DOCUMENT PERSPECTIVE TRANSFORMATION

DENPHUM APHIMETEETAMRONG 6010501113

โจทย์ปัญหา

ปรับปรุงภาพถ่ายที่ไม่ตั้งฉากกับเลนส์กล้องซึ่งทำให้ภาพและตัวอักษรบิดเบี้ยว

ให้กลับมาอยู่ในลักษณะปกติ

ความสำคัญ และ ประโยชน์

ทำให้ตัวอักษรที่อ่านได้ยาก อ่านได้ง่ายขึ้น และ สามารถเอาภาพที่เป็นปกติแล้วไปทำ TEXT RECOGNITION ต่อได้

แนวทางการแก้ปัญหา

Preprocessing

1. Input RGB image
2. RGB –> GRAY
3. OTSU Thresholding
4. Erosion
5. Fill image
6. Find the biggest object

Processing

1. Corner Harris Detection
2. Coordinate of Corner Object
3. Max Area (Shoelace formula)
4. Assign Coordinate to each corner
5. Calculate Transfer Function
6. warpPerspective –> Output

https://en.wikipedia.org/wiki/Shoelace_formula

แนวทางการแก้ปัญหา

Input RGB image

Gray Scale Image

```
import numpy as np
import cv2
from itertools import permutations
```

แปลงภาพจาก RGB เป็น GRAY SCALE

```
gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
```

แนวทางการแก้ปัญหา

Gray Scale Image

Binary image

แปลงภาพจาก GRAY SCALE เป็น BINARY โดยใช้ otsu thresholding

```
ret,bw =
cv2.threshold(gray,0,255,cv2
.THRESH_BINARY+cv2.THRESH_OT
SU)
```

แนวทางการแก้ปัญหา

Binary image

Eroded Binary image

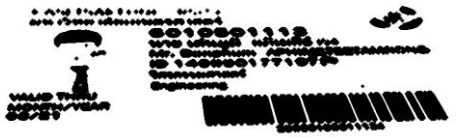
```
kernel =
cv2.getStructuringElement(cv2.MORPH_CRO
SS,(5,5))

bw_erode =
cv2.erode(bw,kernel,iterations = 2)
```

แนวทางการแก้ปัญหา

Eroded Binary image

Floodfill image



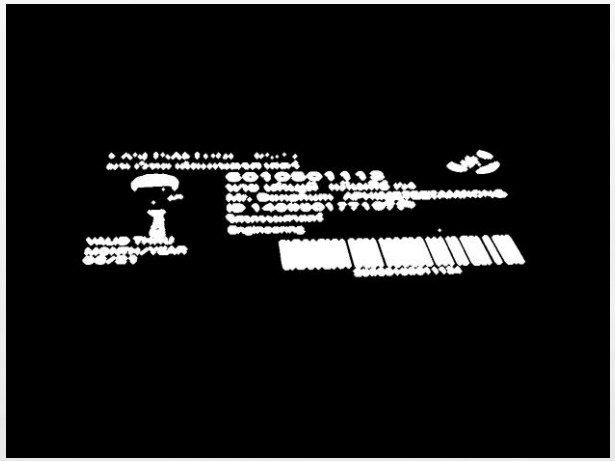
```
im_floodfill = bw_erode.copy()

cv2.floodFill(im_floodfill, None,
(0,0), 255)
```

แนวทางการแก้ปัญหา

Floodfill image

Invert Floodfill image



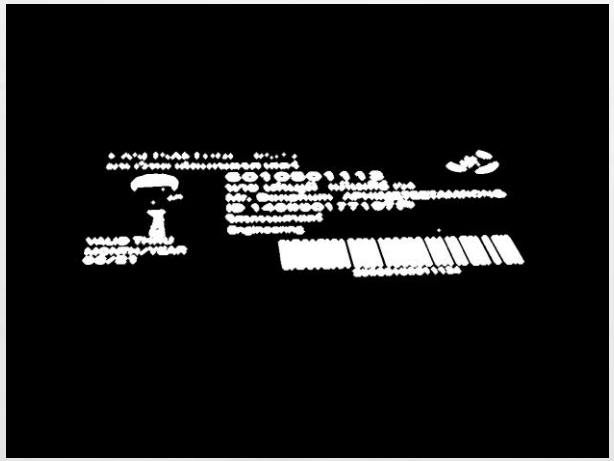
```
im_floodfill_inv =
cv2.bitwise_not(im_floodfill)
```

แนวทางการแก้ปัญหา

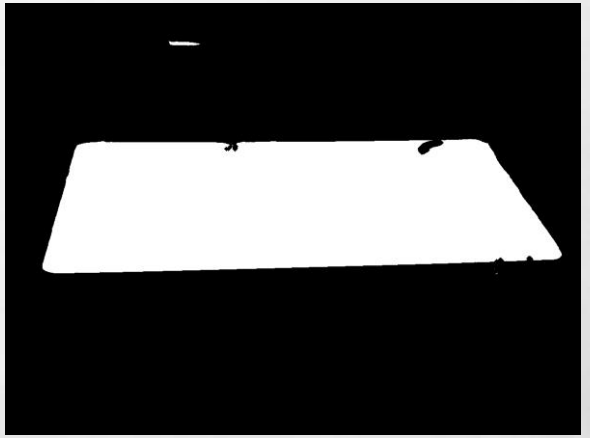
Invert Floodfill image

Eroded Binary image

```
im_out = bw_erode | im_floodfill_inv
```



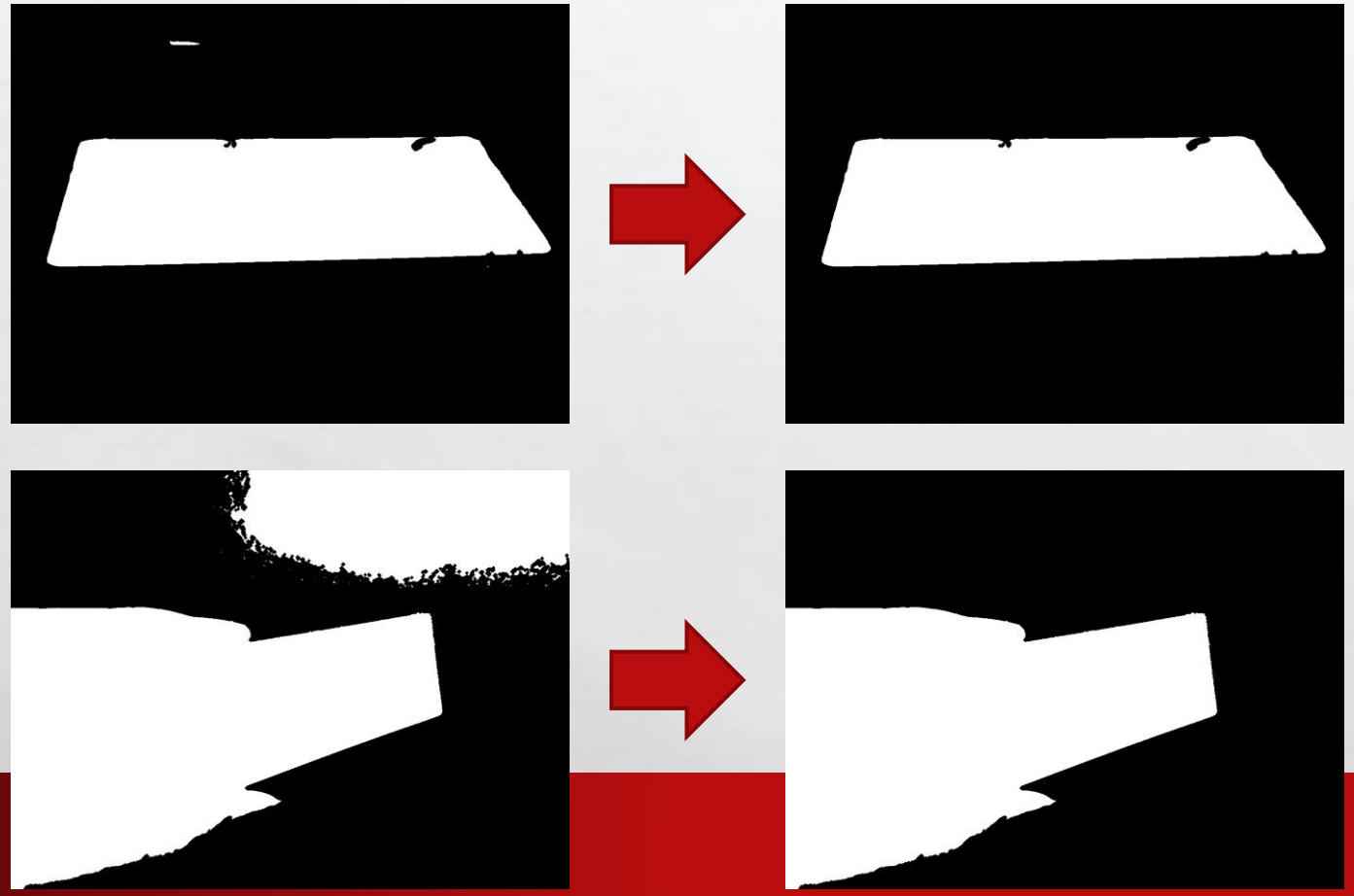
OR



OR

แนวทางการแก้ปัญหา

Find Biggest object

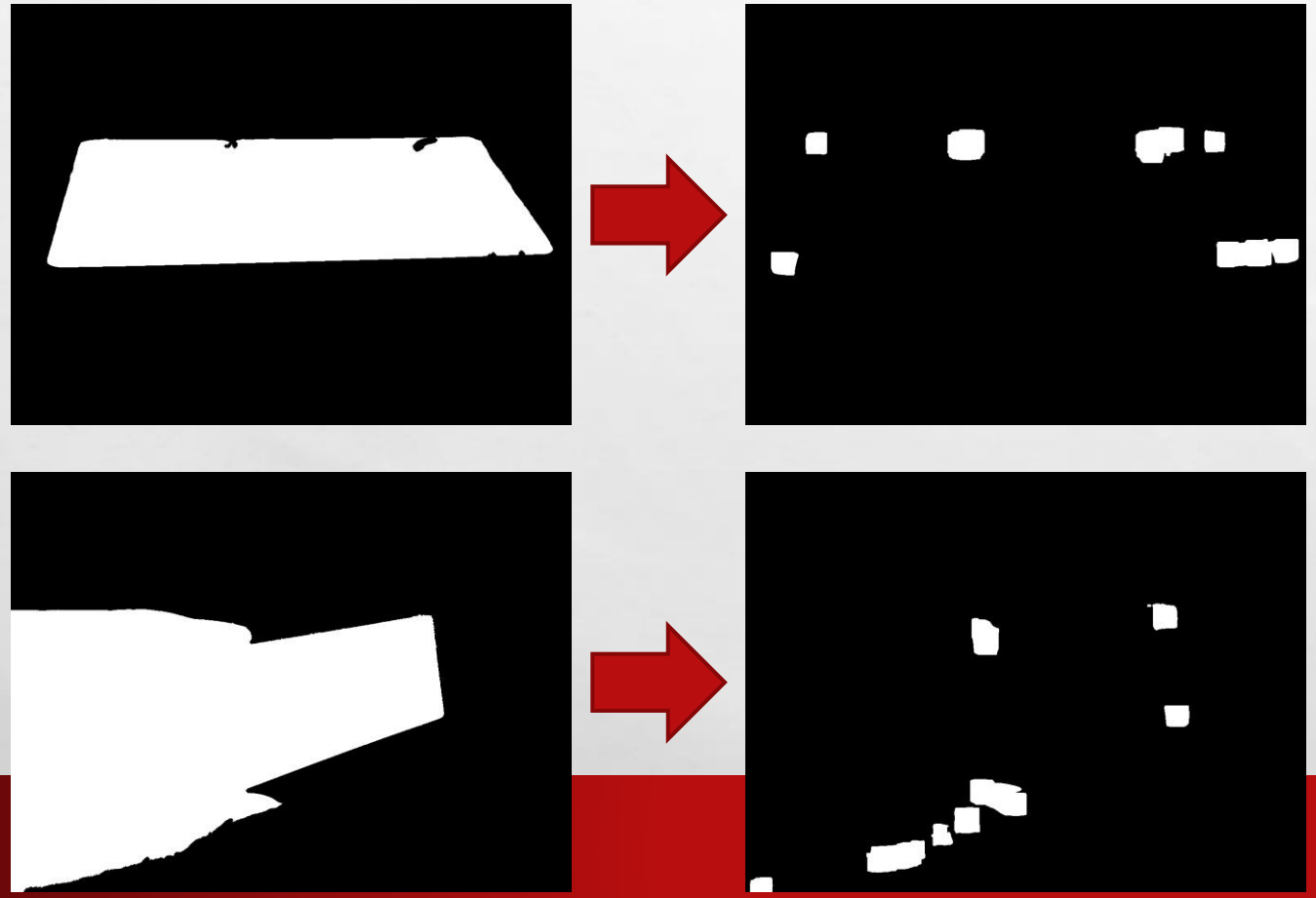


```
def find_biggest(img):
    img_copy = img.copy()
    img_ = img.copy()
    white_obj = {}
    i_object = 0
    corner = []
    max = 0
    while sum(sum(img_copy)) != 0:
        white_y, white_x = np.where(img_copy ==
255)
        cv2.floodFill(img_, None, (white_x[0],
white_y[0]), 0)
        temp = img_copy - img_
        white_obj[i_object] = temp
        i_object = i_object + 1
        cv2.floodFill(img_copy, None, (white_x[0],
white_y[0]), 0)
    for i in range(len(white_obj)):
        white_y, white_x = np.where(white_obj[i]
== 255)
        if np.size(white_y) > max:
            max = np.size(white_y)
            biggest = white_obj[i]
    return biggest
```

แนวทางการแก้ปัญหา

Biggest object

Corner Harris Detection



```
dst = cv2.cornerHarris(im_out_biggest,50,15,0.005)

corner_obj = np.zeros(bw_erode.shape[:2])
corner_obj[dst>0.005*dst.max()]=[255]
corner_obj = corner_obj.astype(np.uint8)
```

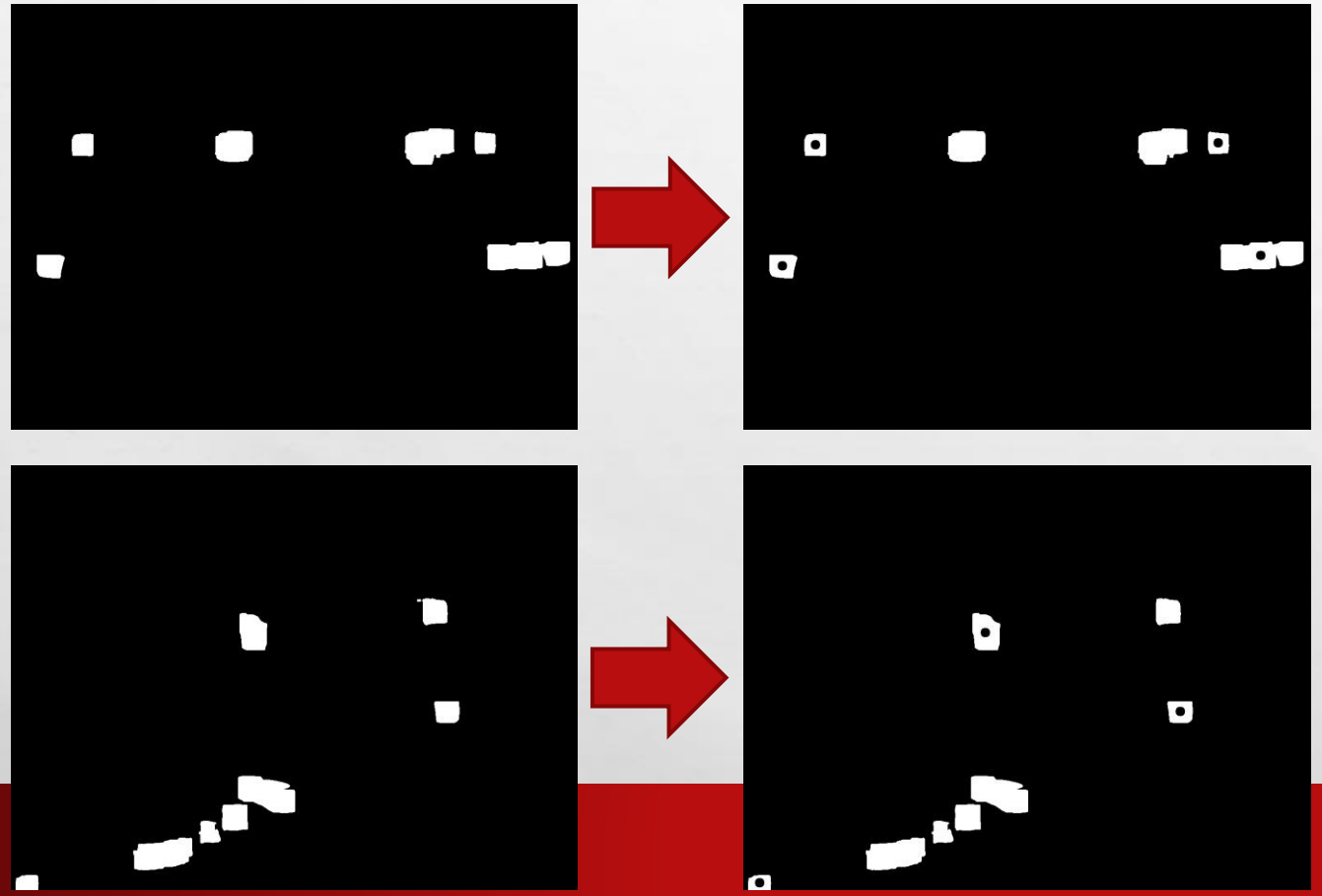
```
def find_object(img):
    img_copy = img.copy()
    img_ = img.copy()
    white_obj = {}
    i_object = 0
    object = []
    while sum(sum(img_copy)) != 0:
        white_y,white_x = np.where(img_copy == 255)
        cv2.floodFill(img_, None, (white_x[0],
white_y[0]), 0)
        temp = img_copy - img_
        white_obj[i_object] = temp
        i_object = i_object + 1

cv2.floodFill(img_copy,None,(white_x[0],white_y[0]),
0)
    for i in range(len(white_obj)):
        white_y, white_x = np.where(white_obj[i] ==
255)
        white_y = int(np.mean(white_y))
        white_x = int(np.mean(white_x))
        object.append((white_x,white_y))
    return np.array(object)
```

แนวทางการแก้ปัญหา

Corner Harris Detection

Coordinate Point

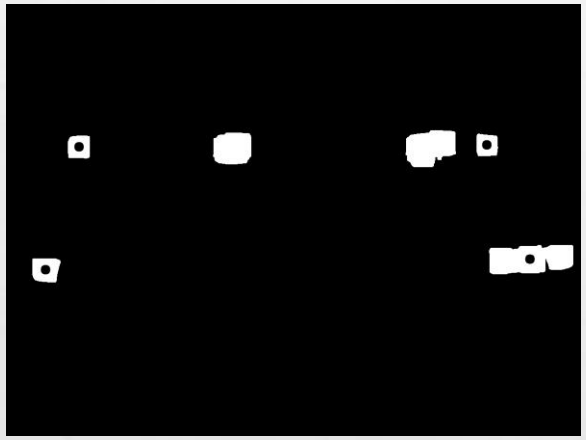


```
def area(coordinate):
    coordinate =
np.append(coordinate,coordinate[[0]],axis=0)
    area = 0.5*abs((sum(coordinate[:-1,0] *
coordinate[1:,1]) - sum(coordinate[:-1,1] *
                        coordinate[1:,0])))
    return area
```

```
def max_area(corner):
    perm = np.array(list(permutations(corner, 4)))
    max = 0
    for i in perm :
        area_of_rect = area(i)
        if area_of_rect > max:
            output = i
            max = area_of_rect
    return output
```

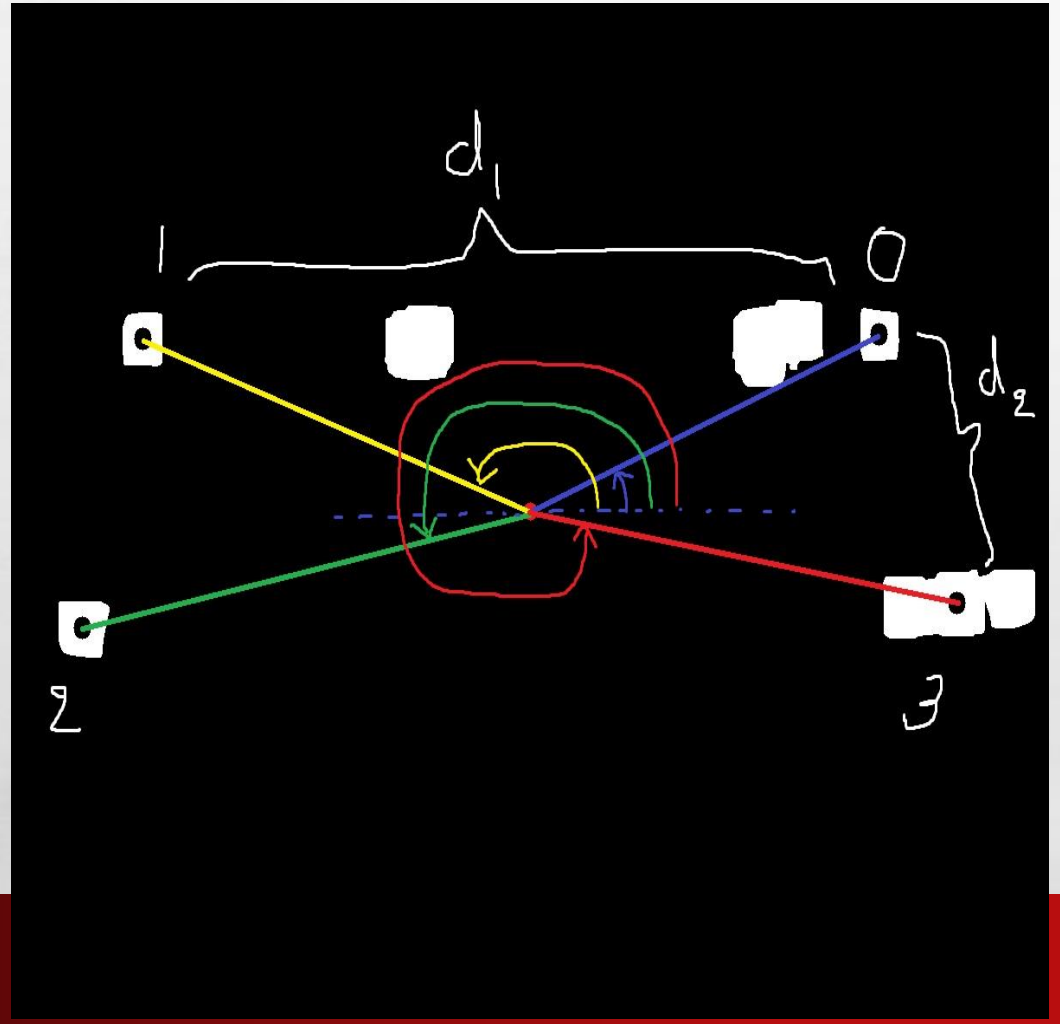
แนวทางการแก้ปัญหา

Coordinate Point



```
def assign_corner(corner,img):
    h, w = img.shape[:2]
    center = [int(w/2),int(h/2)]
    deg = []
    for l in corner:
        c = np.arctan2(center[1]-l[1],l[0]-center[0])
        if c>0:
            deg.append(c*(180/np.pi))
        else:
            deg.append(c * (180 / np.pi)+360)
    deg = np.array([deg])
    deg = np.transpose(deg)
    corner = np.concatenate((corner, deg), axis=1)
    corner = corner[corner[:,2].argsort()]
    corner = np.delete(corner,2,1)
    condition1 = np.sqrt((corner[1][0]-corner[0][0])**2 +
(corner[1][1]-corner[0][1])**2)
    condition2 = np.sqrt((corner[3][0]-corner[0][0])**2 +
(corner[3][1]-corner[0][1])**2)
    if condition2>condition1:
        corner = corner[[1,2,3,0],:]
    return corner
```

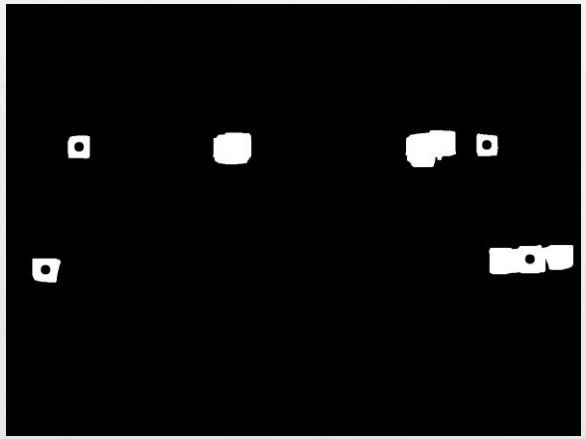
แนวทางการแก้ปัญหา

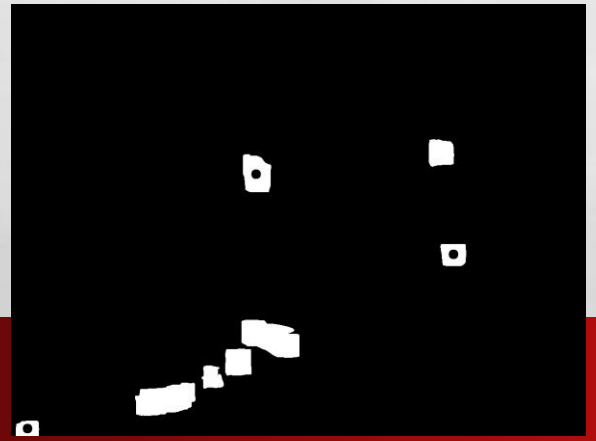



```
def assign_corner(corner,img):
    h, w = img.shape[:2]
    center = [int(w/2),int(h/2)]
    deg = []
    for l in corner:
        c = np.arctan2(center[1]-l[1],l[0]-center[0])
        if c>0:
            deg.append(c*(180/np.pi))
        else:
            deg.append(c * (180 / np.pi)+360)
    deg = np.array([deg])
    deg = np.transpose(deg)
    corner = np.concatenate((corner, deg), axis=1)
    corner = corner[corner[:,2].argsort()]
    corner = np.delete(corner,2,1)
    condition1 = np.sqrt((corner[1][0]-corner[0][0])**2 +
(corner[1][1]-corner[0][1])**2)
    condition2 = np.sqrt((corner[3][0]-corner[0][0])**2 +
(corner[3][1]-corner[0][1])**2)
    if condition2>condition1:
        corner = corner[[1,2,3,0],:]
    return corner
```

แนวทางการแก้ปัญหา

Coordinate Point





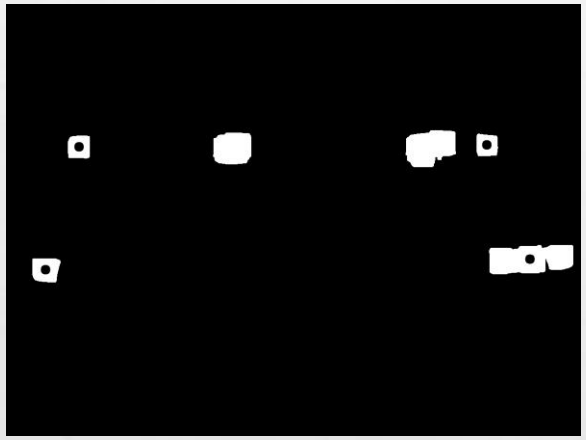
Output image

หา Transfer จาก pts1(มุมของรูป input) และ pts2 และ ใช้ warpPerspective เพื่อบิดรูปกลับคืน

```python
def transform(src,dst):
    a = np.zeros([8,8])
    b = np.zeros([8,1])
    for i in range(0,4):
        a[i][0] = a[i + 4][3] = src[i][0]
        a[i][1] = a[i + 4][4] = src[i][1]
        a[i][2] = a[i + 4][5] = 1
        a[i][6] = -src[i][0] * dst[i][0]
        a[i][7] = -src[i][1] * dst[i][0];
        a[i + 4][6] = -src[i][0] * dst[i][1]
        a[i + 4][7] = -src[i][1] * dst[i][1]
        b[i] = dst[i][0]
        b[i + 4] = dst[i][1]
    x = np.linalg.solve(a, b)
    x = np.append(x,1)
    x = x.reshape(3,3)
    return x
```

แนวทางการแก้ปัญหา

Coordinate Point



Output image

หา Transfer จาก pts1(มุมของรูป input) และ pts2 และ ใช้ warpPerspective เพื่อบิดรูปกลับคืน

```
pts1 =
np.float32(lib.assign_corner(corner_4,
img))

pts2 =
np.float32([[w,0],[0,0],[0,h],[w,h]])

matrix = lib.transform(pts1,pts2)
output = cv2.warpPerspective(img,
matrix, (w,h))
```

ผลการทดลอง

จากการทดลองกับรูปทั้งหมด 35 รูป

จำนวนภาพที่ให้ OUTPUT ออกมา และ ถูกต้อง (TRUE POSITIVE) = 12 รูป

จำนวนภาพที่ให้ OUTPUT ออกมา แต่ ไม่ถูกต้อง (FALSE POSITIVE) = 4 รูป

จำนวนภาพที่ไม่ให้ OUTPUT ออกมา เพราะเกิด ERROR (FALSE NEGATIVE) = 19 รูป

RECALL = TP/(TP+FN) = 38.71%

PRECISION = TP/(TP+FP) = 75%

สรุปผลการทดลอง

จากการทดลองผลที่ได้ คือ ยังมีปัญหาที่ทำให้เกิด ERROR อยู่จึงทำให้บางภาพไม่สามารถให้ OUTPUT ออกมาได้ จึงทำได้ RECALL และ PRECISON ที่ได้ไม่มากนัก

แนวทางการแก้ไข คือ ถ้าแก้ไขในส่วน PREPROCESSING ก่อนที่จะทำการหามุม CORNER DETECTION ให้ดีขึ้นได้ก็จะมีผลที่ดีมากขึ้น