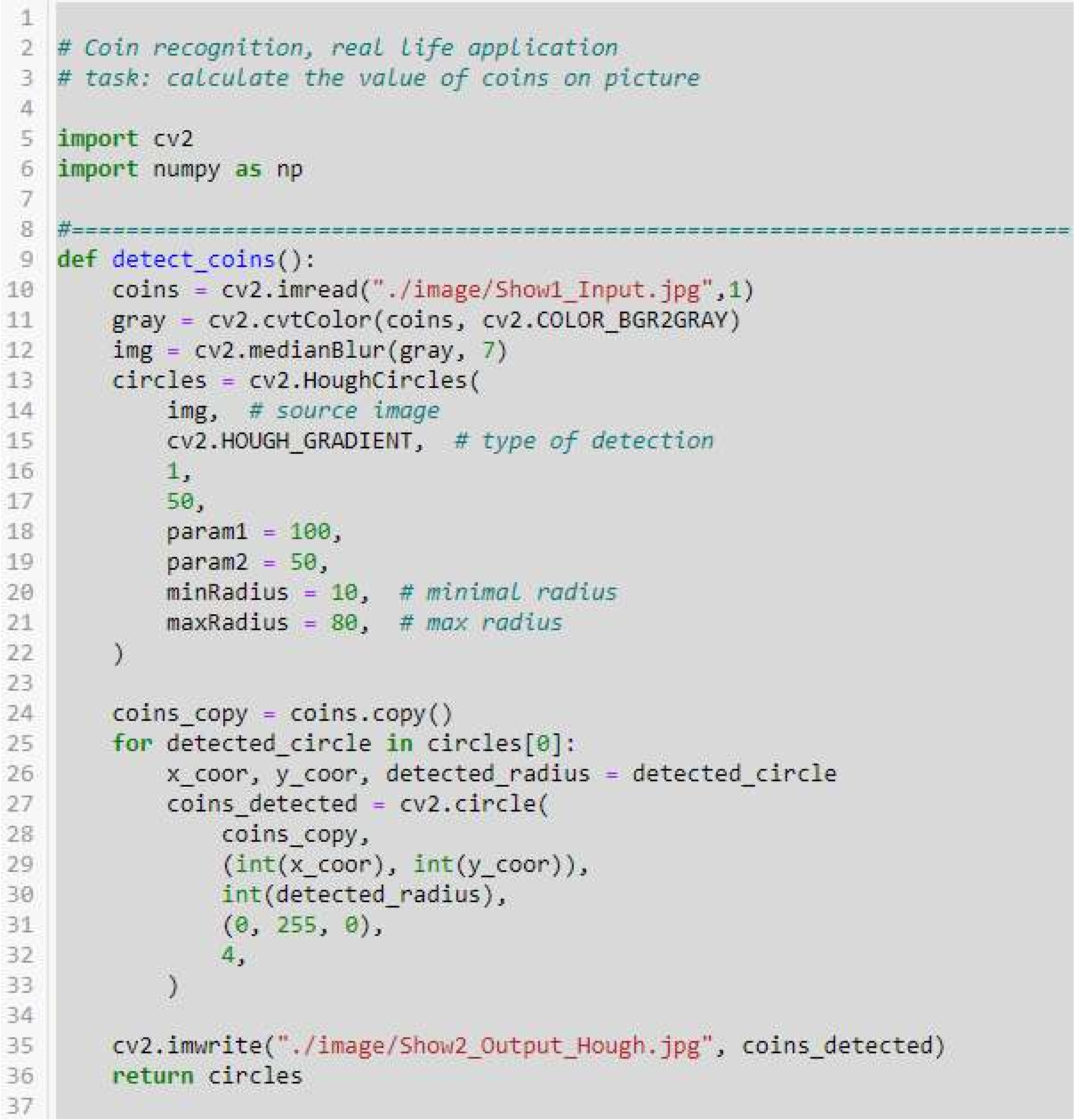
With stretching the gamma

Logically, by increasing the quality of input data, the quality (here it would be precision of circles drawn) would also increase. Consequently, making sure that the picture quality was high enough and stretching the gamma to bring out the contrast helped Hough to better detect the edges. To some degree, the previous improvements like black background helped as well. Then suddenly, few tweaks later, success!
Here is the final output:

This project again represents the accumulation of previous knowledge but is more valuable for me as it has a tangible real-life application that I could test at home. The biggest lesson learnt was to think about why the program is not performing well (bad preprocessing of image) rather than just trying to change various parameters of the Hough Circle Transformation and hope for the best. As always, may the Python be with you.

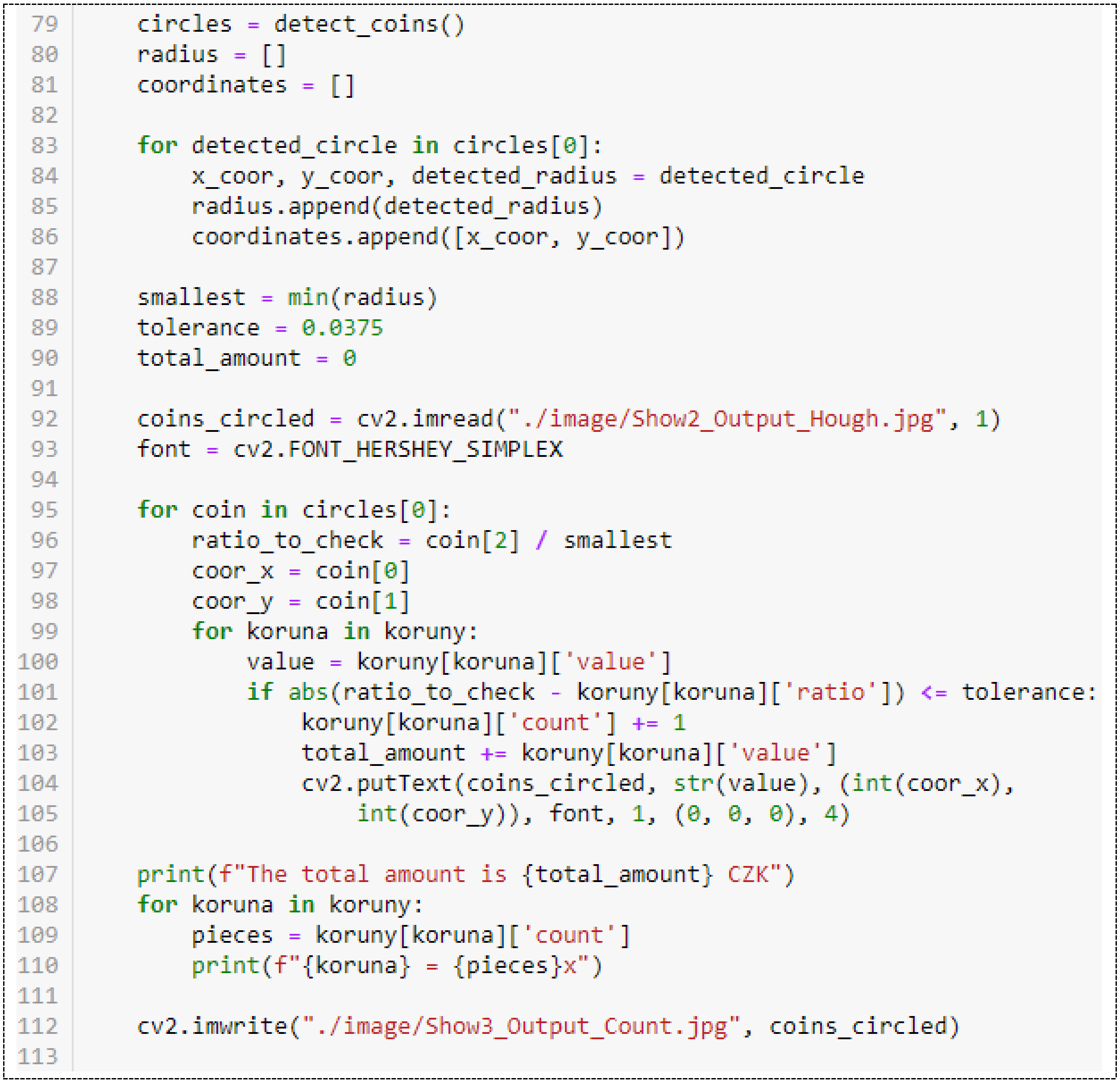
Test Code → Prog_L405B1

Test03: Prog_L405B1_Coil Segmentation

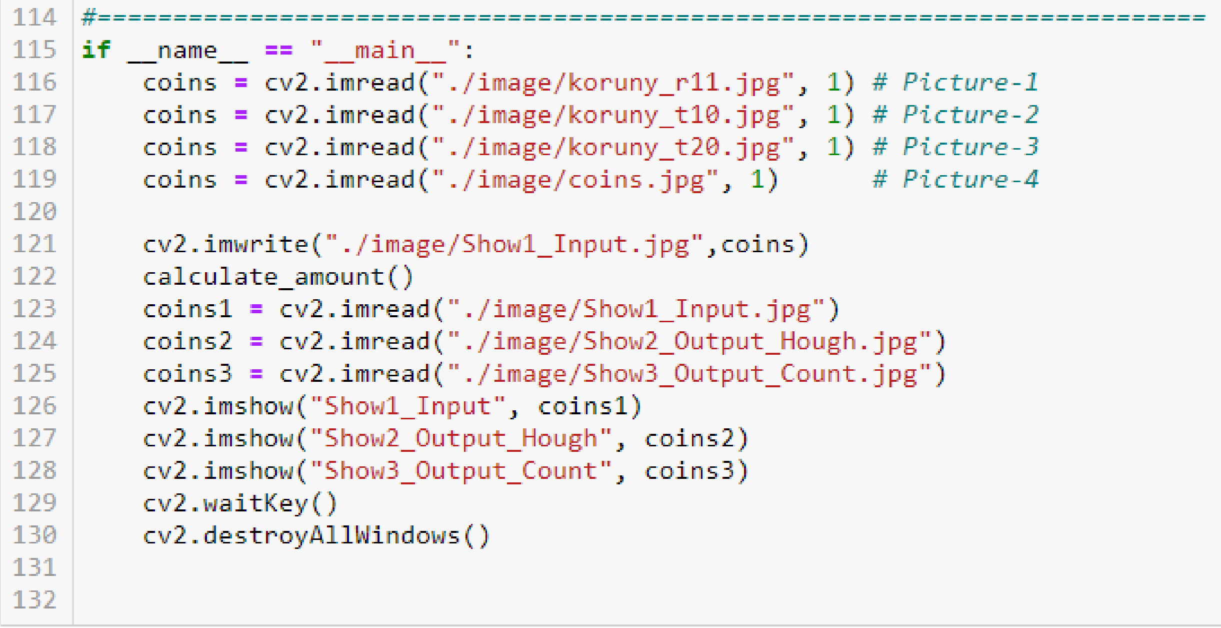


```

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

import cv2
import numpy as np

#==========================================================================
def detect_coins():
    coins = cv2.imread("./image/Show1_Input.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/Show2_Output_Hough.jpg", coins_detected)
    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,
        },
    }

```



```python
    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/Show2_Output_Hough.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/Show3_Output_Count.jpg", coins_circled)
```

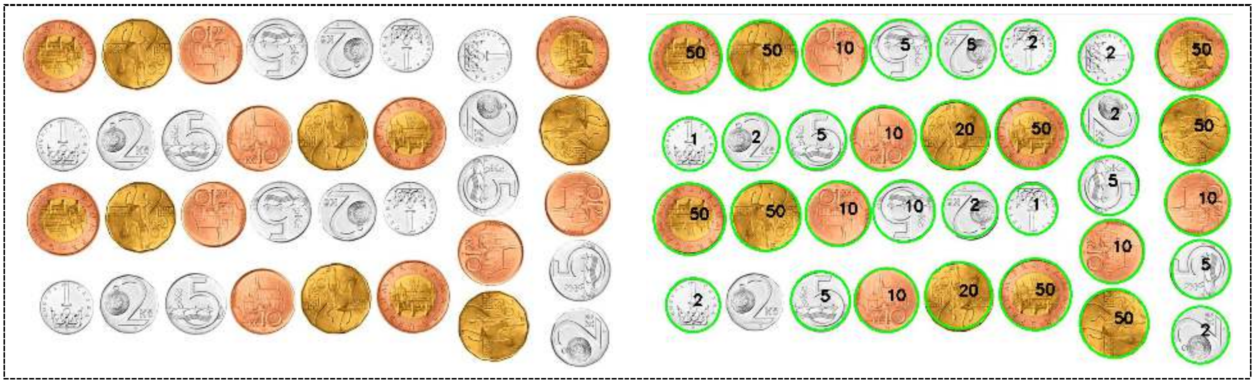


```
#==========================================================================
if __name__ == "__main__":
    coins = cv2.imread("./image/koruny_r11.jpg", 1) # Picture-1
    coins = cv2.imread("./image/koruny_t10.jpg", 1) # Picture-2
    coins = cv2.imread("./image/koruny_t20.jpg", 1) # Picture-3
 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

    cv2.imwrite("./image/Show1_Input.jpg",coins)
    calculate_amount()
    coins1 = cv2.imread("./image/Show1_Input.jpg")
    coins2 = cv2.imread("./image/Show2_Output_Hough.jpg")
    coins3 = cv2.imread("./image/Show3_Output_Count.jpg")
    cv2.imshow("Show1_Input", coins1)
    cv2.imshow("Show2_Output_Hough", coins2)
    cv2.imshow("Show3_Output_Count", coins3)
    cv2.waitKey()
    cv2.destroyAllWindows()


```

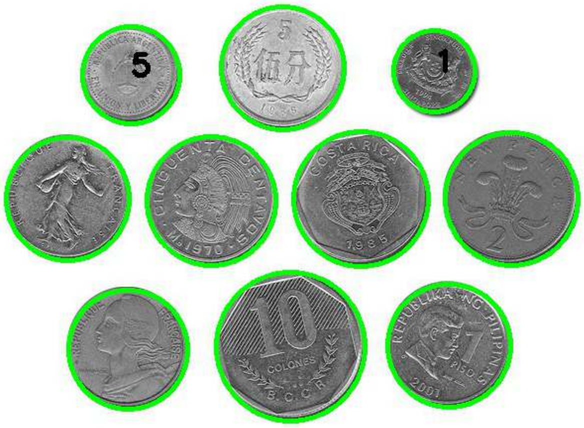
```
# Coin recognition, real life application
# task: calculate the value of coins on picture

import cv2
import numpy as np

#=========================================================================
def detect_coins():
    coins = cv2.imread("./image/Show1_Input.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/Show2_Output_Hough.jpg", coins_detected)
    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,
        },
    }

    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/Show2_Output_Hough.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/Show3_Output_Count.jpg", coins_circled)

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

    cv2.imwrite("./image/Show1_Input.jpg",coins)
    calculate_amount()
    coins1 = cv2.imread("./image/Show1_Input.jpg")
    coins2 = cv2.imread("./image/Show2_Output_Hough.jpg")
    coins3 = cv2.imread("./image/Show3_Output_Count.jpg")
    cv2.imshow("Show1_Input", coins1)
    cv2.imshow("Show2_Output_Hough", coins2)
    cv2.imshow("Show3_Output_Count", coins3)
    cv2.waitKey()
    cv2.destroyAllWindows()
```

คำถาม

- ทดสอบ Picture -1 { koruny_r11.jpg } < หากทำงานไม่ถูกต้องให้ปรับค่าตัวแปร พารามิเตอร์ >
- ทดสอบ Picture -2 { koruny_t10.jpg } < หากทำงานไม่ถูกต้องให้ปรับค่าตัวแปร พารามิเตอร์ >
- ทดสอบ Picture -3 { koruny_t20.jpg } แล้วใช้ paint ในการสร้างภาพที่มีเหรียญมากกว่า 24 เหรียญ < หากทำงานไม่ถูกต้องให้ปรับค่าตัวแปร พารามิเตอร์ >



- ทดสอบ Picture -4 {coins.jpg} หากต้องปรับให้ทำงานให้ถูกต้อง ต้องแก้ไขอะไรบ้าง



Gamma correction {Add to Project}

https://dev.to/tinazhouhui/discovering-opencv-with-python-gamma-correction-3cnh

Another practical exercise that I had quite some fun with was gamma correction. This concept is mainly used when we want to adjust the brightness of an image. We could also use it to restore the faded pictures to their previous depth of colour. Since I am just a Python Padawan, we will be demonstrating this on grayscale pictures but I promise, the concept works on coloured images as well.

In this short article, I will focus on the restoration of faded pictures.

A bit of (mathematical) background

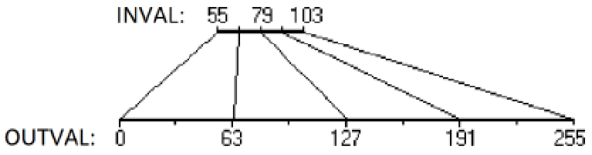
The logic behind is based on a concept called linear stretching. The faded picture simply means that the values of pixels are compressed to a smaller range and therefore not using the full range of values (in grayscale that would be from 0 to 255). For example, in the faded picture below, the values of the pixels range from 101 to 160. What linear stretching does, is that it re-scales the values to their full range from 0 to 255.

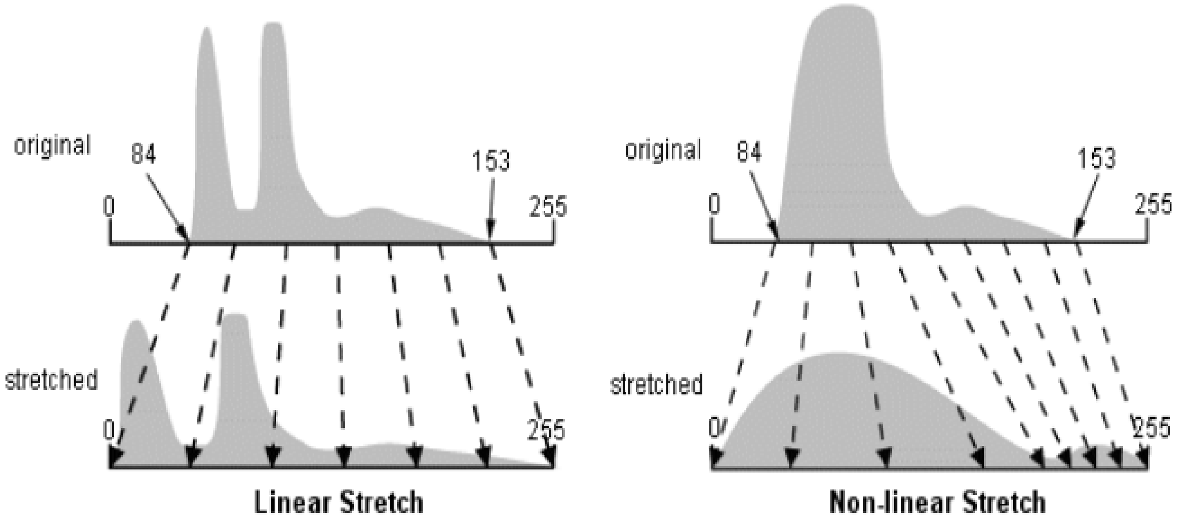
Here is the mathematical formula on how this can be achieved with each value:

$$\text{OUTVAL} = (\text{INVAL} - \text{INLO}) * ((\text{OUTUP} - \text{OUTLO})/(\text{INUP} - \text{INLO})) + \text{OUTLO}$$

where:

OUTVAL	Value of pixel in output map
INVAL	Value of pixel in input map
INLO	Lower value of 'stretch from' range
INUP	Upper value of 'stretch from' range
OUTLO	Lower value of 'stretch to' range
OUTUP	Upper value of 'stretch to' range

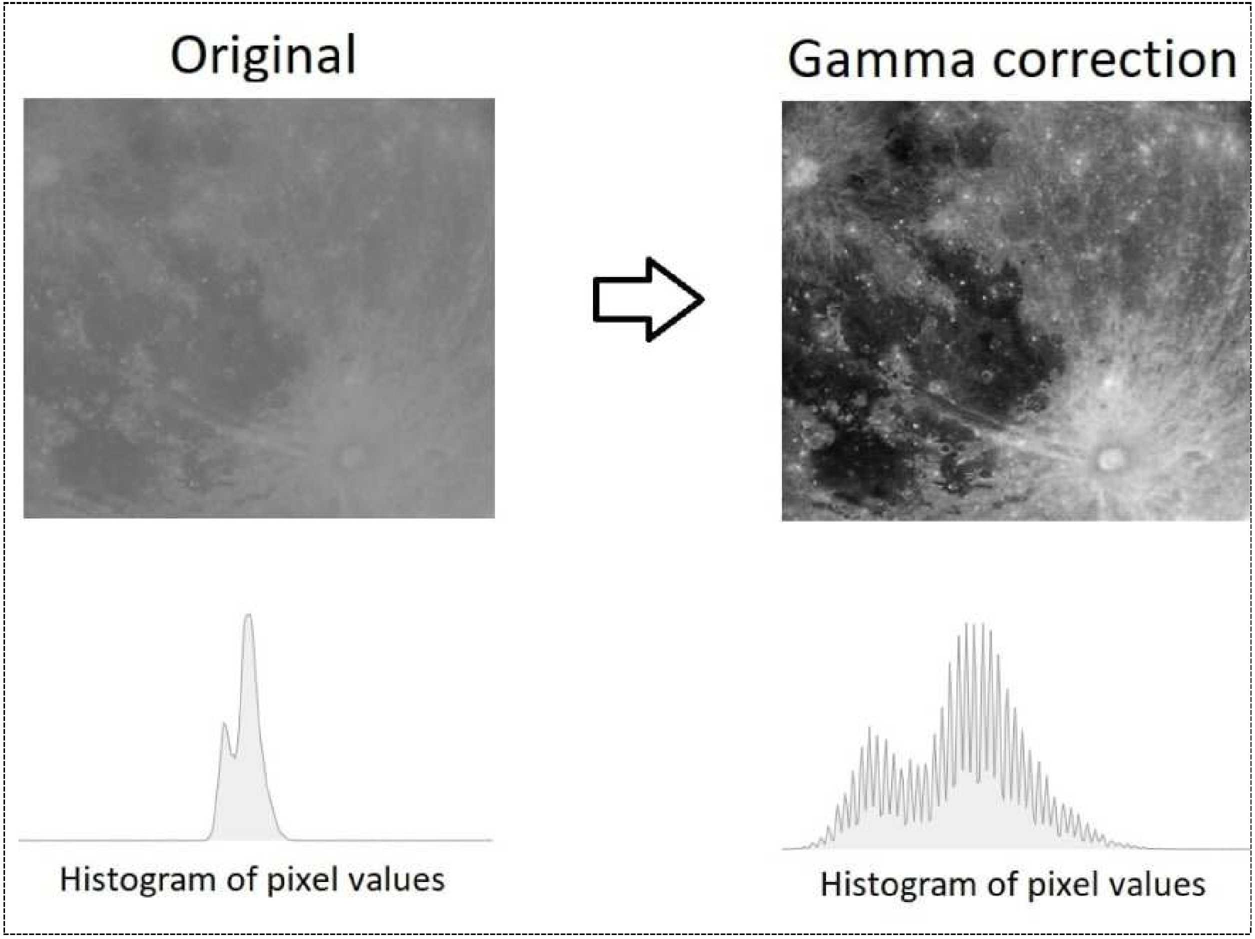




Python implementation

Just like in convolution, the necessary step is to cycle through every pixel and apply this mathematical formula to each of them. Be sure to check out my GitHub to see how it can be done.

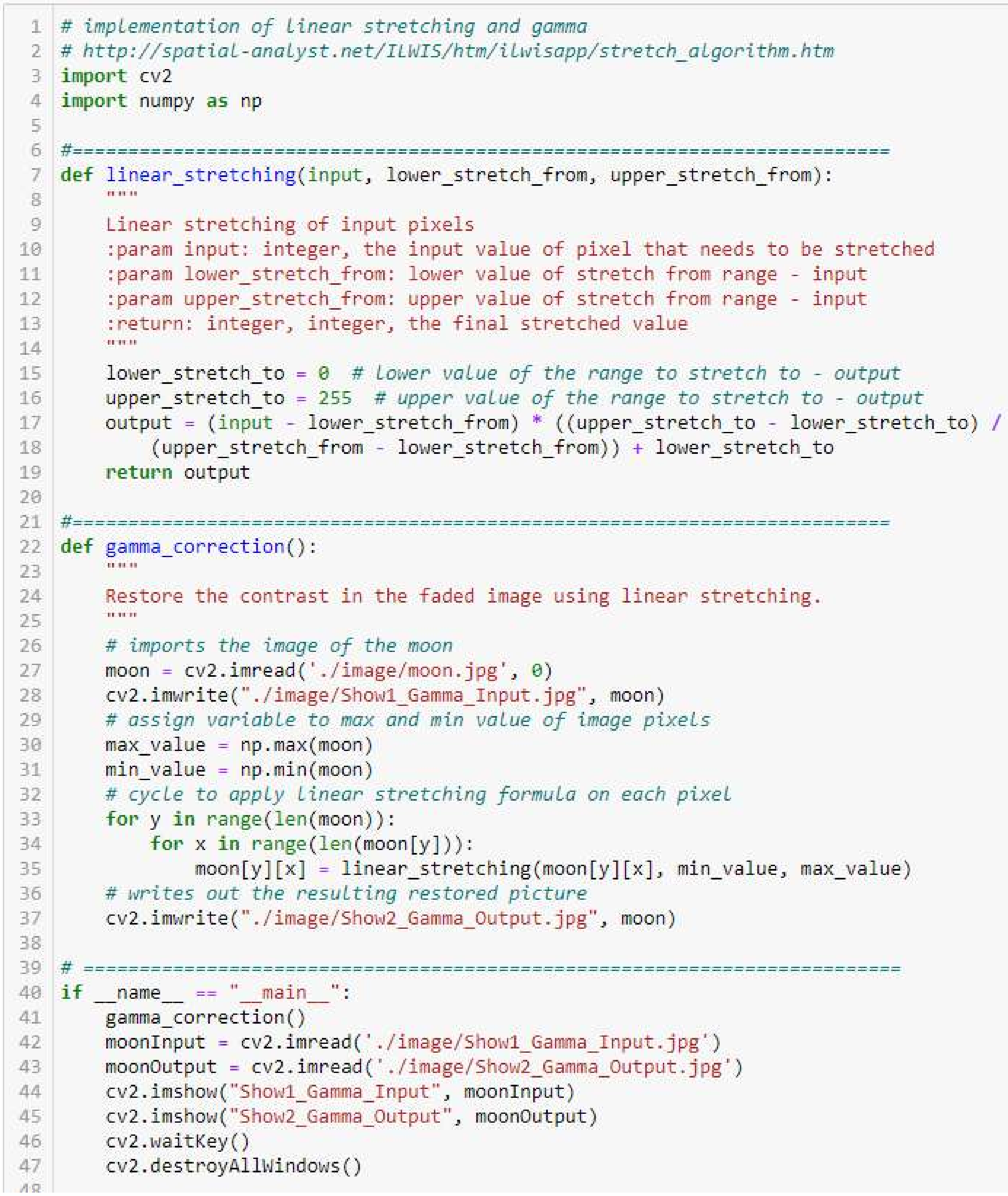
And voila, below is the result, look closely at how the histogram of pixel values stretched out from the original narrow range:



Tool used to create histogram

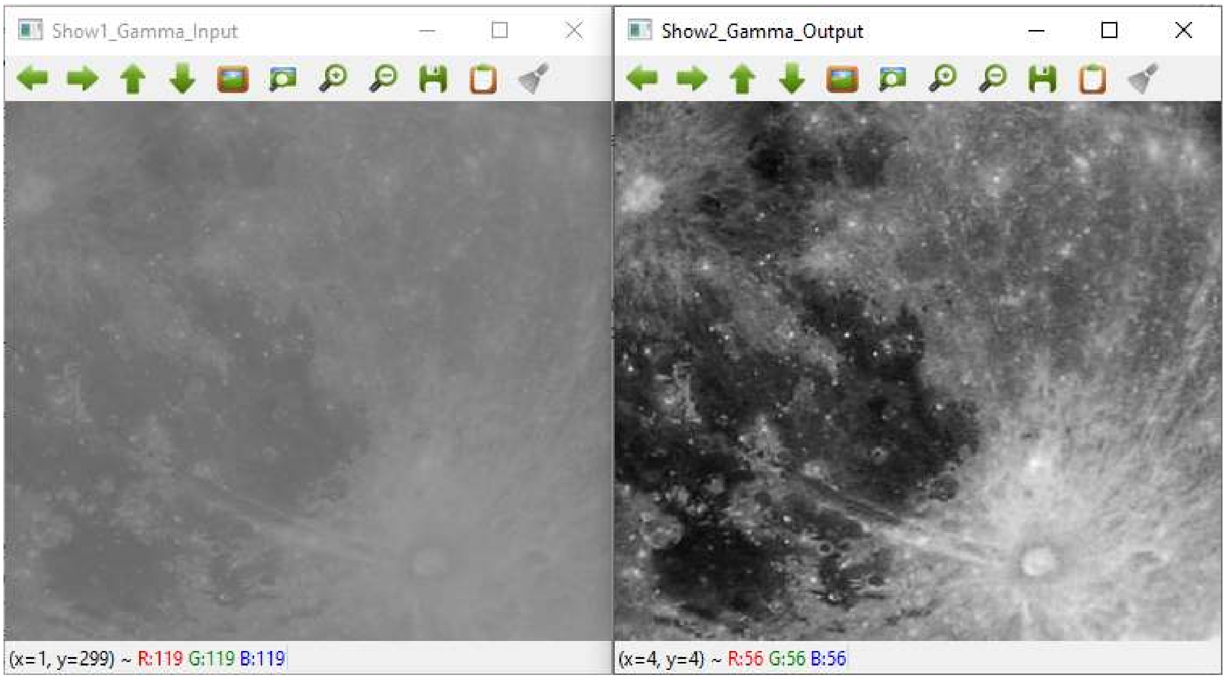
Hope you enjoyed the post, I recommend having a go at this and playing around with different images. Again, would appreciate if you let me know your thoughts. May the Python be with you.

Test04: Prog_L405B2_Gamma correction



```python
# implementation of linear stretching and gamma
# http://spatial-analyst.net/ILWIS/htm/ilwisapp/stretch_algorithm.htm
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)
    cv2.imwrite("./image/Show1_Gamma_Input.jpg", moon)
    # 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
    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)

# ========================================================================
if __name__ == "__main__":
    gamma_correction()
    moonInput = cv2.imread('./image/Show1_Gamma_Input.jpg')
    moonOutput = cv2.imread('./image/Show2_Gamma_Output.jpg')
    cv2.imshow("Show1_Gamma_Input", moonInput)
    cv2.imshow("Show2_Gamma_Output", moonOutput)
    cv2.waitKey()
    cv2.destroyAllWindows()
```

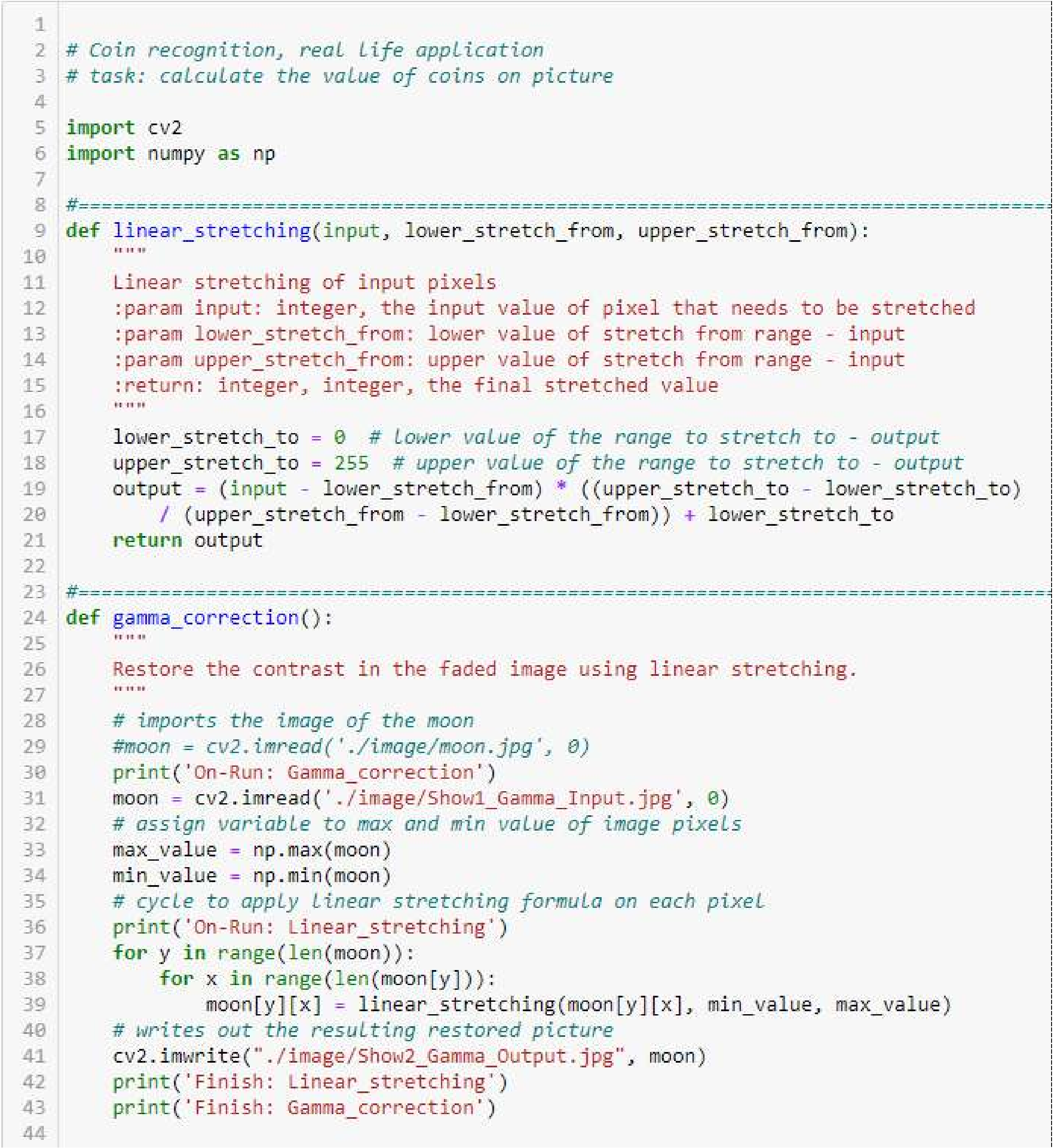
```
# implementation of linear stretching and gamma
# http://spatial-analyst.net/ILWIS/htm/ilwisapp/stretch_algorithm.htm

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)
    cv2.imwrite("./image/Show1_Gamma_Input.jpg", moon)
    # 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
    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)

# =========================================================================
if __name__ == "__main__":
    gamma_correction()
    moonInput = cv2.imread('./image/Show1_Gamma_Input.jpg')
    moonOutput = cv2.imread('./image/Show2_Gamma_Output.jpg')
    cv2.imshow("Show1_Gamma_Input", moonInput)
    cv2.imshow("Show2_Gamma_Output", moonOutput)
    cv2.waitKey()
    cv2.destroyAllWindows()
```



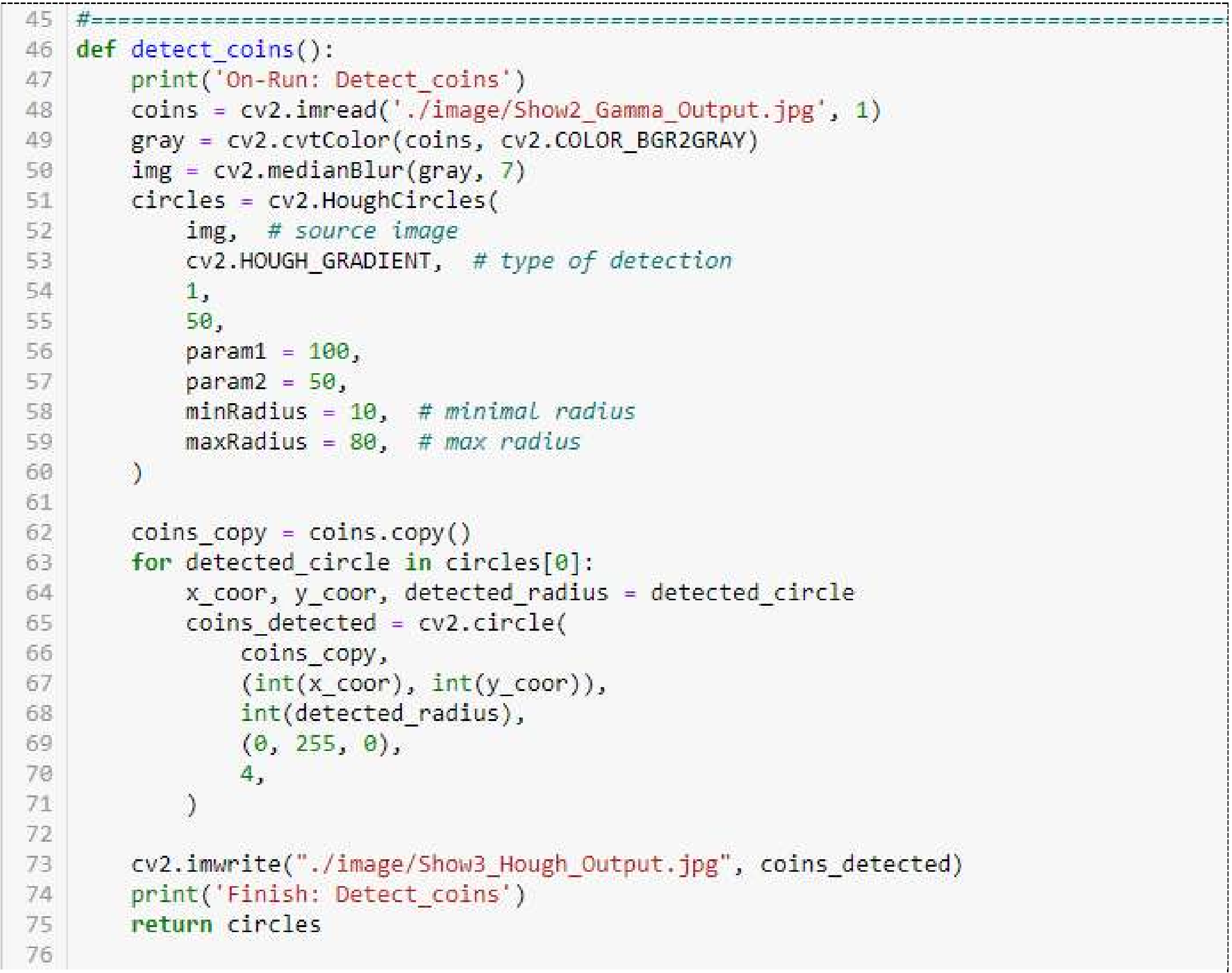
(x=1, y=299) ~ R:119 G:119 B:119

(x=4, y=4) ~ R:56 G:56 B:56

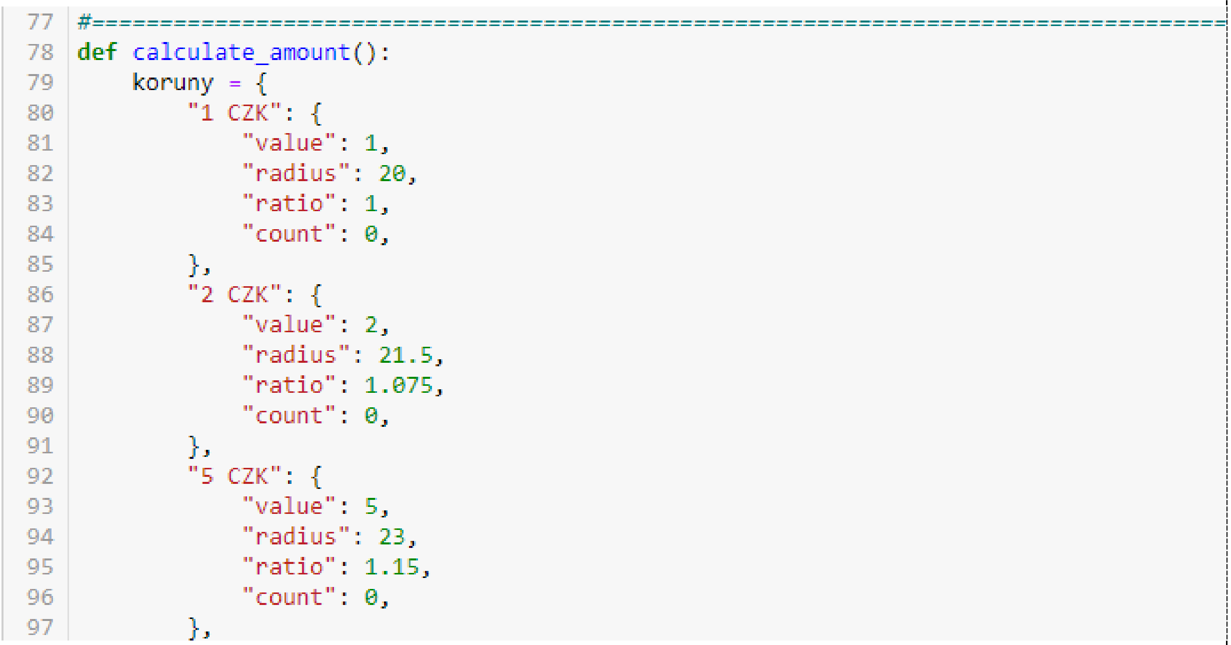
Test05: Prog_L4O5B3_ Coin Amount Calculation with Gamma Stretching



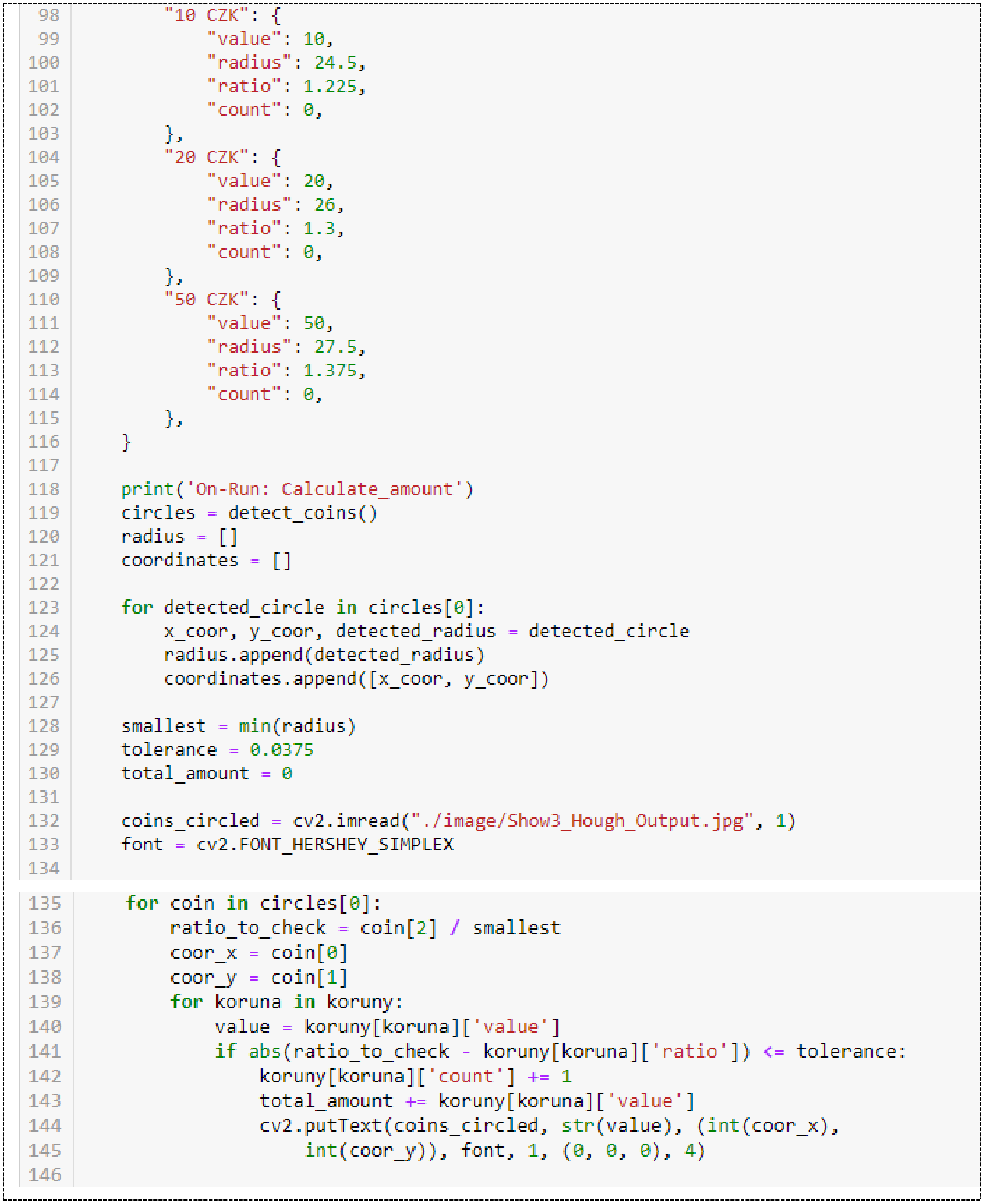
```

# 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')

```



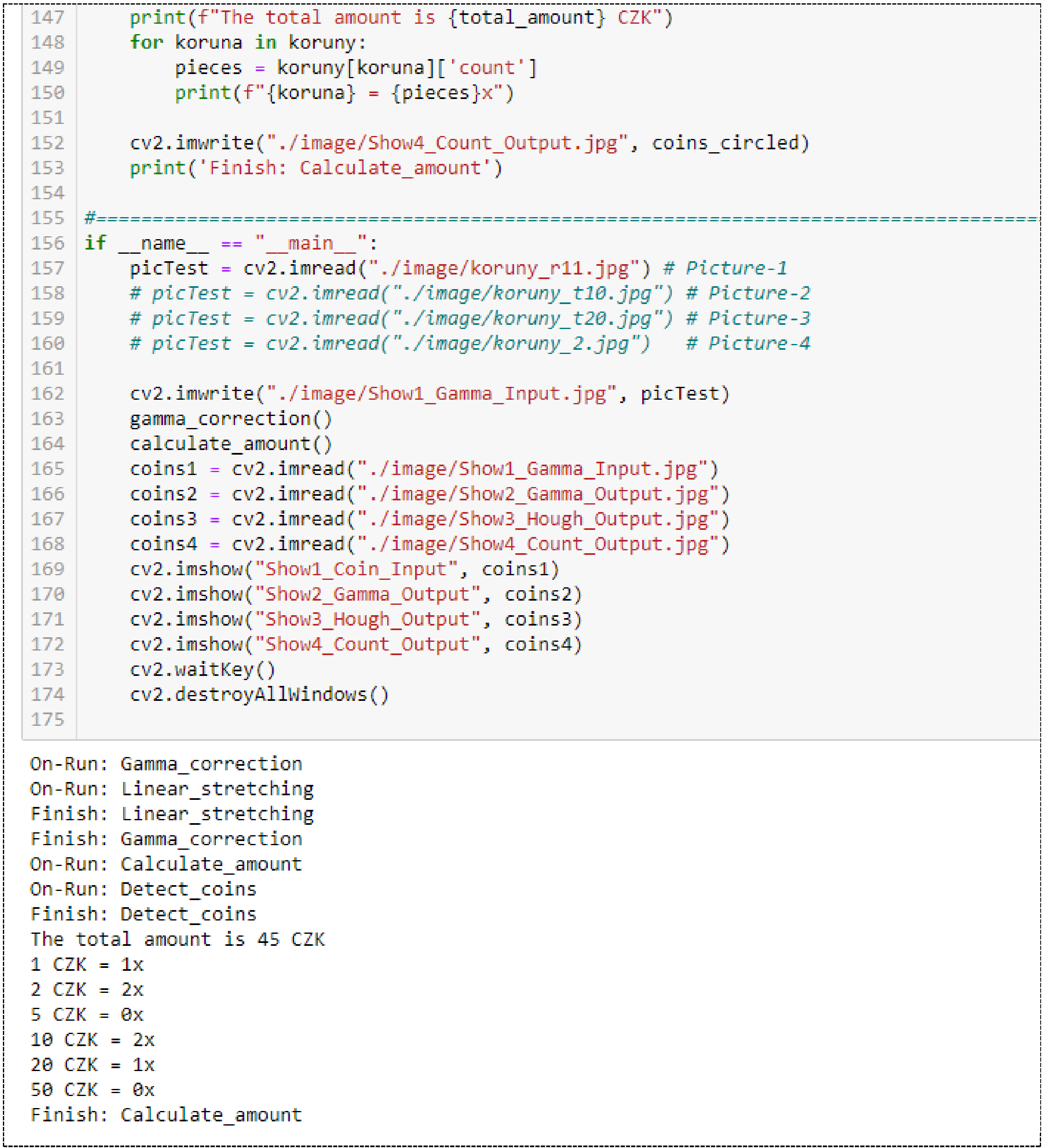
```python
#==========================================================================================
def detect_coins():
    print('On-Run: Detect_coins')
    coins = cv2.imread('./image/Show2_Gamma_Output.jpg', 1)
    gray = cv2.cvtColor(coins, cv2.COLOR_BGR2GRAY)
    img = cv2.medianBlur(gray, 7)
    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

```



```python
#==========================================================================================
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,
        },
```



```python
        "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)

```



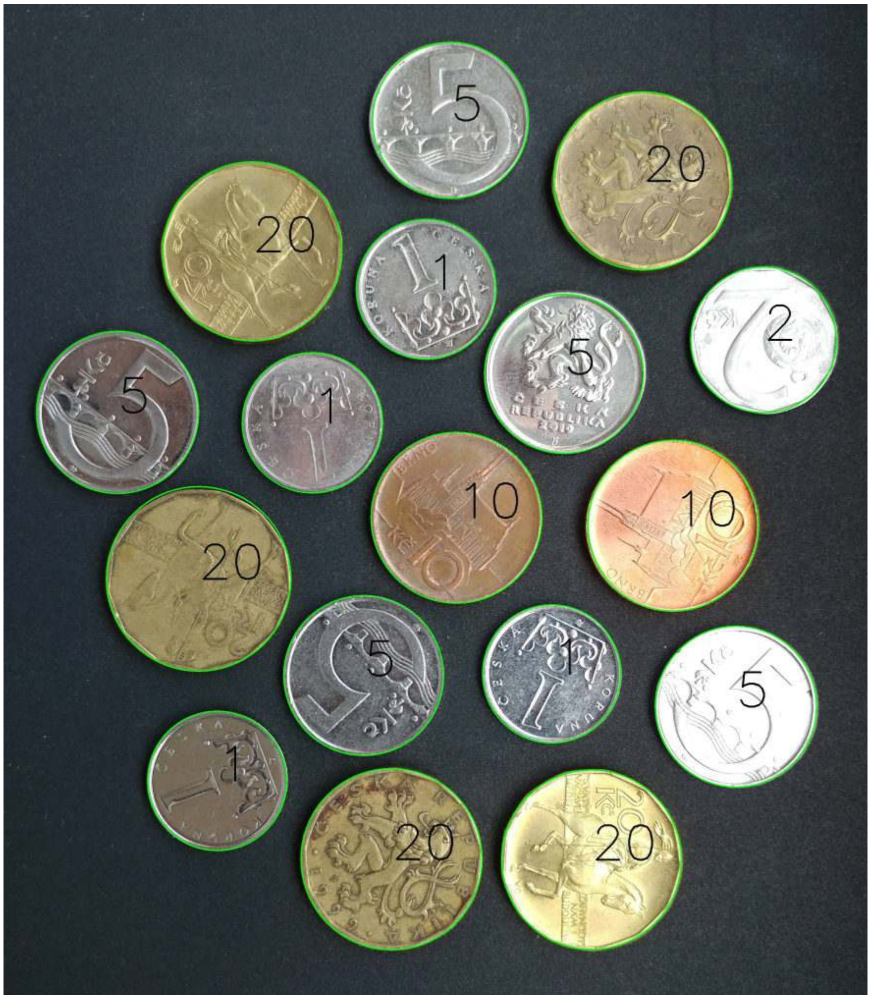
```
    print(f"The total amount is {total_amount} CZK")
    for koruna in koruny:
        pieces = koruny[koruna]['count']
        print(f"{koruna} = {pieces}x")

    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()
```

```
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 45 CZK
1 CZK = 1x
2 CZK = 2x
5 CZK = 0x
10 CZK = 2x
20 CZK = 1x
50 CZK = 0x
Finish: Calculate_amount
```

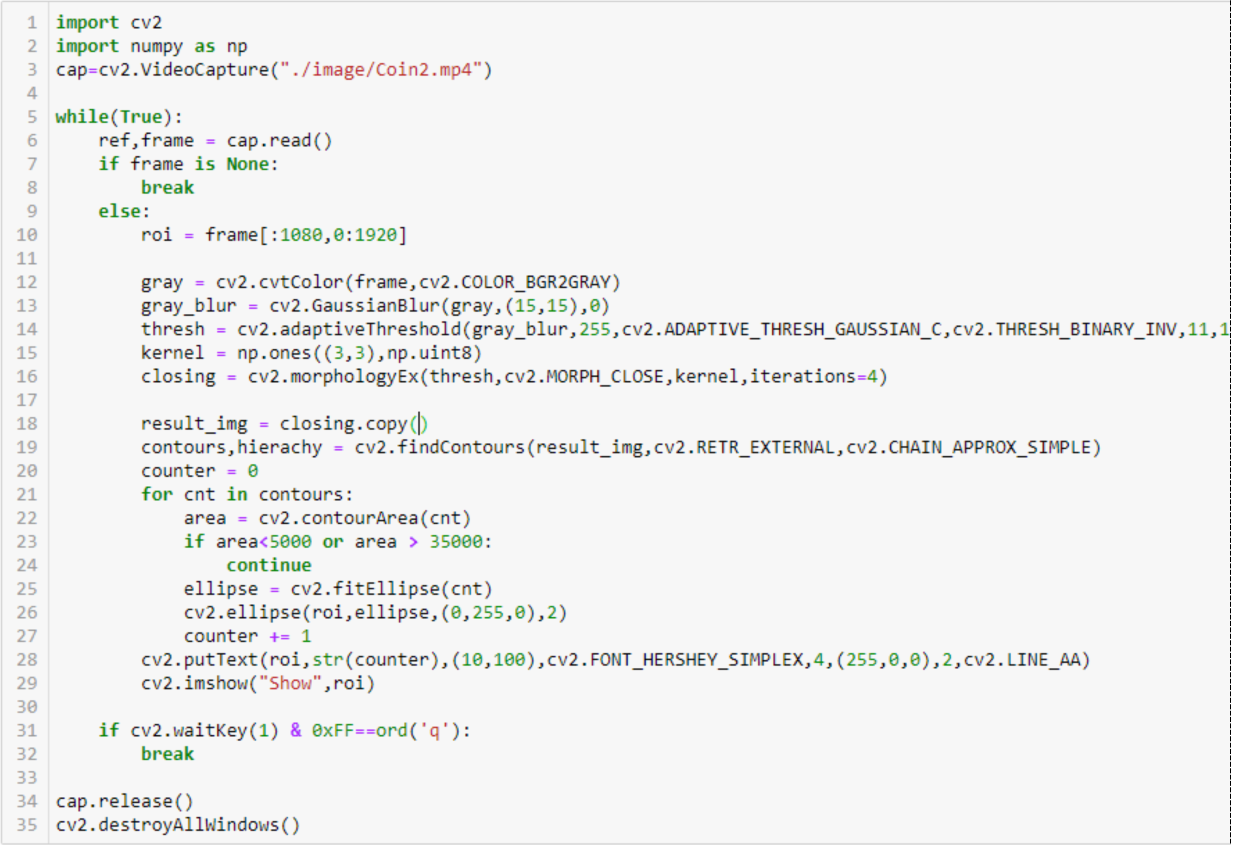
```
# Coin recognition, real life application
# task: calculate the value of coins on picture

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": 1.075,
            "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()
```

คำถาม

- ทดสอบ Picture -1 { koruny_r11.jpg } < หากทำงานไม่ถูกต้องให้ปรับค่าตัวแปร พารามิเตอร์ >
- ทดสอบ Picture -2 { koruny_t10.jpg } < หากทำงานไม่ถูกต้องให้ปรับค่าตัวแปร พารามิเตอร์ >
- ทดสอบ Picture -3 { koruny_t20.jpg } ใช้รูปเดิมที่มีเหรียญมากกว่า 24 เหรียญ < หากทำงานไม่ถูกต้องให้ปรับค่าตัวแปร พารามิเตอร์ >
- ทดสอบ Picture -4 { koruny_1.jpg หรือ koruny_2.jpg } ให้ทำงานให้ถูกต้อง

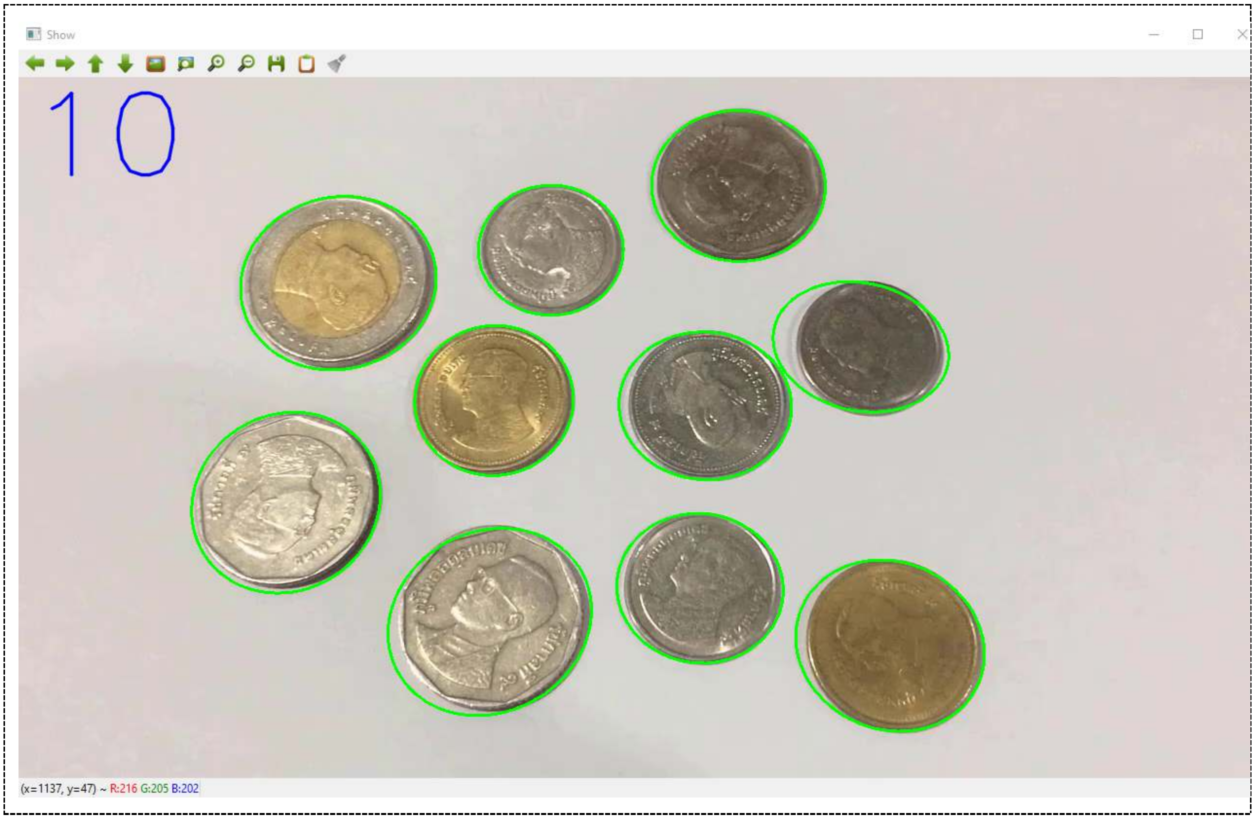

5
20
20
1
2
5
5
1
10
10
20
5
1
5
1
20
20

Test06: Prog_L405B4_Video Coin Amount Calculation



```
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()
```

```
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()
```



Show
10
(x=1137, y=47) ~ R:216 G:205 B:202

Lab405c: Car Counting

https://github.com/celalaygar/car-counting-with-python

Car counting process was made with Numpy, opencv on python.
if you wanna to watch the video as result of this project you can click and watch on youtube

- You can watch this video: as https://youtu.be/Oq7JGBhgvl4 result.
  For counting_car.py_1 project < https://github.com/celalaygar/car-counting-with-python/blob/master/counting_car_1.py >
- You can watch this video: as https://youtu.be/qm-Ha_ZrGrw result.
  For counting_car.py_2 project < https://github.com/celalaygar/car-counting-with-python/blob/master/counting_car_2.py >

requirements

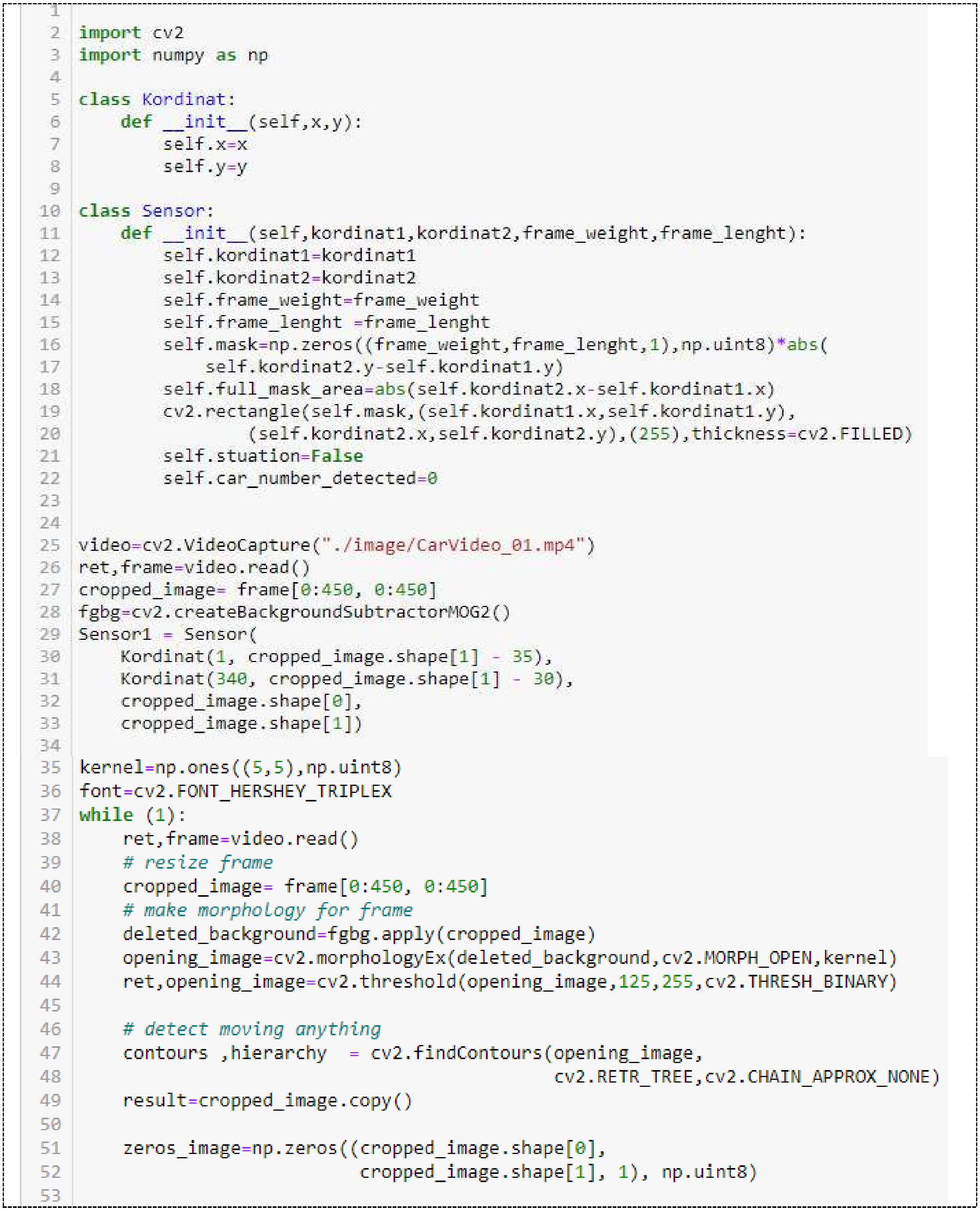
- pycharm
- python 3+
- numpy
- cv2 (opencv)

İmportant

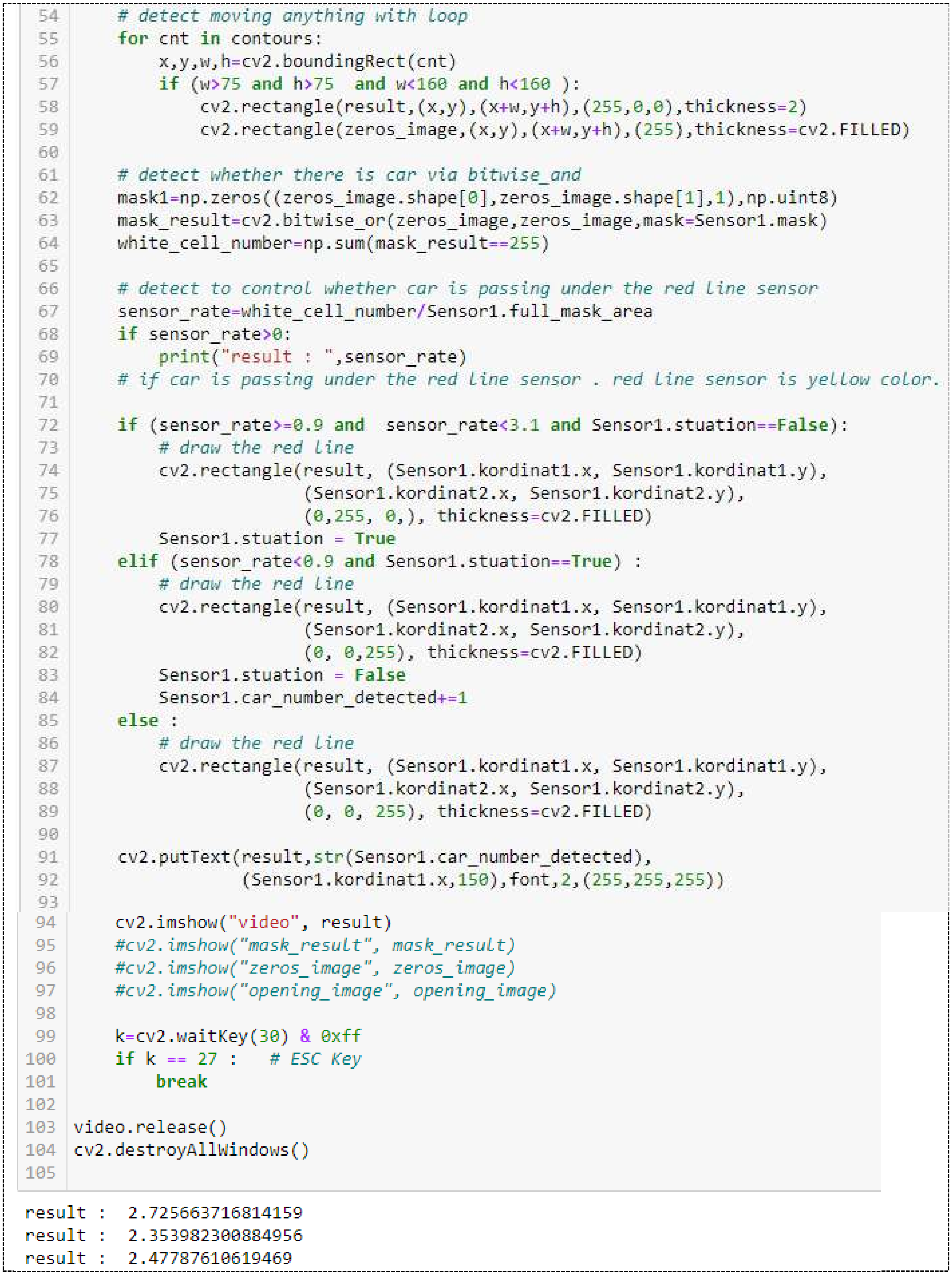
There is a few problems in these projects. These problems will be solved soon.

Test07: Prog_L405C1_ComeIn

```
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()
```

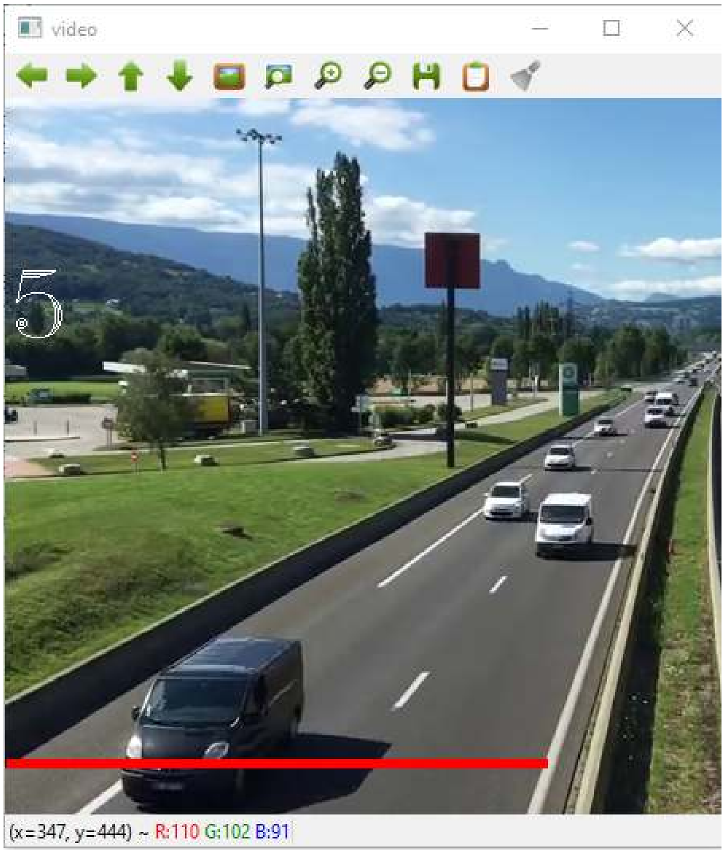


```python

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)

```



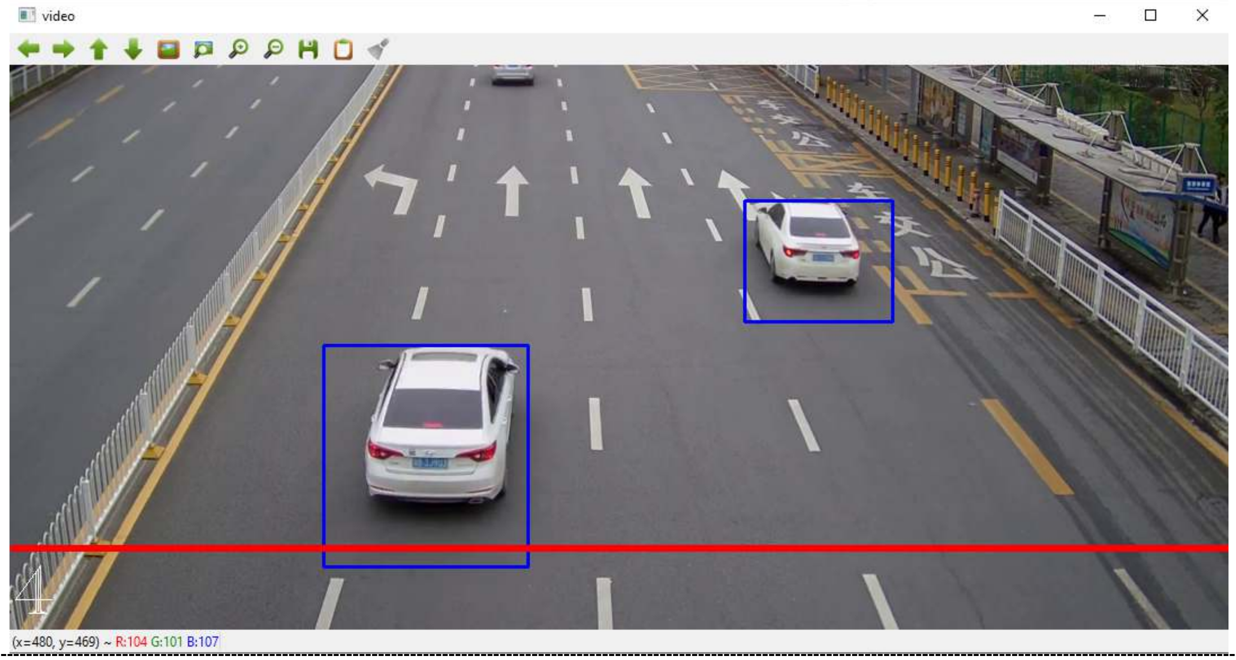
```
    # detect moving anything with loop
    for cnt in contours:
        x,y,w,h=cv2.boundingRect(cnt)
        if (w>75 and h>75  and w<160 and h<160 ):
            cv2.rectangle(result,(x,y),(x+w,y+h),(255,0,0),thickness=2)
            cv2.rectangle(zeros_image,(x,y),(x+w,y+h),(255),thickness=cv2.FILLED)

    # 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()
```

```
result :  2.725663716814159
result :  2.353982300884956
result :  2.47787610619469
```

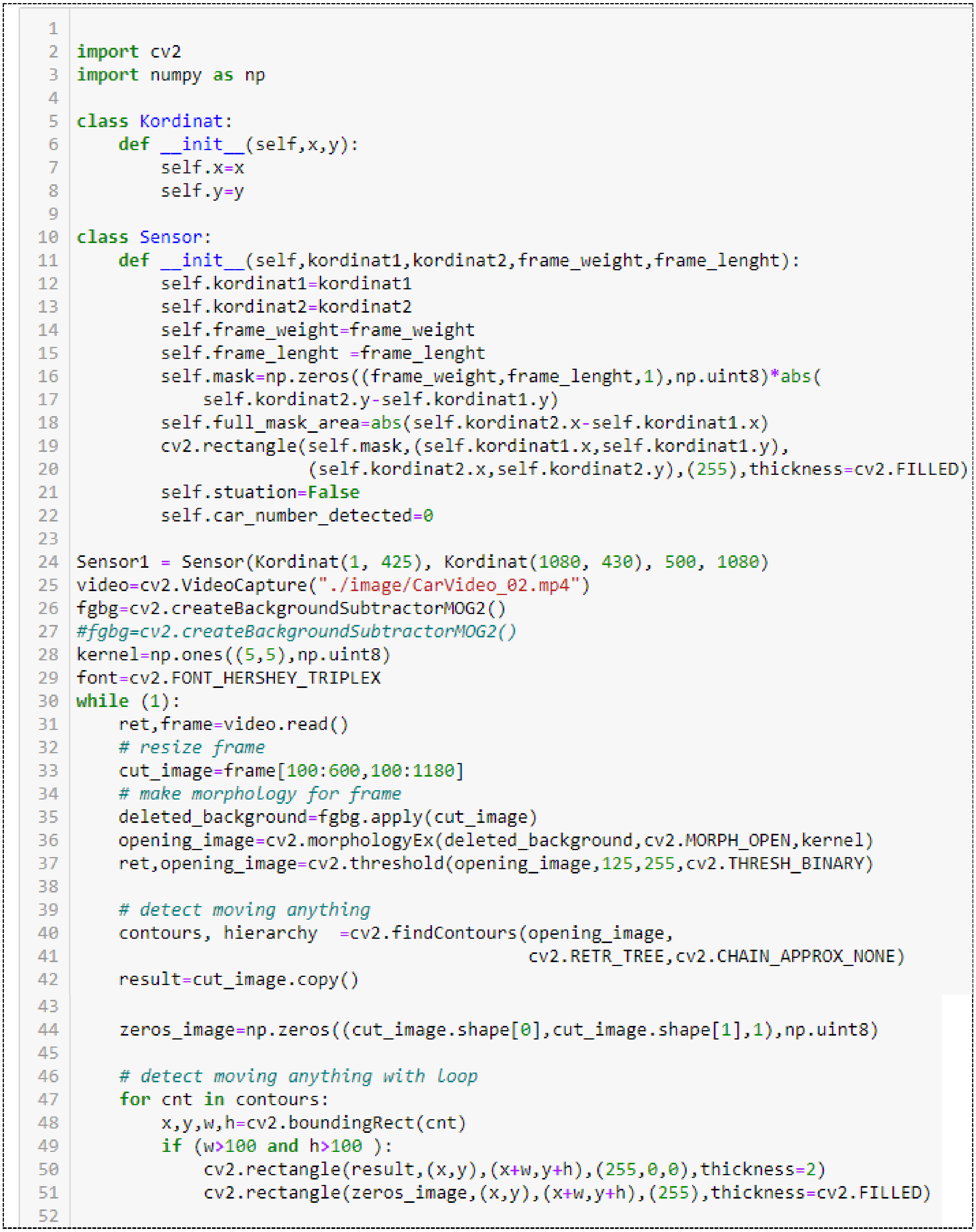

video
5
(x=347, y=444) ~ R:110 G:102 B:91

Test08: Prog_L405C2_GoOut

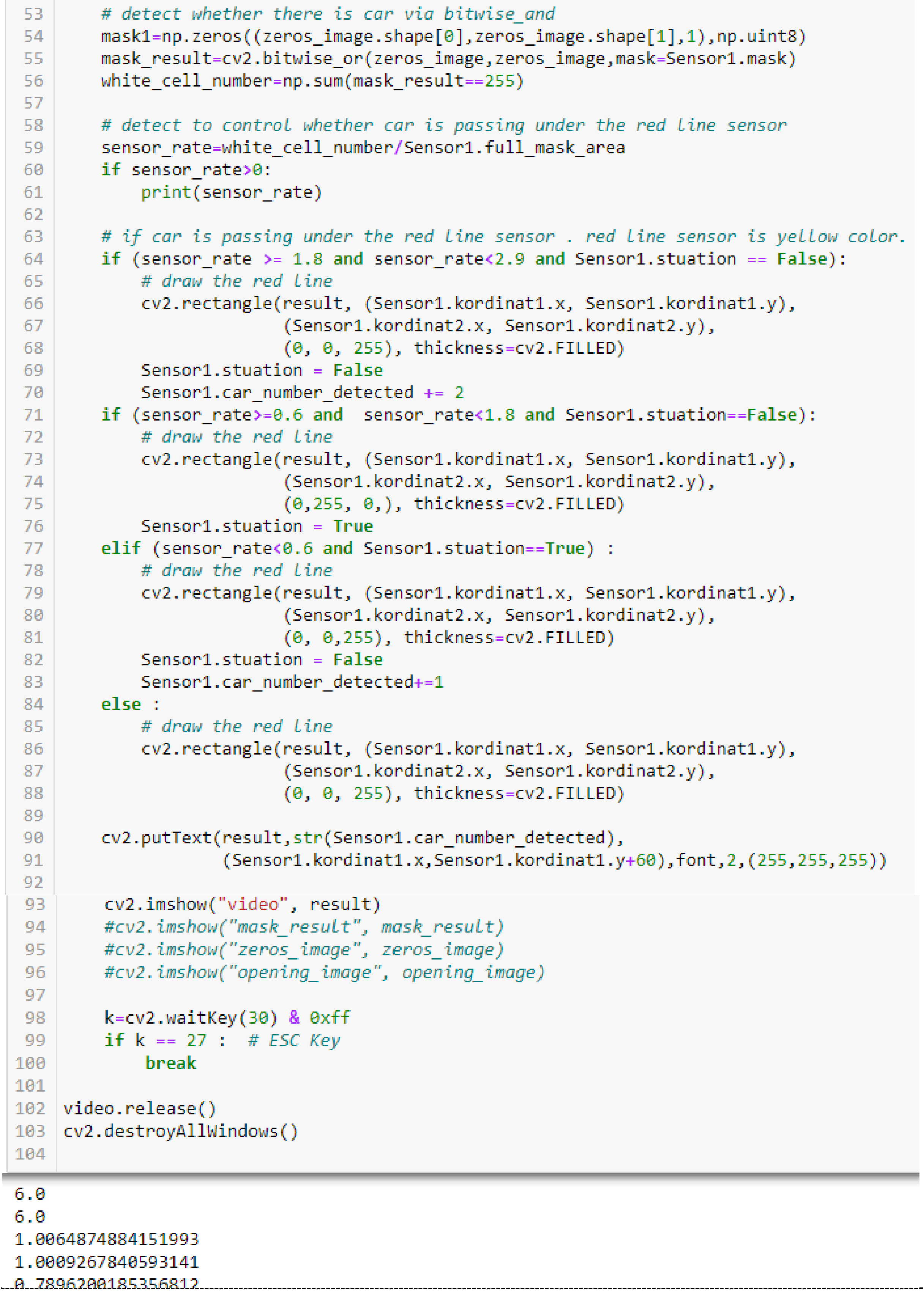
```
import cv2
import numpy as np

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()
```



(x=480, y=469) ~ R:104 G:101 B:107



```python

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()

```

```
6.0
6.0
1.0064874884151993
1.0009267840593141
0.7896200185356812
```

Lab405d: Object Detection and Tracking

https://pysource.com/2021/01/28/object-tracking-with-opencv-and-python/

First of all it must be clear that what the difference between object detection and object tracking is:

- **Object detection** is the detection on every single frame and frame after frame.
- **Object tracking** does frame-by-frame tracking but keeps the history of where the object is at a time after time

We will talk first about object detection and then about how to apply object tracking to the detection.

What are the possible applications?

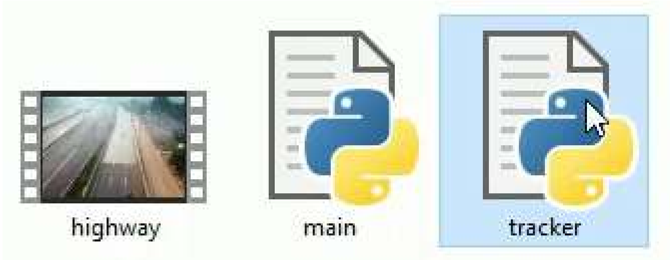
The possible applications are different for example, counting how many people are in a certain area, checking how many objects pass on a conveyor belt, or counting the vehicles on a highway.

Surely where having seen the tutorial you will easily think of thousands of ideas applied to real-life or potentially to industry. Certainly, if you need to design a tri-section of objects this is the tool you need.

What do we need?

In this tutorial we will use 3 files:

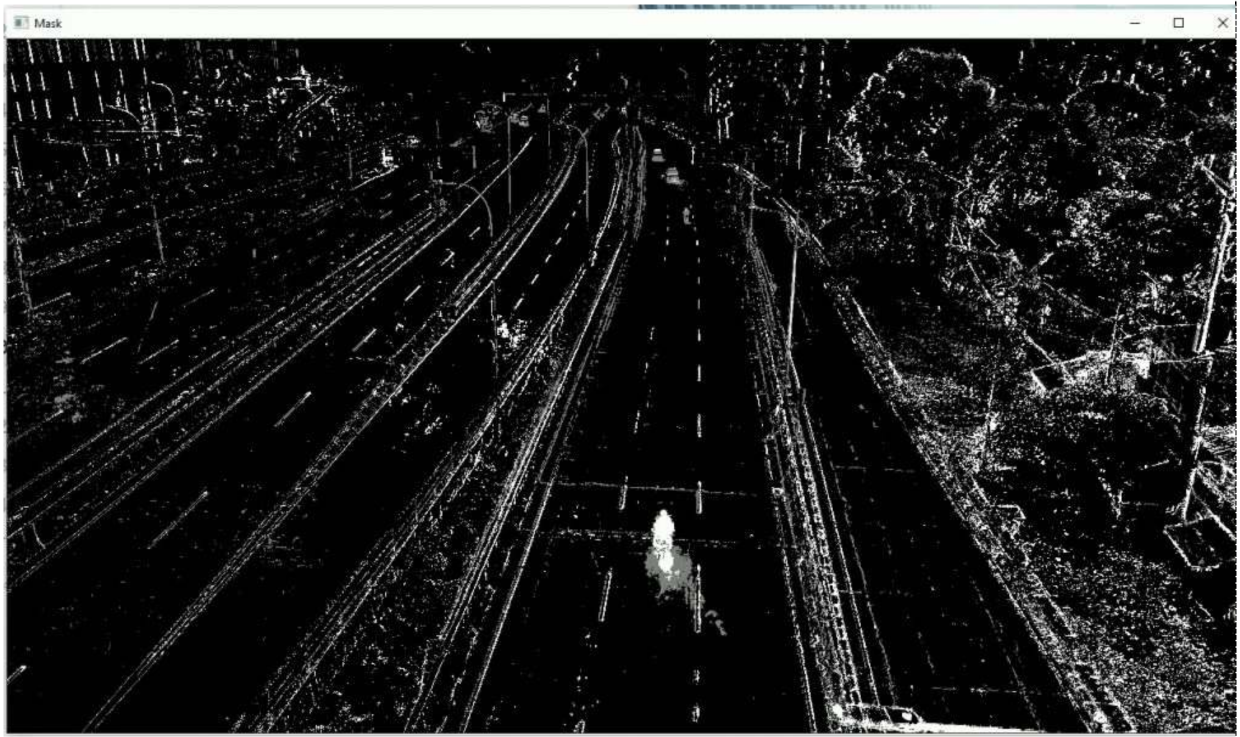
1. The video of the highway we will use to count the vehicles
2. Tracker files. This has already been written and you can simply download it
3. Main file. Write me in real-time and we will proceed step by step with the integration of the libraries



Object detection – Step-by-Step

1. First we need to call the highway.mp4 file and create a mask

```
cap = cv2.VideoCapture("highway.mp4")
# Object detection from Stable camera
object_detector = cv2.createBackgroundSubtractorMOG2()
while True:
            ret, frame = cap.read()
            # 1. Object Detection
            mask = object_detector.apply(frame)
```

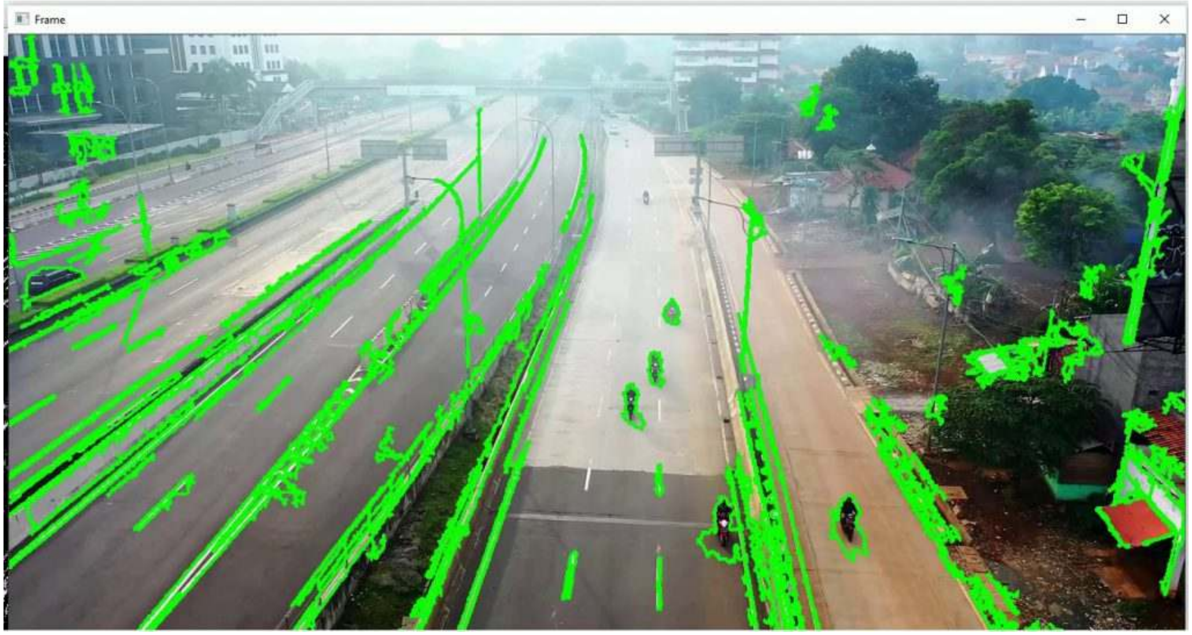


As you can see in the example code we also used the createBackgroundSubtractorMOG2 function which Returns the "background ratio" parameter of the algorithm and then create the mask.

As you can see, however, there is a lot of noise in the image. So let's improve the extraction by removing all the smaller elements and focus our attention on objects that are larger than a certain area.

2. Drawing the contours with OpenCV's cv2.drawContours function we obtain this result. You won't need to use this function, consider it as a debug of a first result

```
    _, mask = cv2.threshold(mask, 254, 255, cv2.THRESH_BINARY)
    contours, _ = cv2.findContours(mask, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
    for cnt in contours:
        # Calculate area and remove small elements
        area = cv2.contourArea(cnt)
        if area > 100:
        #Show image
```

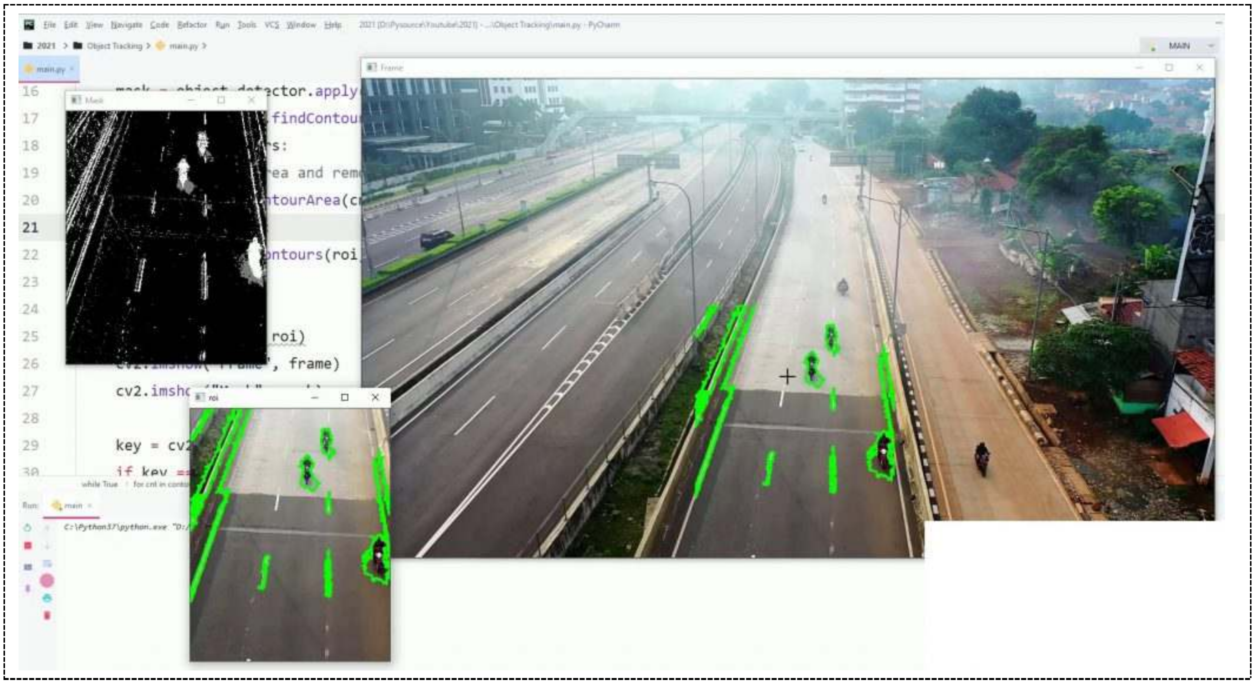


3. We define a Region of interest

For the purpose of this tutorial, it is not important to analyze the entire window. We are only interested in counting all the vehicles that pass at a certain point, for this reason, we must define a region of interest ROI and apply the mask only in this area.

Already in the image you can see a good first result.

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

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

    # 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)

    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)
```



```
mask = object_detector.apply(
                    findContou
                    s:
                    ea and rem
                    ntourArea(c

                    ntours(roi

                    roi)
cv2.imshow("Frame", frame)
cv2.imsh

key = cv2
if key ==
```

The function cv2.createBackgroundSubtractorMOG2 was added at the beginning without defining parameters, now let's see how to further improve our result. history is the first parameter, in this case, it is set to 100 because the camera is fixed. var Threshold instead is 40 because the lower the value the greater the possibility of making false positives. In this case, we are only interested in the larger objects.

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

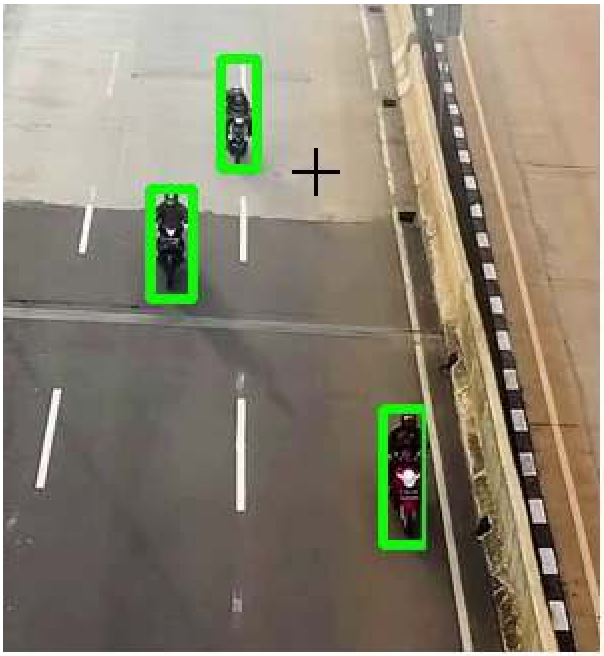
4. Draw the box around the object

Before proceeding with the rectangle, we do a further cleaning of the image. To do this, the threshold function comes in handy. Starting from our mask we tell it that we want to show only the white or black values so by writing "254, 255" only the values between 254 and 255 will be considered.

```
_, mask = cv2.threshold(mask, 254, 255, cv2.THRESH_BINARY)
```

We then insert the coordinates of the found object into the if condition and draw the rectangle and this is the final result

```
x, y, w, h = cv2.boundingRect(cnt)
cv2.rectangle(roi, (x, y), (x + w, y + h), (0, 255, 0), 3)
```



As you can see, we have everything you need to proceed with object tracking.

5. Object Tracking

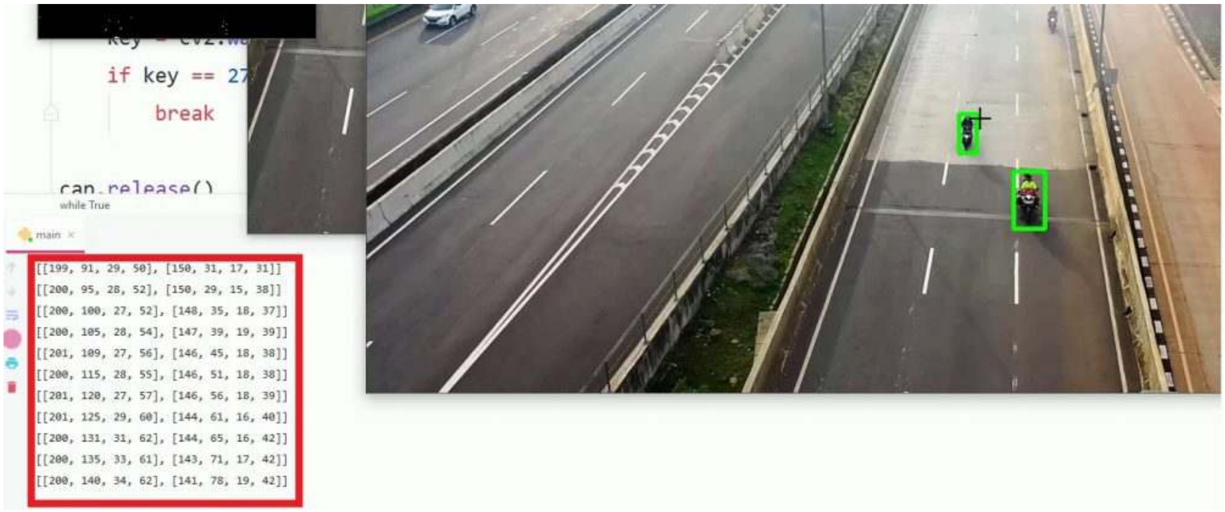
We now simply have to import and integrate the tracking functions.

```
from tracker import *
# Create tracker object
tracker = EuclideanDistTracker()
```

Once the object has been created, we must therefore take each position of the bounding box and insert them in a single array.

```
detections.append([x, y, w, h])
```

By showing the result on the screen you can see how all the lanes that pass through our ROI are identified and their positions inserted in a specific array. Obviously, the more motorcycles identified the larger our array will be



[redacted]

```
key = cv2.wa
    if key == 27
        break

cap.release()
```

```
[[199, 91, 29, 50], [150, 31, 17, 31]]
[[200, 95, 28, 52], [150, 29, 15, 38]]
[[200, 100, 27, 52], [148, 35, 18, 37]]
[[200, 105, 28, 54], [147, 39, 19, 39]]
[[201, 109, 27, 56], [146, 45, 18, 38]]
[[200, 115, 28, 55], [146, 51, 18, 38]]
[[201, 120, 27, 57], [146, 56, 18, 39]]
[[201, 125, 29, 60], [144, 61, 16, 40]]
[[200, 131, 31, 62], [144, 65, 16, 42]]
[[200, 135, 33, 61], [143, 71, 17, 42]]
[[200, 140, 34, 62], [141, 78, 19, 42]]
```

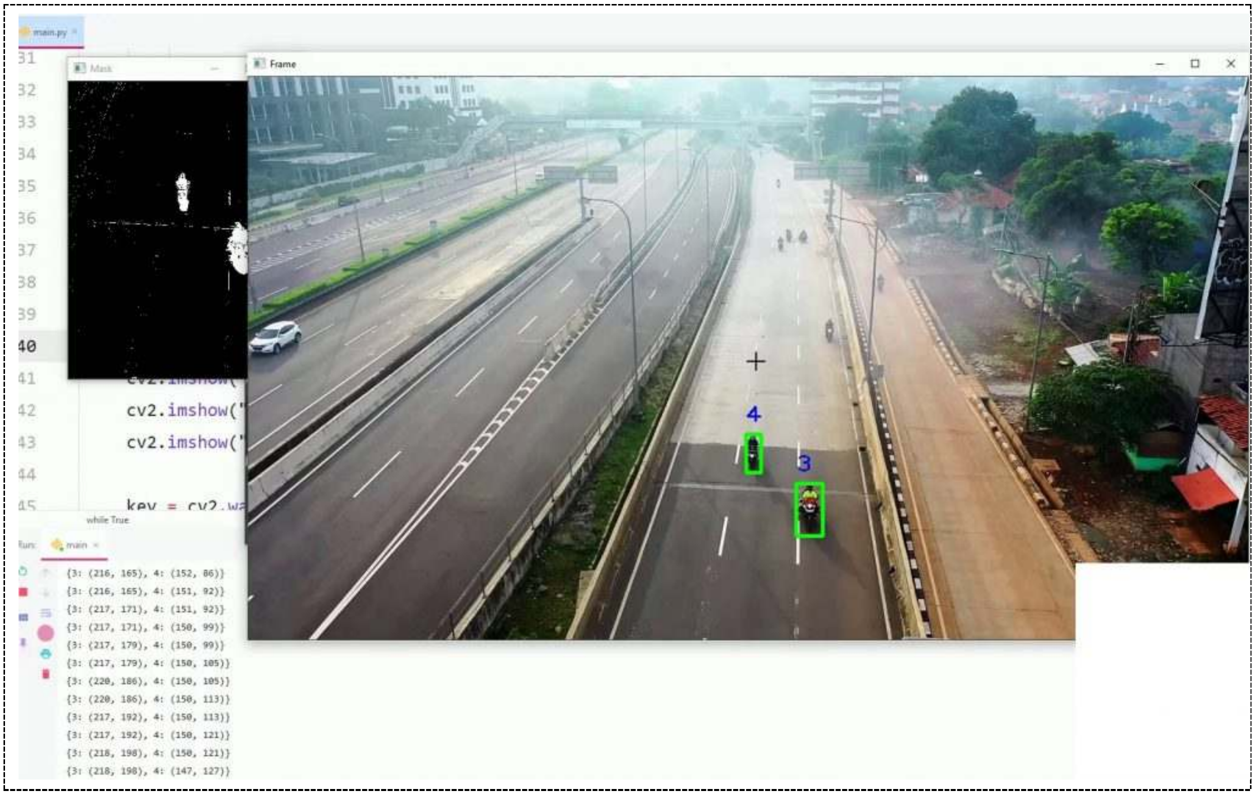
6. Associate unique ID to the object

Let’s now pass our array with positions to tracker.update(). We will again get an array with the potions but in addition, a unique id will be assigned for each object.

As you can see from the code we can analyze everything with a for a loop. At this point we just have to draw the rectangle and show the vehicle ID.

In the image you can see the result

```
# 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(
    cv2.imshow("
    cv2.imshow("

    key = cv2.wa
```

```
{3: (216, 165), 4: (152, 86)}
{3: (216, 165), 4: (151, 92)}
{3: (217, 171), 4: (151, 92)}
{3: (217, 171), 4: (150, 99)}
{3: (217, 179), 4: (150, 99)}
{3: (217, 179), 4: (150, 105)}
{3: (220, 186), 4: (150, 105)}
{3: (220, 186), 4: (150, 113)}
{3: (217, 192), 4: (150, 113)}
{3: (217, 192), 4: (150, 121)}
{3: (218, 198), 4: (150, 121)}
{3: (218, 198), 4: (147, 127)}
```

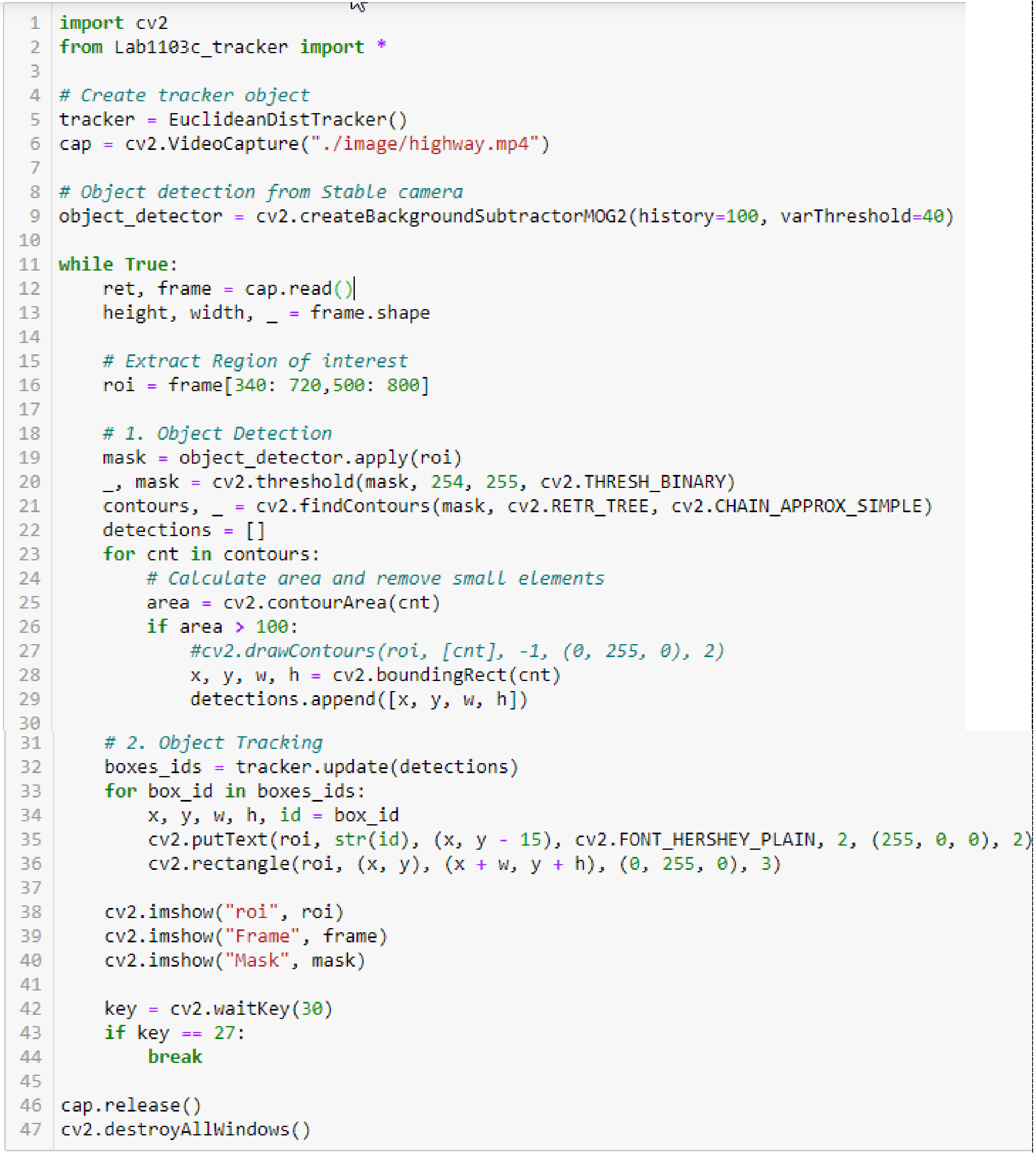
Conclusions

As you can also see from the video we have obtained the result that we set ourselves at the beginning of this tutorial.

However, you must consider it as an exercise or starting point because on this topic there is a lot to say and the aim of this tutorial was only to make you understand the principle of object tracking.

If you want to integrate Object Tracking into your project, you should use more reliable and advanced object detection methods, as well as tracking methods.

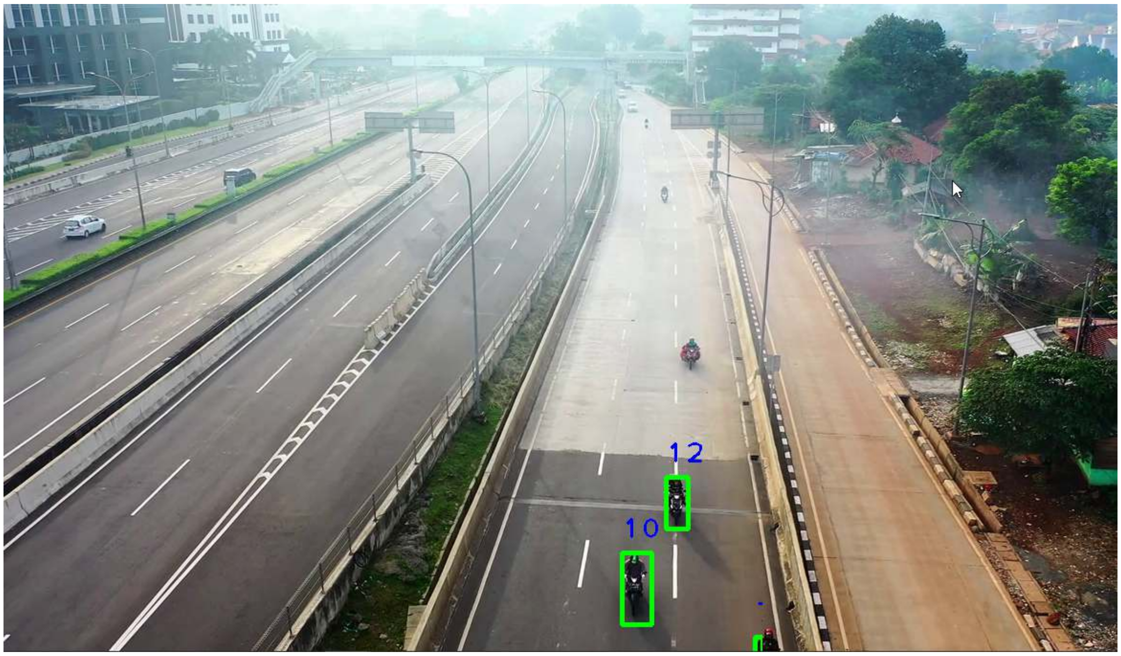
Test09 -- Prog_L405D1 -- Result_Bike



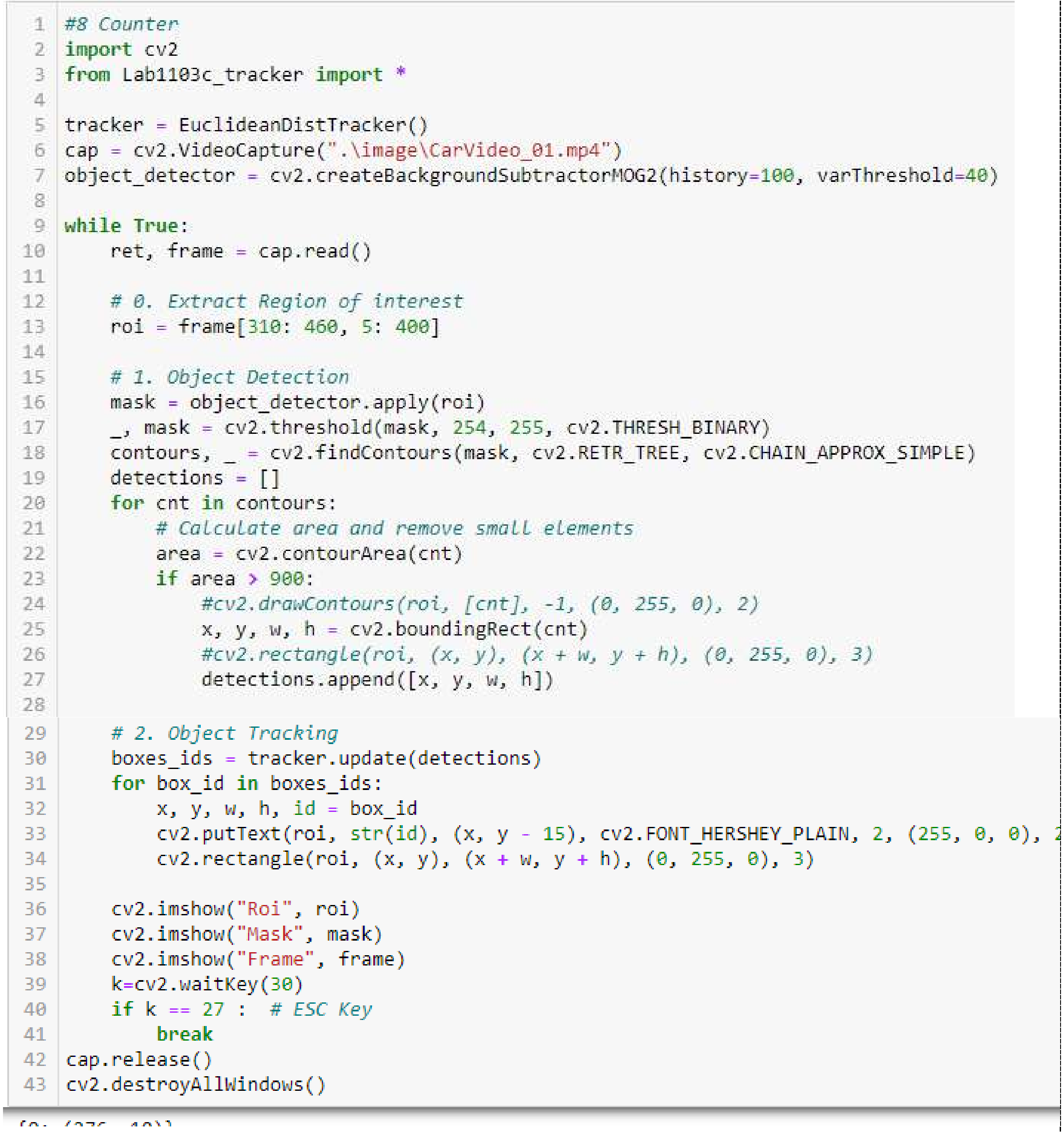
```python
import cv2
from Lab1103c_tracker import *

# 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()
```

```python
import cv2
from Lab1103c_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()
```



Test10 -- Prog_L405D1 -- Result_Car

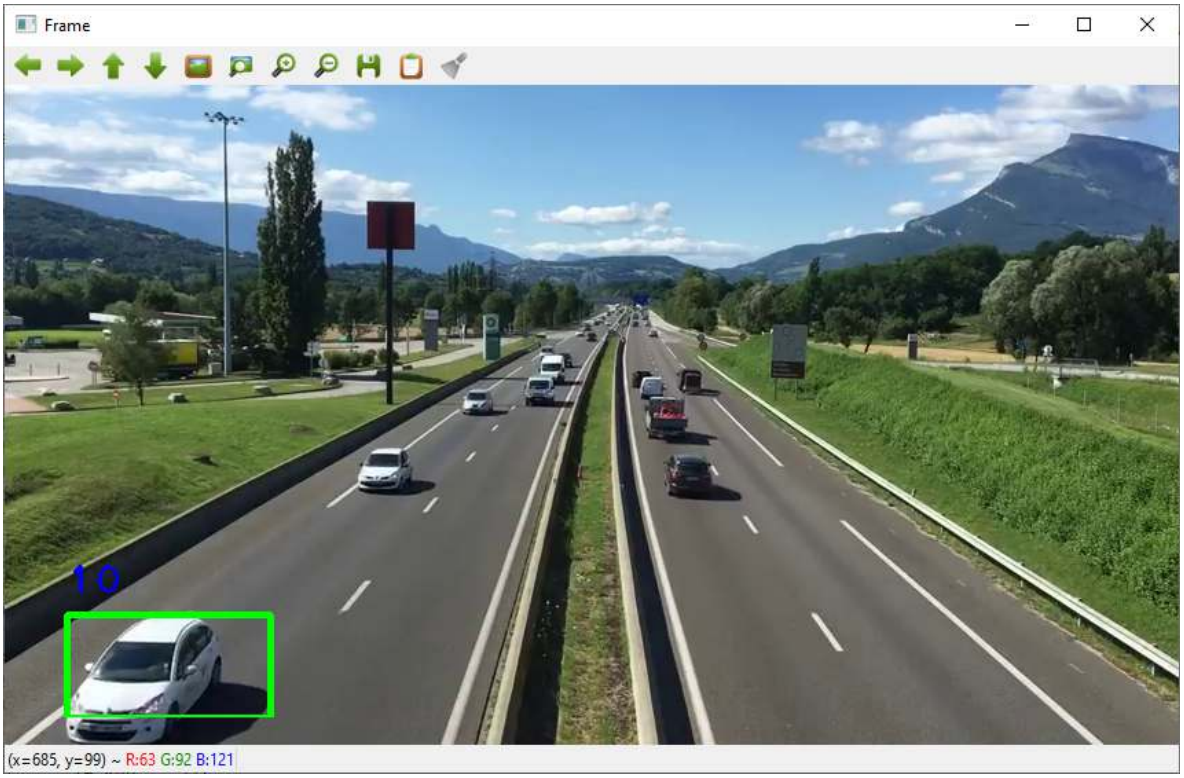


```
#8 Counter
import cv2
from Lab1103c_tracker import *

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()
```

```
{0: (276, 10)}
```

```
#8 Counter
import cv2
from Lab1103c_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()
```



(x=685, y=99) ~ R:63 G:92 B:121

กิจกรรมที่ 5/6 – Object Detection and Tracking

- Capture ผลการทำงานที่ได้ลองปฏิบัติทั้ง 4 กรณี A, B, C, D
- ลองใช้ตัวอย่างอื่นในการทดสอบ
- อภิปรายผลการทดสอบ แต่ละหัวข้อ (A, B, C, D)
- คำถามที่อยากถาม
- บอกแนวการใช้งาน กับงานที่รับผิดชอบ