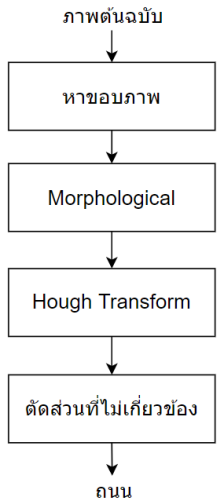
การตรวจจับเส้นบนถนน

Road Lane line detection

พิชยุทธ บุญตน 61011212012

---

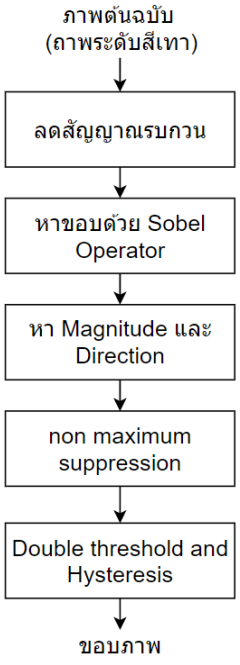
1. ขั้นตอนและวิธีการ



รูปที่ 1 ขั้นตอนและวิธีการ

1.1 การหาขอบภาพ (Edge Detection)

ขั้นตอนนี้ใช้วิธีการหาขอบด้วย Canny Edge Detection คือการค้นหาขอบภาพโดยพิจารณาทั้งขนาดและทิศทางของอนุพันธ์ลำดับที่ 1 ของภาพระดับสีเทาซึ่งมีขั้นตอนดังนี้



รูปที่ 2 Canny Edge Detection

ขั้นตอนหาขอบภาพ ได้ใช้หลักสถิติในการหาเกณฑ์การตัดสินใจด้วย Otsu Algorithm ซึ่งให้ค่าที่ดีที่สุดสำหรับภาพนั้น ๆ เพื่อให้ได้เส้นที่ดีที่สุด เพื่อใช้แทนการกำหนดค่าคงที่จากพารามิเตอร์ของ Canny ฟังก์ชันใน OpenCV แสดงโค้ดได้ดังนี้

```
1.    img = cv2.imread("1.jpg",0)
2.    ret, thr = cv2.threshold(gray, 0, 255,
                        cv2.THRESH_TOZERO+cv2.THRESH_OTSU)
```

ใน OpenCV มีฟังก์ชันให้เรียกใช้ Canny Edge Detection ได้เลยโดยส่งภาพที่ลดสัญญาณรบกวนแล้วเข้าไปแสดงโค้ด Canny Edge Detection และ Otsu Algorithm ได้ดังนี้

```
1.    img = cv2.imread("1.jpg",0)
2.    img = cv2.GaussianBlur(img, (5,5),1.4)
3.    ret, thr = cv2.threshold(gray, 0, 255,
                        cv2.THRESH_TOZERO+cv2.THRESH_OTSU)
4.    dst = cv2.Canny(img, ret, 200, None, 3)
```

ภาพ (ก)

ภาพ (ข)

รูปที่ 3 ผลลัพธ์การหาขอบ

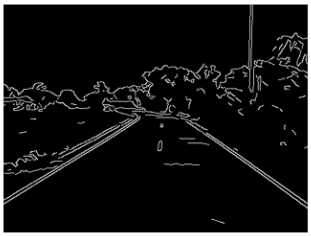
เมื่อแสดงภาพที่หาขอบด้วย Canny Edge Detection แล้วจะได้ ภาพ (ข) ดังรูปที่ 3

1.2 เปลี่ยนรูปร่าง (Morphological)

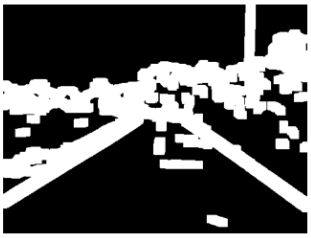
การประมวลผลภาพกับรูปร่างและโครงร่างของภาพ โดยมีความสามารถในการย่อขยายจุดพิกเซลของภาพได้ โดยการหาเส้นถนนโดยขั้นตอนนี้ใช้ Structural Element (SE) แบบ RECT ขนาด 3 x 3 และนำภาพไปทำ Dilation

ที่ 4 รอบ และนำภาพใช้ Erosion ต่อเนื่องกันที่ 2 รอบ จากนั้นนำภาพทั้งสองไปหักล้างกันเรียกว่า Boundary Extraction โดยแสดงโค้ดได้ดังนี้

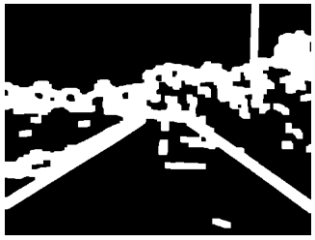
```
##dst คือภาพที่หาขอบมาแล้ว
kernel = cv2.getStructuringElement(cv2.MORPH_RECT,(3,3))
dilation = cv2.dilate(dst,kernel,iterations = 4)
erosion = cv2.erode(dilation,kernel,iterations = 2)
boun = dilation - erosion
```



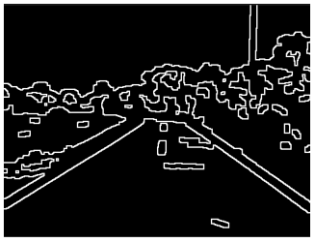
ภาพ (ก)



ภาพ (ข)



ภาพ (ค)



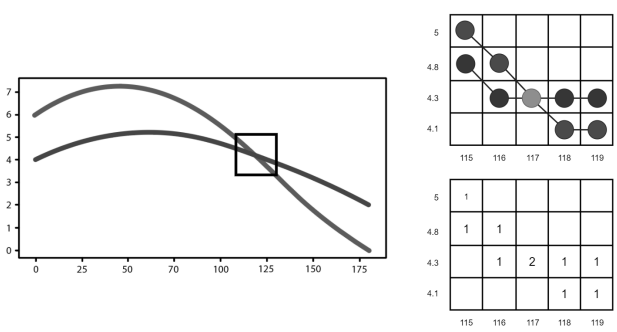
ภาพ (ง)

รูปที่ 4 ชุดข้อมูลภาพการทำ Morphological

ผลลัพธ์เมื่อทำ Dilation ที่ 4 รอบ ภาพ (ข) และนำภาพที่ทำ Dilation ไปทำ Erosion ต่อกัน 2 รอบ ภาพ (ค) นำภาพทั้งสองมาหักล้างกัน ภาพ (ง) ดังรูปที่ 4

1.3 Hough Transform

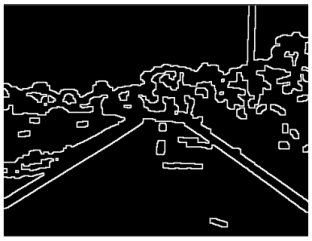
Hough Transform คือเทคนิคสำหรับการตรวจจับรูปร่างในทางคณิตศาสตร์ โดยเส้นนั้นจะสามารถแสดงเป็นสูตรในรูปแบบของ พาราเมตริก จะได้ว่า $\rho = x\cos\theta + y\sin\theta$ ค่า ρ จะถูกเก็บไว้ในตารางการให้คะแนนแสดงตัวอย่างตารางให้คะแนนโดยตั้งองศาไว้ที่ 0 ถึง 180 แสดงรูปได้ดังนี้



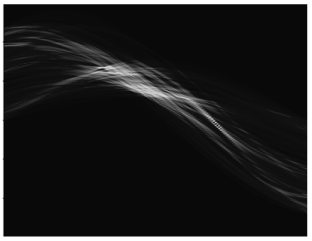
รูปที่ 5 ตัวอย่างตารางการให้คะแนน

โดย OpenCV มีให้เรียกใช้ฟังก์ชัน cv2.HoughLines เมื่อใช้คำสั่งแล้วผลลัพธ์ที่ได้จะออกมาเป็นชุดข้อมูล (Array) ต้องใช้การวนลูปดึงเส้นออกมา แสดงโค้ดได้ดังนี้

```
##dst คือภาพที่หาขอบมาแล้ว
lines = cv2.HoughLines(boun, 1, np.pi/180, 150, None, 0, 0)
if lines is not None:
    for i in range(0, len(lines)):
        rho = lines[i][0][0]
        theta = lines[i][0][1]
        a = math.cos(theta)
        b = math.sin(theta)
        x0 = a * rho
        y0 = b * rho
        pt1 = (int(x0 + 1000*(-b)), int(y0 + 1000*(a)))
        pt2 = (int(x0 - 1000*(-b)), int(y0 - 1000*(a)))
```



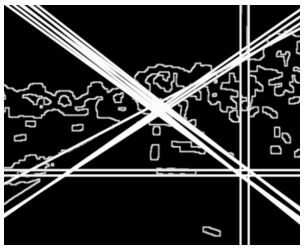
ภาพ (ก)



ภาพ (ข)

รูปที่ 6 ผลลัพธ์เมื่อใช้ฟังก์ชัน cv2.HoughLines

ต่อจากนั้นทำการขยายเส้นในโค้ดบรรทัดที่ 11 และ 12 ด้วยการนำองศาของแต่ละเส้นมาขยายออกเพื่อวาดเส้นทับกับรูปเป็นเส้นตรง ณ ตำแหน่งของจุดนั้น ๆ ผลลัพธ์ที่ได้แสดงให้เห็นในรูปที่ 7 ภาพ (ค) ดังนี้



ภาพ (ค)

รูปที่ 7 ชุดข้อมูลภาพการทำ Hough Transform

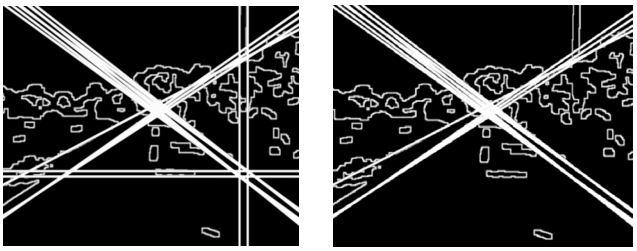
จากรูปที่ 7 ภาพ (ค) ผลลัพธ์ที่ได้ยังมีเส้นที่ที่ไม่ใช่เส้นของถนน จึงแก้ปัญหาด้วยการหาความชันของเส้นทุกเส้น โดยเลือกความชันตั้งแต่ 0.25 ถึง 0.8 เพราะมีความเอียงที่ใกล้เคียงมุมเส้นของถนนที่สุด โดยมีสมการการหาดังนี้

$$slop = \frac{y_2 - y_1}{x_2 - x_1}$$

ผลลัพธ์ที่ได้ออกมาสามารถกรองเส้นที่ไม่เกี่ยวข้องได้ โดยจะใส่ไว้ในลูปการดึงเส้นออกมาจากชุดข้อมูล (Array) ของ cv2.HoughLines แสดงโค้ดได้ดังนี้

```
lines = cv2.HoughLines(boun, 1, np.pi / 180, 150, None, 0, 0)
##Hough คัดเลือกเส้น
if lines is not None:
    for i in range(0, len(lines)):
        rho = lines[i][0][0]
        theta = lines[i][0][1]
        a = math.cos(theta)
        b = math.sin(theta)
        x0 = a * rho
        y0 = b * rho
        pt1 = (int(x0 + 1000*(-b)), int(y0 + 1000*(a)))
        pt2 = (int(x0 - 1000*(-b)), int(y0 - 1000*(a)))
        ##หาความชัน
        slop = abs((pt2[1] - pt1[1])/(pt2[0] - pt1[0]))
        if (slop > 0.25 and slop < 0.8):
            lineP1.append(pt1)
            lineP2.append(pt2)
```

แสดงผลลัพธ์ให้เห็นใน รูปที่ 8 ภาพ (ข) ดังนี้



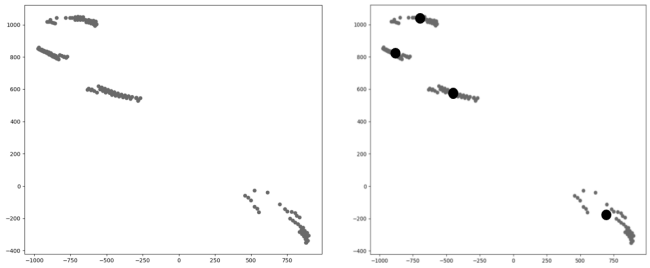
ภาพ (ก) ภาพ (ข)

รูปที่ 8 ผลลัพธ์การหาความชัน

จากรูปที่ 8 ภาพ (ข) ผลลัพธ์ที่ได้ยังมีเส้นที่มากเกินไป ขั้นตอนต่อไปจะนำเส้นทั้งหมดมาหาค่าเฉลี่ยเพื่อหาค่ากึ่งกลางของทุกเส้น ทำให้เกิดเพียงเส้น ๆ เดียวโดยใช้วิธี K-mean Clustering โดยนำเส้นทุกเส้นรวมเป็นชุดข้อมูลเดียวกันและส่งไปจัดกลุ่มด้วยโค้ดฟังก์ชันดังนี้

```
def kmean(kmean):
    km = KMeans(
        n_clusters=2, init='random',
        n_init=2, max_iter=300,
        tol=1e-04, random_state=0
    )
    km.fit_predict(kmean)
    return km.cluster_centers_
```

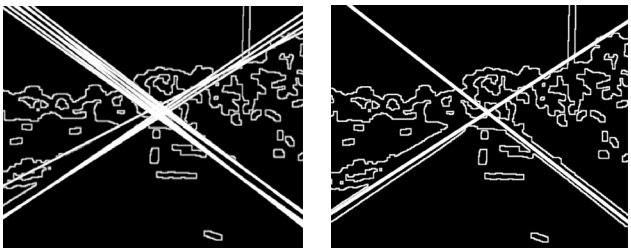
ผลลัพธ์ที่ได้ในรูปที่ 9 ภาพ (ก) คือการรวมชุดข้อมูลเส้นทั้งหมดให้อยู่ด้วยกัน ส่วนภาพ (ข) คือการหา K-mean Clustering



ภาพ (ก) ภาพ (ข)

รูปที่ 9 K-mean Clustering

จากรูปที่ 9 ภาพ (ข) ทำให้เกิดจุดตรงกลางระหว่างกลุ่ม เมื่อนำค่าตรงกลางเหล่านี้ไปสร้างจุดจะได้เส้นทั้งสี่จุด หรือได้ออกมาสองเส้นทำให้เกิดผลลัพธ์ในรูปที่ 10 ภาพ (ข) ดังนี้

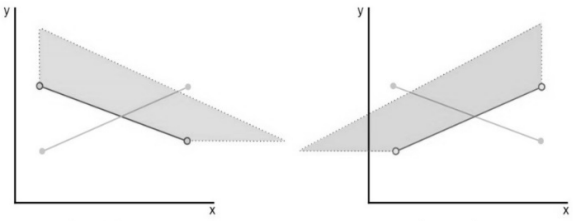


ภาพ (ก) ภาพ (ข)

รูปที่ 10 วาดเส้นใหม่หลังจากจัดกลุ่ม

1.4 ตัดส่วนที่ไม่เกี่ยวข้อง

เมื่อได้เส้นทั้งสองเส้นมาแล้ว นำเส้นแต่ละเส้นมาขยายออกเป็นสี่จุด ดังนั้นภาพผลลัพธ์ที่ได้มีสองเส้นจะได้แปดจุด แสดงตัวอย่างวิธีคิดได้ดังนี้



ภาพ (ก) ภาพ (ข)

รูปที่ 11 การขยายเส้นทั้งสี่จุด

โดย ถ้าเป็นเส้นที่ 1 ภาพ (ก) ให้เพิ่มจุดอีกสองจุด (x+1000, y), (x, y+1000) ถ้าเป็นเส้นที่ 2 ภาพ (ข) ให้เพิ่มจุดอีกสองจุด (x-1000, y), (x, y+1000) และใช้คำสั่งของ

OpenCV ในการเติบจุดและเติมสีดำลงไป โดยแสดงโค้ดได้ดังนี้

```
1.      ##ทำให้เส้นสูงขึ้นเพื่อ mask เส้นที่ 1
2.      vertices = np.array([pt1,pt3,[pt3[0]-1000,pt1[1]-
                                                2000]],np.int32)
3.      pts = vertices.reshape((-1, 1, 2))
4.      cv2.polylines(img, [pts], isClosed=True, color=(0, 0, 0),
                                                thickness=20)
5.      cv2.fillPoly(img, [pts], color=(0, 0, 0))
6.
7.      ##ทำให้เส้นสูงขึ้นเพื่อ mask เส้นที่ 2
8.      vertices = np.array([pt2,pt4,[pt4[0]-500,pt2[1]-
                                                3000]],np.int32)
9.      pts = vertices.reshape((-1, 1, 2))
10.     cv2.polylines(img, [pts], isClosed=True, color=(0, 0, 0),
                                                thickness=20)
11.     cv2.fillPoly(img, [pts], color=(0, 0, 0))
```

เมื่อแสดงภาพแล้วจะได้ผลลัพธ์ดังนี้

รูปที่ 12 ผลลัพธ์การหาถนน