การพัฒนาโปรแกรมประยุกต์และปัญญาประดิษฐ์ เพื่อการมองเห็นของเครื่องจักร
Computer Programing and Artificial Intelligence in Machine Vision

4/4 – Machine Learning + Case Study

7/8 -- Case Study 4 -- Visual Inspection

https://shodensha.co.th/products/auto-visual-inspector/ai-detector.html
https://pysource.com/2020/05/19/identify-objects-moving-on-a-conveyor-belt-using-opencv-with-python/
https://youtu.be/A29IqeahI84

Application Example:

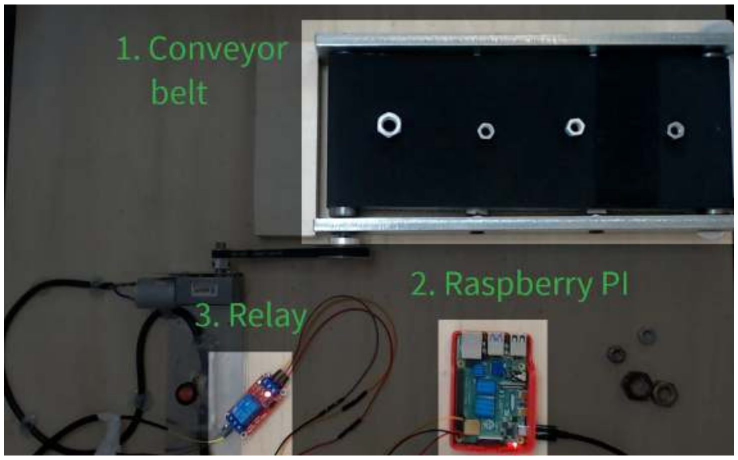
- https://www.youtube.com/watch?v=VmnUKyRj0fw
- https://ivanursul.com/counting-eggs-in-opencv

In this tutorial we will learn how to create a simple prototype to detect objects passing on a conveyor belt.

We will use exagonal nuts as objects. I have two sizes, a small one and a bigger one. If the **small nuts are detected the belt keeps moving**, in case a **bigger one** is detected, the conveyor **is going to stop automatically**.

This project is done using a:

1. **Small conveyor belt** which is turned on/off by using a relay.
2. **Raspberry PI**.
3. **Relay**
4. **Webcam**

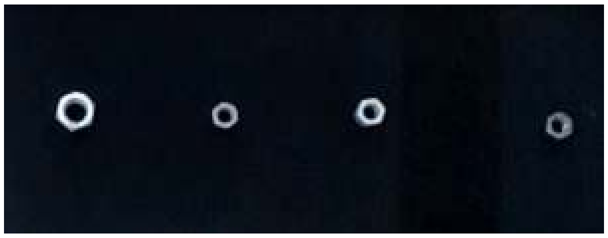


In this article I will only describe the code only for the key parts of the project. If you want to understand all the source code, you can follow the video tutorial where I explain everything.

1. Detect Objects on the conveyor

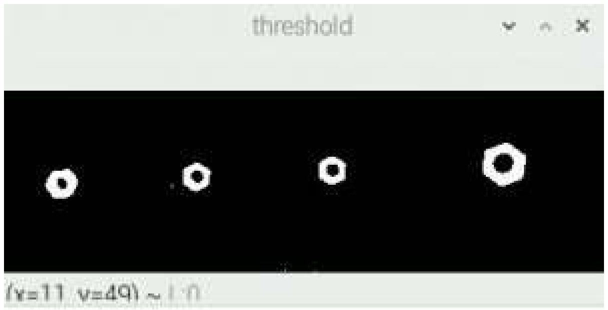
Considering that the view from the webcam takes also elements outside of the belt, we need first to cut the area of the belt and remove everything else.

```
# Belt
belt = frame[209: 327, 137: 280]
gray_belt = cv2.cvtColor(belt, cv2.COLOR_BGR2GRAY)
```



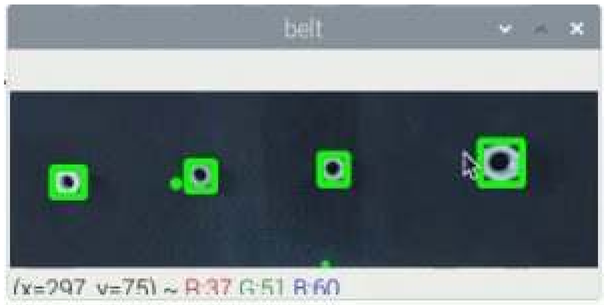
Once we have the belt, it's easy to detect the objects that are on it because the belt is green and the nuts are gray, so a simple threshold operation can easily make a distinction between the belt and the objects.

```
_, threshold = cv2.threshold(gray_belt, 80, 255, cv2.THRESH_BINARY)
```



First thing we need to find the contours. Contours are the boundaries of an object, in this case, the boundaries of the nuts.

```
# Detect the Nuts
_, contours, _ = cv2.findContours(threshold, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
for cnt in contours:
(x, y, w, h) = cv2.boundingRect(cnt)
```

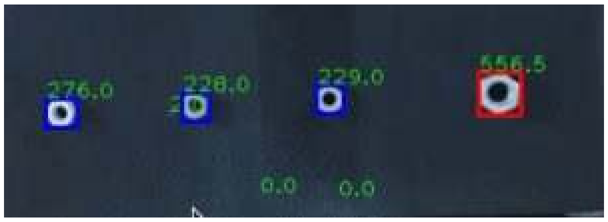


Then we get to key part of the project where we make the actual distinction between a big and a small nut.

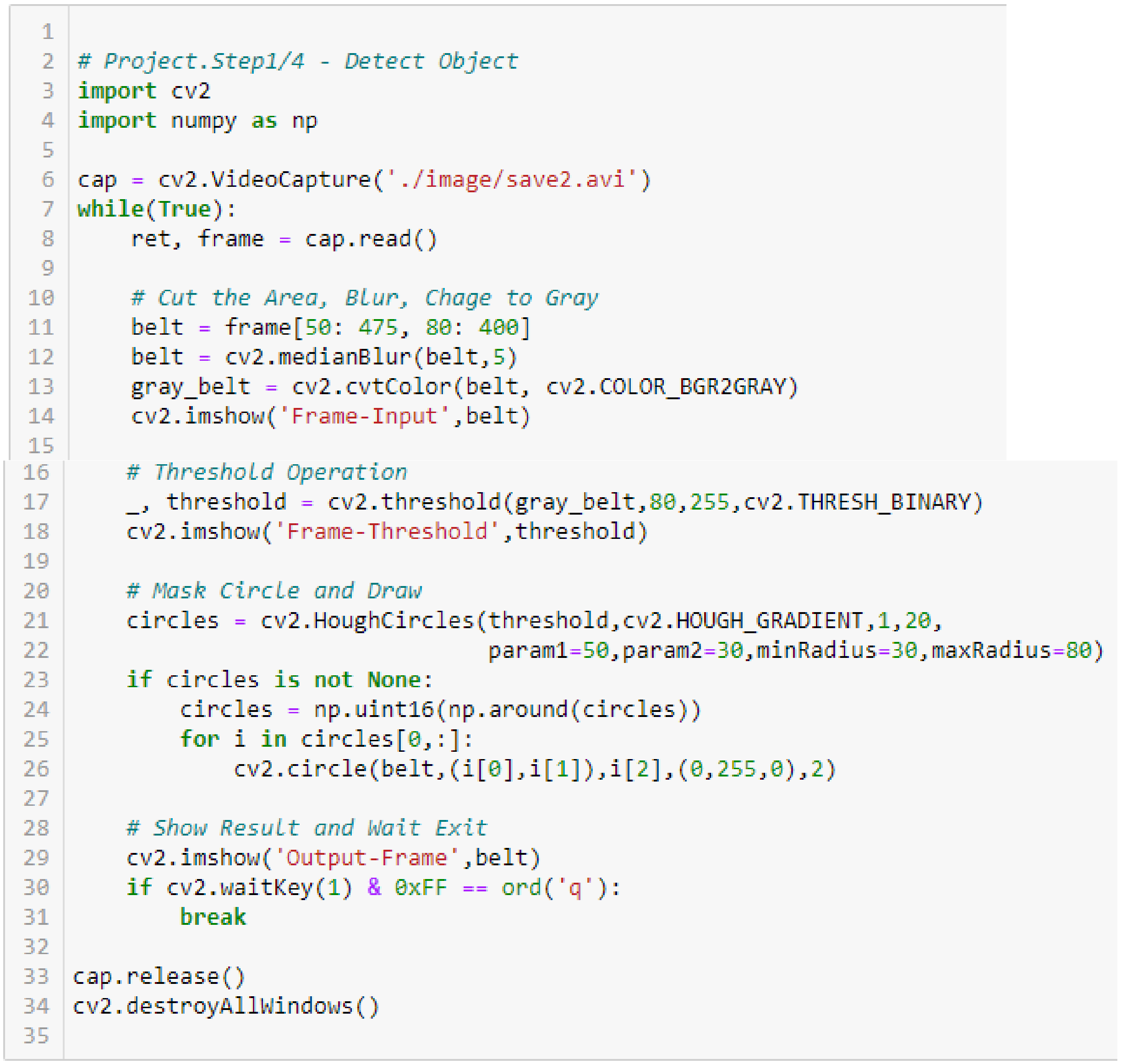
We can do this by calculating the area of the contour. The small nut has a area of around 250 pixels, while the big nut is above 400 pixel.

So we define the conditions, if they are greater than 400 it’s a big nut and we make draw a red rectangle. if the area is between 100 and 400 we make a blue rectangle. Everything smaller than a hundred pixel can be considered as noise, so that we we discard it.

```
# Calculate area
area = cv2.contourArea(cnt)
# Distinguish small and big nuts
if area > 400:
# big nut
cv2.rectangle(belt, (x, y), (x + w, y + h), (0, 0, 255), 2)
elif 100 < area < 400:
cv2.rectangle(belt, (x, y), (x + w, y + h), (255, 0, 0), 2)
cv2.putText(belt, str(area), (x, y), 1, 1, (0, 255, 0))
```



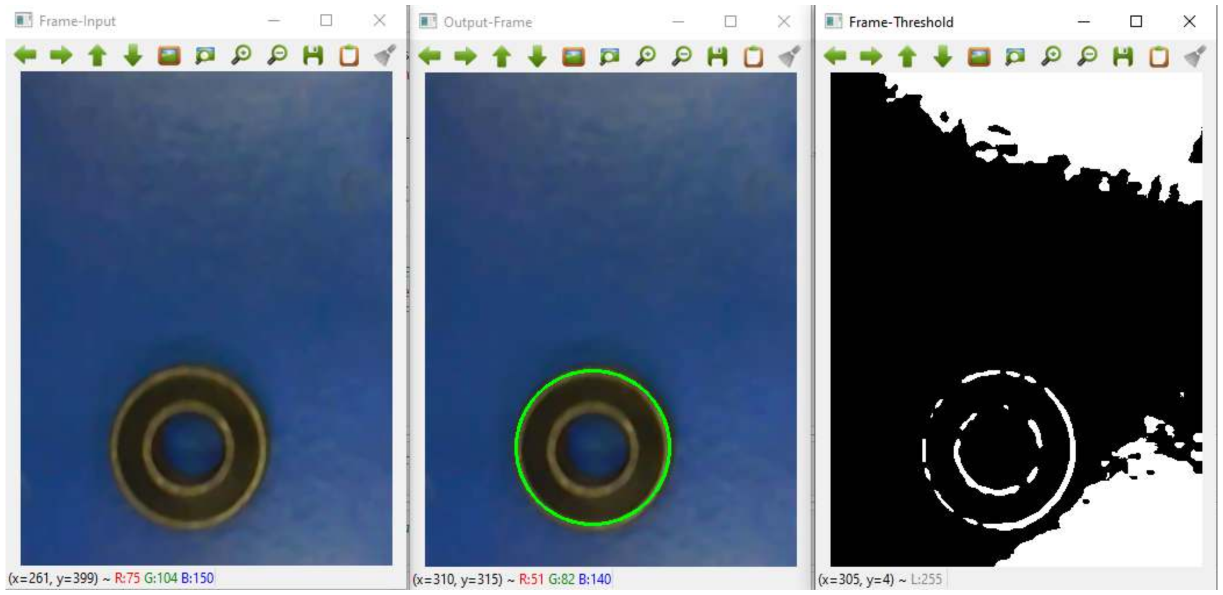
Test 01



```

# Project.Step1/4 - Detect Object
import cv2
import numpy as np

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

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

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

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

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

cap.release()
cv2.destroyAllWindows()

```

```
# Project.Step1/4 - Detect Object
import cv2
import numpy as np

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

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

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

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

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

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


Frame-Input
(x=261, y=399) ~ R:75 G:104 B:150
Output-Frame
(x=310, y=315) ~ R:51 G:82 B:140
Frame-Threshold
(x=305, y=4) ~ L:255

Quiz_406 – กิจกรรมที่ 6/6 – Visual Inspection

- Capture ผลการทำงานที่ได้ลองปฏิบัติ
- ลองใช้ข้อมูลอื่นในการทดสอบ
- อภิปรายผล
- คำถามที่อยากถาม
- บอกแนวการใช้งาน กับงานที่รับผิดชอบ