การจับส่วนแบ่งของภาพ

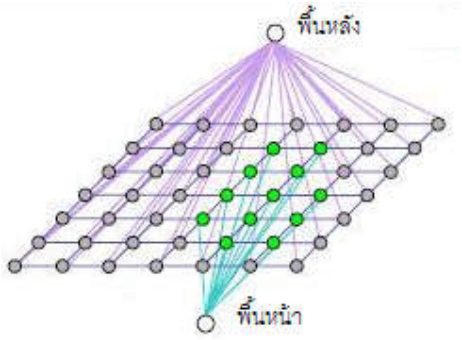
การจับส่วนแบ่งของภาพหมายถึงการ จับภาพเฉพาะส่วนที่เป็นพื้นที่ส่วนที่พื้นหลัง และส่วนเบื้องหน้าออกจากกัน เช่น ภาพคนที่อยู่ในสนามหญ้า ภาพคนถือเป็นภาพเบื้องหน้า ส่วนสนามหญ้าและอื่นๆ ถือเป็นพื้นหลัง ในบทนี้เราจะการสกัดพื้นหน้าและหลังออกจากกัน โดยใช้อัลกอลิทึม ใช้ GMBCUT, Watershed กับภาพนิ่ง และ Gaussian Mixture-based Background/Foreground กับภาพวิดีโอ

แนวคิดการตรวจสอบเส้นขอบที่ต่อเนื่องจนกลายเป็นกรอบ และพื้นที่ที่ติดกันภายในเส้นกรอบ ทำให้แยกส่วนแบ่งของภาพได้ แต่ในบางครั้งเส้นขอบที่ไม่ชัดเจน จึงทำให้แนวคิดการหาเส้นของทำได้ยาก จึงเป็นที่มาของสร้างอัลกอริทึมที่ดีกว่าการใช้การตรวจสอบเส้นกรอบทั่วไป

ภาพเบื้องหน้า (Foreground)

การคัดแยกภาพเบื้องหน้า ด้วยอัลกอริทึม **GrabCut** คิดค้นโดยคณะนักวิจัยของ Microsoft Research Cambridge ประเทศอังกฤษ

แนวคิดคือ ให้ผู้ใช้ขีดกรอบพื้นที่สี่เหลี่ยมที่จะเป็นภาพเบื้องหน้าเอง ซึ่งแสดงว่าพื้นที่นอกกรอบจะเป็นภาพพื้นหลังก่อน (ทำให้เรียกชื่อว่า GrabCut ซึ่งแปลว่า จับ-ตัด) และภาพพื้นหลังนี้จะนำมาถูกเปลี่ยนเทียบกับภาพเบื้องหน้า การเปรียบเทียบใช้วิธีการ Gaussians Mixture Model(GMM) ซึ่งเป็นวิธีการทางสถิติอย่างหนึ่ง ที่ความคล้ายของสี สีที่คล้ายกับสีภายนอกกรอบจะถือว่าเป็นพื้นหลัง และสีที่ต่างจากนอกกรอบถือเป็นภาพเบื้องหน้า แต่ละหน่วยสี (pixel) จะถูกกำกับอักษร (Label) หน่วยสีใดเป็นพื้นหลัง หรือพื้นหน้า ซึ่งต่อไปจะตัดพื้นที่แยกออกได้ว่าเป็นพื้นหน้าและ พื้นหลัง



รูป 1 การกำกับคำพื้นหน้า และหลัง ของแต่ละหน่วยสี

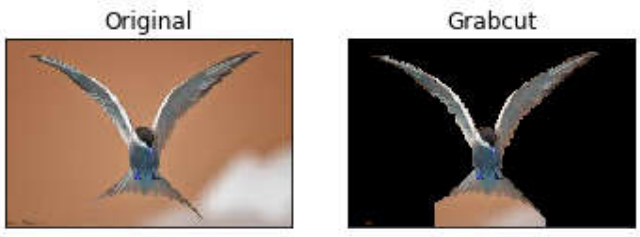
ต่อไปจะเป็นตัวอย่างโปรแกรมการสกัดภาพเบื้องหน้าออกจากพื้นหลัง ด้วยแนวคิดนี้ ฟังก์ชัน grabCut( ) มีตัวแปรตามลำดับดังนี้ดังนี้

- img เป็นภาพที่ต้องการแยกพื้นเบื้องหน้า
- mask เป็นเหมือนหน้ากาก ที่สร้างเป็นพื้นหลัง หรือพื้นหน้าของรูป หลังการทำสร้างการแยกพื้นแล้ว ซึ่งจะเป็นตัวเลข 0, 2 แทนพื้นหลัง และ 1, 3 แทนพื้นหน้า
- bgdModel, fgdModel เป็นอาร์เรย์ศูนย์ขนาด (1, 65) ที่ใช้ในการคำนวณของอัลกอริทึมนี้
- interCount ใช้ในการวนซ้ำหาค่าความเหมือน-คล้ายของภาพ ยิ่งใช้ตัวเลขสูงยิ่งใช้เวลานาน

- mode ในที่นี้คือ GC_INIT_WITH_RECT เพราะใช้กรอบสี่เหลี่ยมเป็นตัวคัดแยก หรือ จะใช้ GC_INIT_WITH_MASK

Code 1.

```
import numpy as np
import cv2

img = cv2.imread('d:/bird_fly.jpg')
mask = np.zeros(img.shape[:2],np.uint8)
bgdModel = np.zeros((1,65),np.float64)
fgdModel = np.zeros((1,65),np.float64)

rect = (10,10,230,250)
cv2.grabCut(img,mask,rect,bgdModel,fgdModel,5,cv2.GC_INIT_WITH_RECT)

mask2 = np.where((mask==2)|(mask==0),0,1).astype('uint8')
img_new = img*mask2[:,:,np.newaxis]

cv2.imshow('Grabcut',img_new)
cv2.waitKey()
cv2.destroyAllWindows()
```



รูป 2 การใช้ GrabCut สกัดภาพเบื้องหน้า

จากตัวอย่างนี้ มีลำดับการทำงานคือ

1. อ่านภาพ เก็บในตัวแปรชื่อ img
2. สร้างหน้ากาก mark ให้มีขนาดเข้าภาพต้นฉบับ ในระนาบสองมิติ เป็นพื้นดำหมด (อาร์เรย์ศูนย์) เช่น จากรูปเดิม มีขนาด Shape (128, 277, 3) ไปเป็น (128, 277)
3. สร้างแบบจำลอง bgdModel, fgdModel เป็นพื้นที่ชั่วคราว เพื่อใช้อ้างอิงสี่ที่ต่างกันของภาพพื้นหน้า - หลัง
4. ขีดกรอบ rect ให้ครอบคลุมรูปพื้นหน้าทีต้องการ ขั้นตอนนี้เป็นขั้นตอนสำคัญ การเลือกพื้นที่เบื้องหน้าที่ตรงกับที่ต้องการที่สุด จะทำให้การทงานของอัลกอริทึมนี้ทำงานได้ถูกต้องที่สุด
5. เรียก grabCut( ) แล้วใส่ตัวแปรต่างๆ คือ ภาพต้นฉบับ (img) มาสค์ (mask) หรือหน้ากาก ต้นแบบพื้นหลัง (bgdModel) ต้นแบบพื้นหน้า (fgdModel) จำนวนการทำซ้ำ (5) และโหมดการทำงานแบบสี่เหลี่ยม (GC_INT_WITH_RECT)
6. สร้างกรอบ mark2 ซึ่งมีค่า 0 - 3 จากใช้ฟังก์ชัน grabCut โดยที่ 0 และ 2 ให้พื้นหลัง จึงตั้งให้ (0, 2) เป็นเลข 0 และตั้งพื้นเบื้องหน้า (1,3) เป็นเลข 1

7. สร้างภาพพื้นหลังและหน้า img_new ที่มาจาก ภาพต้นฉบับ (img) คูณกับ mark2 ที่ทำเป็นสามมิติแล้วด้วยการเพิ่ม newaxis ซึ่งจะเห็นว่า พื้นที่ใดคูณด้วย 0 จะเป็นสีดำซึ่งจะกลายเป็นภาพพื้นหลัง และพื้นที่ใดคูณด้วย 1 จะเป็นภาพเดิมซึ่งจะกลายเป็นภาพพื้นหน้า

ภาพเบื้องหน้า-หลัง

การแยกเบื้องหน้าหลัง โดยไม่ต้องตัดแยกเองก่อน ซึ่งดีกว่าการใช้ Grabcut คืออัลกอริทึม **Watershed** ซึ่งแปลว่าลุ่มน้ำ แนวคิดคือการสร้างาลุ่มน้ำจากภาพโทนเทาที่มองดังเป็นภูมิประเทศ โทนสีหนักเปรียบได้ดังเป็นที่เนิน และโทนสีเบาเปรียบได้ดังเป็นลุ่มน้ำหรือที่ต่ำ แล้วทำการแบ่งพื้นที่ตามขนาดน้ำหนังสีดังกล่าวนี้

ตัวอย่างต่อไปนี้แสดงถึงการสกัดภาพพื้นหลังและภาพเบื้องหน้า ซึ่งมีขั้นตอนหลายอย่าง เริ่มจากการอ่านภาพให้อยู่ในรูปโทนเทาก่อน

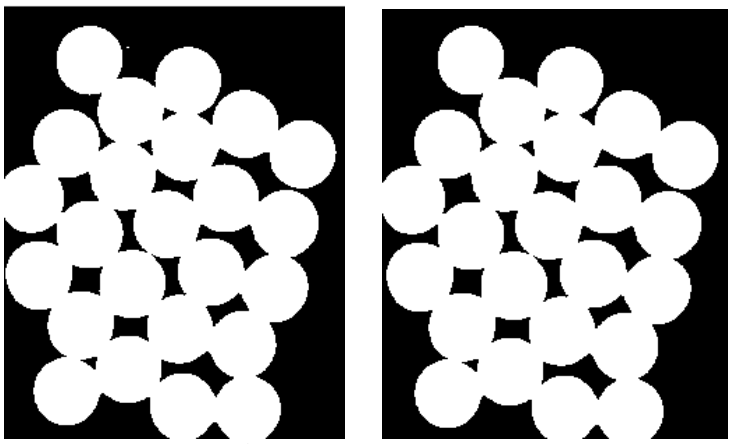
Code 2.

```
import numpy as np
import cv2
img = cv2.imread('d:/water_coins.jpg')
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
ret, thresh = cv2.threshold(gray,0,255,cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)

kernel = np.ones((3,3),np.uint8)
opening = cv2.morphologyEx(thresh,cv2.MORPH_OPEN,kernel, iterations = 2)
sure_bg = cv2.dilate(opening,kernel,iterations=3)

dist_transform = cv2.distanceTransform(opening,cv2.DIST_L2,5)
ret, sure_fg = cv2.threshold(dist_transform,0.7*dist_transform.max(),255,0)

sure_fg = np.uint8(sure_fg)
unknown = cv2.subtract(sure_bg,sure_fg)

ret, markers = cv2.connectedComponents(sure_fg)
markers = markers+1
markers[unknown==255] = 0

markers = cv2.watershed(img,markers)
img[markers == -1] = [255,0,0]

cv2.imshow('coins',img)
cv2.waitKey()
cv2.destroyAllWindows()
```

รูป 3 การใช้ Watershed สกัดภาพเบื้องหน้า-หลัง

หลังจากนั้นหาค่าความต่างระดับสีด้วยฟังก์ชัน threshold ทำให้ได้ภาพขาวและดำ ต่อจากนั้นจะต้องปรับจุดด่างต่างๆ (เม็ดสีขาวเล็กๆ) โดยการทำฟังก์ชัน morphologyEx โดยใช้ kernel เป็นฟิลท์เตอร์

```
kernel = np.ones((3,3),np.uint8)
opening = cv2.morphologyEx(thresh,cv2.MORPH_OPEN,kernel, iterations = 2)
```



รูป 4 รูปที่มีจุดด่าง (ซ้าย) และลบจุดด่าง (ขวา)

และขยายวงภาพพื้นให้กว้างขึ้นเล็กน้อยจะได้ภาพครอบคลุมพื้นหลังทั้งหมด

```
sure_bg = cv2.dilate(opening,kernel,iterations=3)
```

ในทางกลับกัน เราใช้ฟังก์ชัน distancTransform เพื่อหาพื้นที่เบื้องหน้า และหาค่าต่างระดับสีด้วยฟังก์ชัน threashold อีกครั้ง ในครั้งนี้เราจะได้ ภาพเบื้องหน้า

```
dist_transform = cv2.distanceTransform(opening,cv2.DIST_L2,5)
ret, sure_fg = cv2.threshold(dist_transform,0.7*dist_transform.max(),255,0)
```

ณ ขณะนี้เราได้พื้นที่ของทั้งภาพเบื้องหน้า และเบื้องหลังแล้ว เมื่อเราลบหรือหาส่วนต่างๆ ของสองพื้นที่นี้ เราจะได้เส้นขอบของสองพื้นนี้ ซึ่งเรียกส่วนพื้นที่นี้ว่า unknown (ป้ายชื่อ)

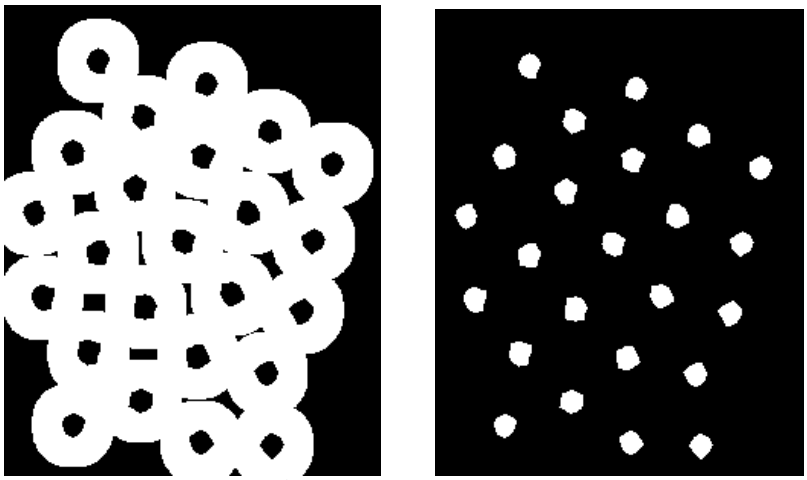
```
sure_fg = np.uint8(sure_fg)
unknown = cv2.subtract(sure_bg,sure_fg)
```

เมื่อได้ส่วนต่างของสองพื้นที่แล้ว เราจะใช้ส่วนต่างของสองพื้นนี้เป็นเหมือนกำแพงกันให้น้ำไหลมารวมกัน ส่วนต่างนี้ก็คือเส้นกรอบระหว่างพื้นหลังและพื้นหน้านั้นเอง และสร้างหน้ากากขึ้นจากภาพพื้นเบื้องหน้า

```
ret, markers = cv2.connectedComponents(sure_fg)
```

เพื่อให้แน่ใจว่า ภาพพื้นทั้งหมดเป็นสีระดับ 0 จึงบวกให้เพิ่มขึ้นอีก 1 เช่น เมื่อค่าเดิมเป็น 0 ก็จะกลายเป็น 1 และกำหนดให้ภาพหน้ากากที่มีป้ายชื่อว่า unknown ใดที่มีค่า 255 จะมีค่าเป็น 0 (เป็นรูปขาวดำ จึงมีเพียงสี 255 กับ 0)

```
markers = markers+1
markers[unknown==255] = 0
```



รูป 5 รูปป้ายชื่อ unknown (ซ้าย) และ sure_fg (ขวา)

สุดท้าย ใส่น้ำลงแผงกันได้ ให้กำแพงกันเป็นสีแดง แผงกันจะมีค่า -1 กำหนดเป็นสีแดง

```
markers = cv2.watershed(img,markers)
img[markers == -1] = [255,0,0]
```

ภาพเบื้องหลัง (Background)

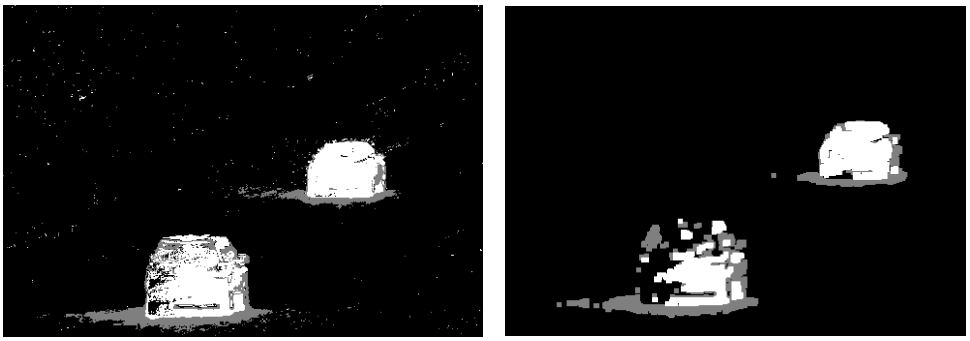
ในตัวอย่างที่ผ่านมา จะเห็นว่าต้องเตรียมขั้นตอนต่างๆ มากมาย ซึ่งมีข้อดีคือ ทำให้เรารู้ว่า วิธีการทำมีความเป็นมาอย่างไร แต่จะวิธีต่อไปนี้ ดูเหมือนว่าทุกขั้นตอนจะถูกรวมในฟังก์ชันเดียว ซึ่งเหมาะมาถ้าทำงานกับภาพเคลื่อนไหวที่มีหลายๆ ภาพต่อเนื่องกัน ซึ่งต่อไป เราสามารถที่จะทำการติดตามวัตถุ หรือภาพเบื้องหน้าได้ว่าเคลื่อนที่ไปในทิศทางใด

Code 3.

```
import numpy as np
import cv2
cap = cv2.VideoCapture('d:/roadTraffic.mp4')
fgbg = cv2.createBackgroundSubtractorMOG2()
while(1):
    ret, frame = cap.read()
    fgmask = fgbg.apply(frame)
    cv2.imshow('frame',fgmask)
    k = cv2.waitKey(30) & 0xff
    if k == 27:
        break
cap.release()
cv2.destroyAllWindows()
```

ในตัวอย่างนี้ใช้ภาพรถยนต์วิ่งไปมา และสกัดภาพพื้นหน้า - หลัง ด้วยฟังก์ชันเดียวคือ createBackgroundSubtractorMOG2 ในการแสดงภาพแต่ละภาพ ใช้ การวนซ้ำตลอดเวลา จาก while(1) และให้หยุดเมื่อกดคีย์ ESC (k==27)

ภาพที่ได้ จะยังคงมี จุดด่าง อยู่มาก ดังนั้นแล้วต้องแก้ด้วย การทำภาพให้สะอาดขึ้น ด้วย cv2.morphologyEx( ) ได้เคยยกตัวอย่างมาก่อนหน้านี้ หรือใช้ฟังก์ชันภาพทำเลื่อนอื่นๆ (*น่าจะทำทดลองทำดูด้วยตนเอง*) หากทำได้ควรจะได้ภาพ ที่ลบจุดด่างออกได้ดังรูปต่อไปนี้



รูป 6 การแยกพื้นหลัง (ซ้าย) และ การแยกพื้นหลังที่มีลบจุดด่านแล้ว (ขวา)

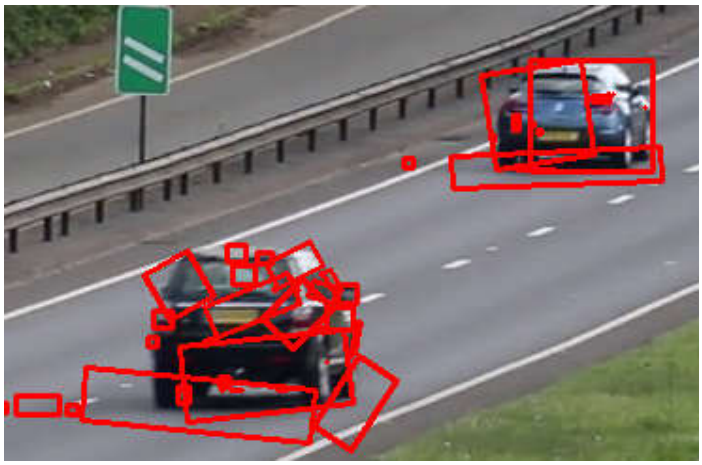
การตีกรอบติดตามวัตถุ

และจากความรู้ก่อนหน้าที่ เราสามารถลบจุดด่างได้ และเรายังจะสามารถวาดกรอบติดตาม การเคลื่อนไขของแต่ละภาพได้

ตัวอย่างต่อไปนี้ ใช้ สกัดภาพพื้นหน้า-หลัง ด้วยวิธีการก่อนหน้านี้ ได้เพิ่มส่วนการลบจุดด่างออก ต่อด้วยการตีกรอบในอัลกอริทึม Canny (หากจำกันได้วิธีการตีกรอบนี้ ในบทที่ว่าด้วยการหาขอบ และการตีกรอบ ซึ่งวิธีการ Canny ทำได้ดี)

Code 4.

```
import numpy as np
import cv2
cap = cv2.VideoCapture('d:/roadTraffic.mp4')
fgbg = cv2.createBackgroundSubtractorMOG2()
while(1):
    ret, frame = cap.read()
    fgmask = fgbg.apply(frame)
    kernel = np.ones((3,3),np.uint8)
    #remove noise
    opening = cv2.morphologyEx(fgmask,cv2.MORPH_OPEN,kernel, iterations = 2)
    thresh = cv2.Canny(opening, 30, 120)
    image, contours, hier = cv2.findContours(thresh,
                                             cv2.RETR_EXTERNAL,
                                             cv2.CHAIN_APPROX_SIMPLE)
    for c in contours:
        x,y,w,h = cv2.boundingRect(c)
        cv2.rectangle(image, (x,y), (x+w, y+h), (0, 255, 0), 2)
        rect = cv2.minAreaRect(c)
        box = cv2.boxPoints(rect)
        box = np.int64(box)
        cv2.drawContours(frame, [box], 0, (0,0, 255), 2)

    cv2.imshow('frame',frame)
    k = cv2.waitKey(30) & 0xff
    if k == 27:
        break
cap.release()
cv2.destroyAllWindows()
```



รูป 7 การใช้ Canny ตีกรอบ

จากตัวอย่าง การทำกรอบทำในทุกกรอบที่หาได้ ซึ่งความจริง กรอบที่เล็ก ไม่น่าจะใช้ว่าเป็น วัตถุรถที่เราเลือก ดังนั้นทางที่ดี คือเราเลือก กรอบที่มีขนาดใหญ่ ที่ระดับหนึ่ง ที่คาดว่าน่าจะใช้วัตถุรถ ดังนั้นแล้ว ควรเพิ่ม การกรองการวาดกรอบ เช่น การวาดกรอบที่มีขนาด เกิน 800

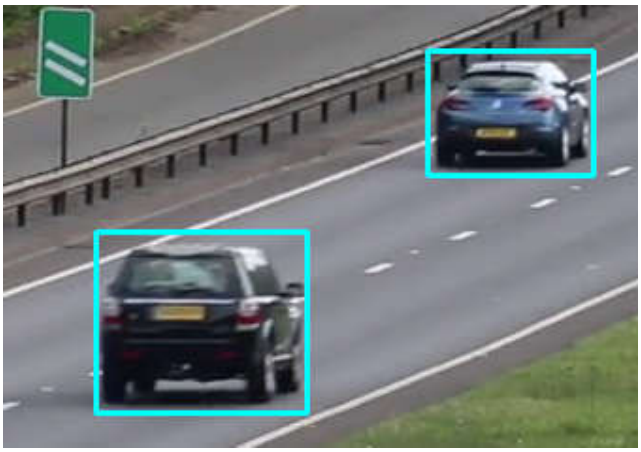
```
for c in contours:
        if cv2.contourArea(c) > 800:
             (x,y,w,h) = cv2.boundingRect(c)
             cv2.rectangle(frame, (x,y), (x+w, y+h), (255, 255, 0), 2)
```

ถึงแม้การเลือกกรอบจะเลือกรูปวัตถุรถ ได้ดีบาง ไม่ได้บ้าง จากตัวอย่างที่ผ่านมา เรามาศึกษากันต่อไปว่า จะใช้แนวทางที่ดีกว่า ซึ่งไม่ได้เขียนเขียนโปรแกรมอะไรมากเลย ลองพิจารณาจากตัวอย่างนี้

Code 5.

```
import numpy as np
import cv2
cap = cv2.VideoCapture('d:/roadTraffic.mp4')
fgbg = cv2.createBackgroundSubtractorMOG2()
while(1):
    ret, frame = cap.read()
    fgmask = fgbg.apply(frame)
    kernel = np.ones((3,3),np.uint8)

    th = cv2.threshold(fgmask.copy(), 244, 255, cv2.THRESH_BINARY)[1]
    dilated = cv2.dilate(th, cv2.getStructuringElement(cv2.MORPH_ELLIPSE,
                                                       (3,3)), iterations = 2)
    image, contours, hier = cv2.findContours(dilated, cv2.RETR_EXTERNAL,
                                                       cv2.CHAIN_APPROX_SIMPLE)
    for c in contours:
        if cv2.contourArea(c) > 1600:
            (x,y,w,h) = cv2.boundingRect(c)
            cv2.rectangle(frame, (x,y), (x+w, y+h), (255, 255, 0), 2)

    cv2.imshow('frame',frame)
    k = cv2.waitKey(30) & 0xff
```

```
    if k == 27:
        break

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



รูป 8 การใช้ Canny ตีกรอบ

แนวคิดหลักคือ เมื่อได้หน้ากากพื้นหน้า fmask ของแต่ละเฟรม ให้ทำค่าระดับต่างสี ด้วย threshold ให้ทุกเม็ดสีจากภาภ fgmask.copy( ) (ตัวแปรที่ 1) ที่มีสีตั้งแต่ 244 (ตัวแปรที่ 2) เป็นค่าต่างระดับสี (threshold) และไม่เกิน สี 255 ให้ (ตัวแปรที่ 3) ให้ทำเป็นค่า สองสี (ตัวแปรที่ 4) ซึ่งค่าใด ที่มีค่า ตั้งแต่ 0 -243 จะเป็นสีดำ และ สี 244 ขึ้นไปเป็นสี ขาว และเลือกส่งเฉพาะ ตัวคืนค่า ที่ 2 หรือ ค่า 1 ในระบบอาร์เรย์ (ค่า [1] หลังฟังก์ชัน threshold) ตามปกติจะคือสองค่า คือ ret ซึ่งคือรูปแบบการทำงานแบบทั่วไป (global threshold) และ th (ภาพที่ทำต่างระดับสีแล้ว)

ต่อมาด้วยทำพื้นหน้าจากฟังก์ชัน dilate ที่ฟองครอบวัตถุพื้นหน้าไว้

```
th = cv2.threshold(fgmask.copy(), 244, 255, cv2.THRESH_BINARY)[1]
dilated = cv2.dilate(th, cv2.getStructuringElement(cv2.MORPH_ELLIPSE,
                                                   (3,3)), iterations = 2)
```

ต่อมาหาเส้นกรอบ จาก dilate แล้ววาดรูปกรอบสี่เหลี่ยม ของแต่ละเฟรม

```
cv2.rectangle(frame, (x,y), (x+w, y+h), (255, 255, 0), 2)
```

ด้วยวิธีการพื้นฐานนี้ เราจะได้กรอบทีติดตามวัตถุ ดังที่ทำ (อีกครั้ง) ระบุตัววัตถุ ทำเส้นโครงรอบวัตถุ และเขียนเส้นโครงรอบวัตถุบน เฟรมต้นฉบับ

อ้างอิง

https://docs.opencv.org/3.1.0/d8/d83/tutorial_py_grabcut.html

https://docs.opencv.org/3.3.0/db/d5c/tutorial_py_bg_subtraction.html

https://docs.opencv.org/3.3.1/d3/db4/tutorial_py_watershed.html