2021 Fall
Robotics HW3 – Robot Vision

Team Members and Division of Work

Part A: R08921109 高逵 & R10921008 朱雁丞

Part B: R08921101 杜盛道 & R10921115 周之蕙

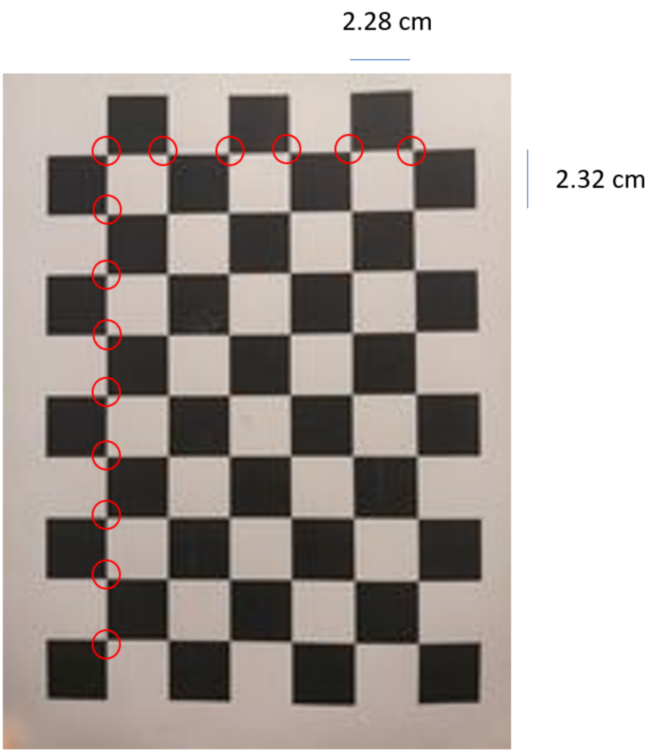
Part A: Camera Calibration

The camera we use:

The rear camera of Samsung Galaxy J7+ (SM-C710F/DS)

The checkerboard we use:

With 9*6 inner corners like following picture:



The calibration process:

Please refer to hw3_a.py for complete calibration code.

Brief explanation of the code is given below:

1. `cv.findChessboardCorners( image, patternSize[, corners[, flags]] ) -> retval, corners`

   is used to find the corner on chessboard in the image.

2. `cv.cornerSubPix( image, corners, winSize, zeroZone, criteria ) -> corners`

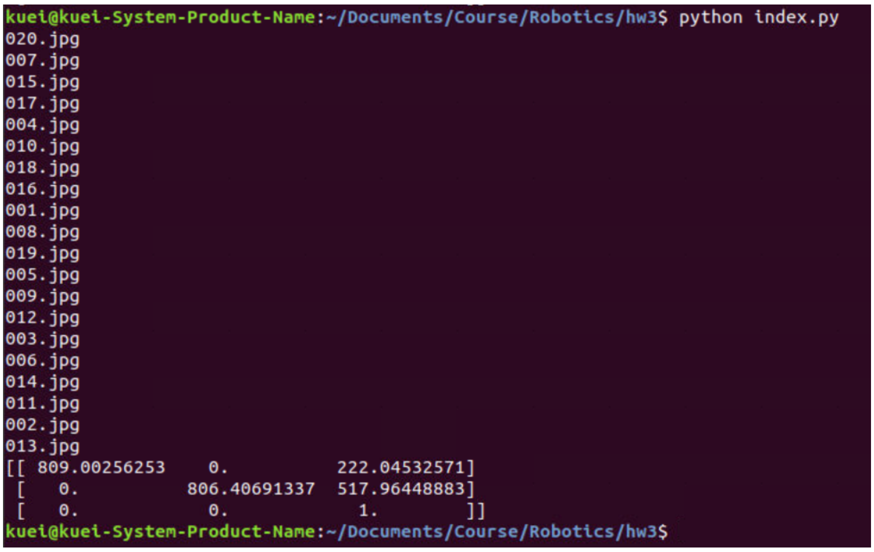
   is used to increase the accuracy of the above result.

3. `cv.calibrateCamera( objectPoints, imagePoints, imageSize, cameraMatrix, distCoeffs[, rvecs[, tvecs[, flags[, criteria]]]]`

is used to calculate intrinsic parameters, extrinsic parameters, distortion coefficients based on the above corner result

The Intrinsic Parameters:

The calibration result shows that the camera matrix of Samsung Galaxy J7+ is:



```
kuei@kuei-System-Product-Name:~/Documents/Course/Robotics/hw3$ python index.py
020.jpg
007.jpg
015.jpg
017.jpg
004.jpg
010.jpg
018.jpg
016.jpg
001.jpg
008.jpg
019.jpg
005.jpg
009.jpg
012.jpg
003.jpg
006.jpg
014.jpg
011.jpg
002.jpg
013.jpg
[[ 809.00256253    0.          222.04532571]
 [   0.          806.40691337  517.96448883]
 [   0.            0.            1.        ]]
kuei@kuei-System-Product-Name:~/Documents/Course/Robotics/hw3$
```

[redacted]

The Intrinsic Parameters are thus:

$$\left(f_x, f_y\right) = (809.00, 806.41)$$

$$\left(c_x, c_y\right) = (222.05, 517.97)$$

where each parameter is defined as following [1][2]:

Intrinsic parameters	Unit	Definition
f_x	Pixel	Focal length multiplied by conversion parameters between length and pixel on the x direction
f_y	Pixel	Focal length multiplied by conversion parameters between length and pixel on the y direction
c_x	Pixel	X Coordinate of the image center with respect to the bottom left of the image
c_y	pixel	Y Coordinate of the image center with respect to the bottom left of the image

Undistort the pictures:

The following command is used to undistort the original picture.

```
cv.undistort( src, cameraMatrix, distCoeffs[, dst[, newCameraMatrix]] ) -> dst
```

The effect is to mitigate the radial distortion and the tangent distortion of the original picture.

The 21 pictures:

The original 21 pictures and their undistorted version are list below.

Among them, the pictures No. 1~20 are used for calibration.

Picture No.	Original Picture	Undistorted Version
1		
2		

3		
4		
5		

6		
7		
8		

9		
10		
11		

12		
13		
14		

15		
16		
17		

18		
19		
20		

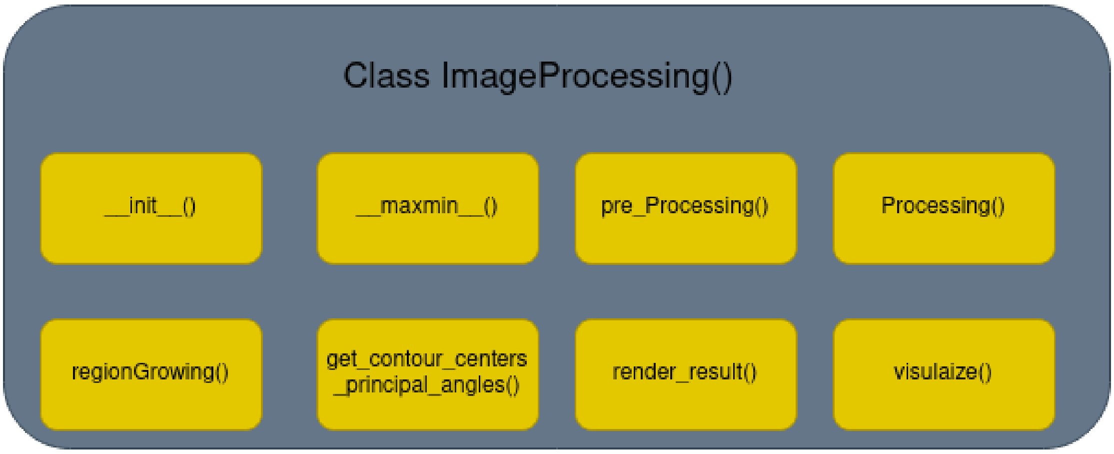
21		

Part B: Object Detection

Process:

Step 1: Image pre-processing
Step 2: Binary Labeling using Region Growing Algorithm
Step 3: Find contour
Step 4: Get centroids and principal angles
Step 5: render result to result image

Function overview:

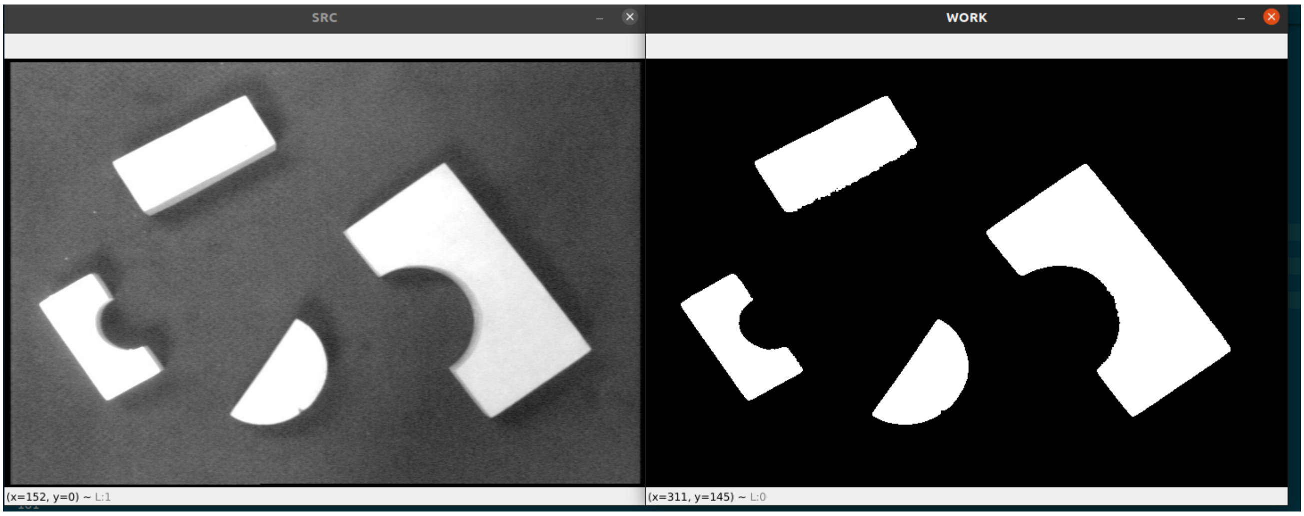


Implement:

Step 1: Image pre-processing

Key Function

```
ret,self.workIMG =
cv.threshold(self.workIMG,180,255,cv.THRESH_BINARY)
cv.erode(self.workIMG,self.kernel,self.workIMG)
cv.dilate(self.workIMG,self.kernel,self.workIMG)
```



Step 2:Binary Labeling using Region Growing Algorithm

1. init row = 1, col = 1
2. access each element I(row,col), if I(row,col) = 255
    a. i = i + 1
    b. call region growing algorithm
    c. if not all element has been travialied, goto step 2

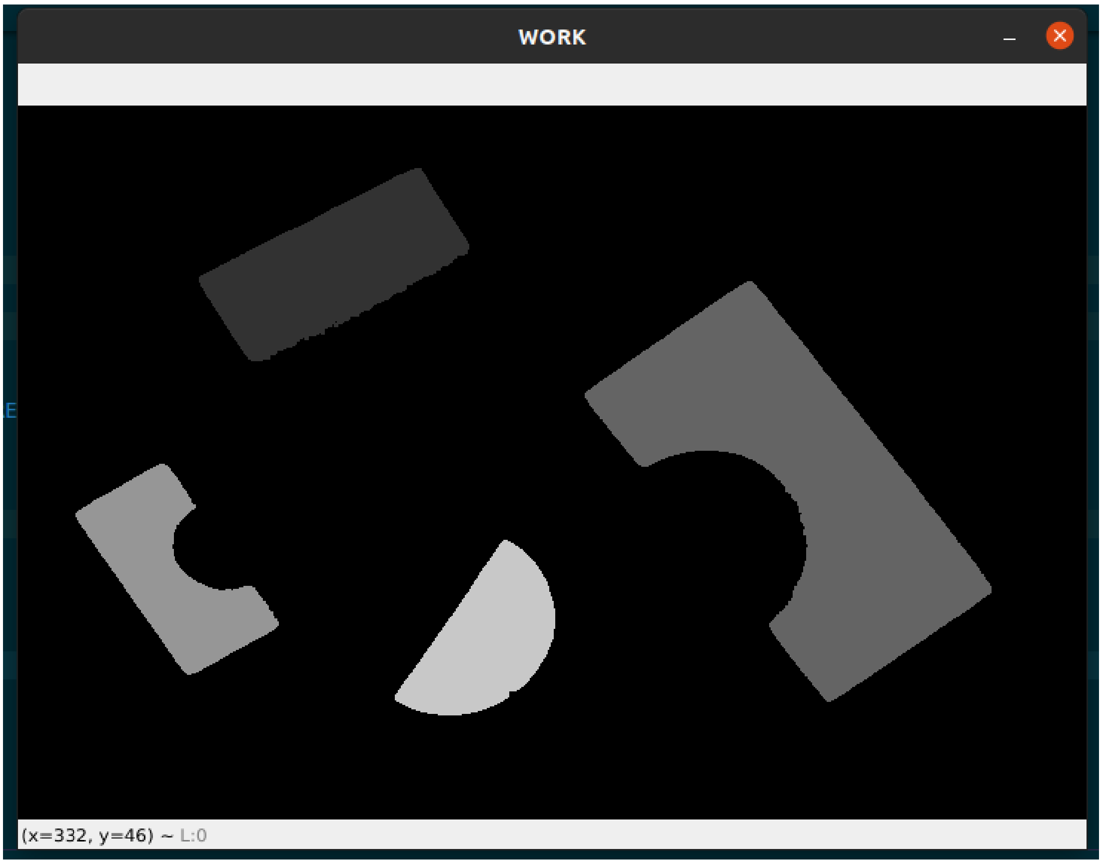
Region growing algorithm:

1. set I(k,j) = i, push (k,j) push(0,0)
2. if(j<n) and I(k,j+1) = 255
    a. I(k,j+1) = i
    b. push (K,j+1)
3. if(k>1) and I(K-1,j) = 255
    a. I(K-1,j) = i
    b. push(K-1,j)
4. if(j>1) and I(k,j-1) = 255
    a. I(k,j-1) = 1
    b. push (k,j-1)
5. if(k<m) and I(K+1,j) = 255
    a. I(K+,j) = i
    b. push(K+1,j)
6. k,j = pop(), if (k,j)!= (0,0), goto step 2
7. pop()

```
pixel = 0
for row in range(self.workIMG.shape[0]):
    for col in range(self.workIMG.shape[1]):
        if(self.workIMG[row,col] == 255):
            # print(row,col)
            pixel = pixel + 1
            self.regionGrowing(row,col,pixel)
```

```python
def regionGrowing(self,k,j,i):
    # k: ROW
    # j: COL
    # m: ROW number
    # n: COL number
    # i: pixel
    #init parameters
    points = []
    m = self.workIMG.shape[0]
    n = self.workIMG.shape[1]

    #step 1
    self.workIMG[k,j] = i
    points.append([k,j])
    points.append([0,0])

    #step 2
    #Kernel Right
    if (j<n) and self.workIMG[k,j+1] == 255:
        self.workIMG[k,j+1] = i
        points.append([k,j+1])

    #step 3
    #Kernel Up
    if (k>1) and self.workIMG[k-1,j] == 255:
        self.workIMG[k-1,j] = i
        points.append([k-1,j])

    #step 4
    #Kernel Left
    if (j>1) and self.workIMG[k,j-1] == 255:
        self.workIMG[k,j-1] = i
        points.append([k,j-1])

    #step 5
    #Kernel Down
    if (k<m) and self.workIMG[k+1,j] == 255:
        self.workIMG[k+1,j] = i
        points.append([k+1,j])
```

```python
    #Step 6
    while(True):
        # print(len(points))
        k,j = points.pop()
        if(k == 0 and j == 0):
            points.pop()
            break
        #step 2
        #Kernel Right
        if (j<n) and self.workIMG[k,j+1] == 255:
            self.workIMG[k,j+1] = i
            points.append([k,j+1])

        #step 3
        #Kernel Up
        if (k>1) and self.workIMG[k-1,j] == 255:
            self.workIMG[k-1,j] = i
            points.append([k-1,j])

        #step 4
        #Kernel Left
        if (j>1) and self.workIMG[k,j-1] == 255:
            self.workIMG[k,j-1] = i
            points.append([k,j-1])

        #step 5
        #Kernel Down
        if (k<m) and self.workIMG[k+1,j] == 255:
            self.workIMG[k+1,j] = i
            points.append([k+1,j])
```



After segmentation, and rendered for visulaization

Step 3: Find contour

key function:

```
self.contour,_ =
cv.findContours(self.workIMG,cv.RETR_EXTERNAL,cv.CHAIN_APPROX_SIM
PLE)
```

Step 4: Get centroids and principal angles

1. Calculate image moments
2. Calculate centroid from image moments
3. Calculate principal angle

Code(part)

```
        for i, c in enumerate(contours):
            self.M = cv.moments(c)
            # print(M)

            center = (int(self.M["m10"] / self.M["m00"]),
int(self.M["m01"] / self.M["m00"]))

            principal_angles[i] =
(math.atan2(2*self.M['mu11'],
self.M['mu02']-self.M['mu20'])/2)*180/math.pi
```

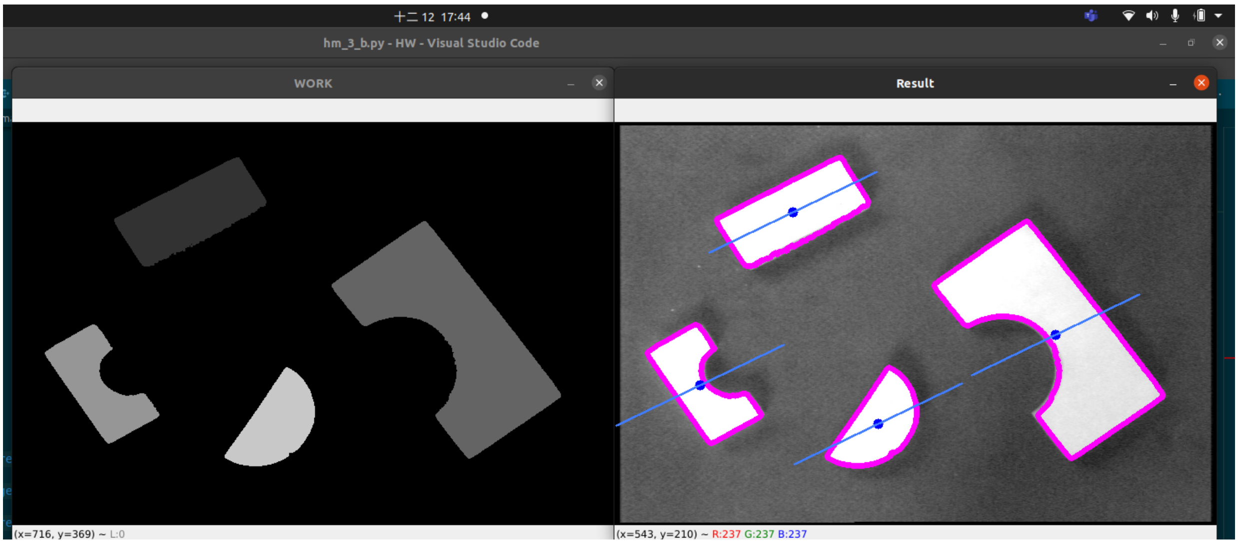
```
            centers[i] = center
```

Step 5:render result to result image
In this step, we draw line which penetrate the principal center,
The default offset pixel is 100,
p1 = (x,y) - (100,100*tan(atan2(mu11,mu20-mu02)/2))
p1 = (x,y) + (100,100*tan(atan2(mu11,mu20-mu02)/2))

```
    def render_result(self,line_length = 100):

        for i in range(len(self.contour)):

cv.drawContours(self.resultIMG,self.contour,i,(255,0,255),5)

cv.circle(self.resultIMG,(self.centers[i][0],self.centers[i][1]),
2,(255,0,0),8)
            # print(self.centers[i][0])
            cv.line(self.resultIMG,
            (self.centers[i][0] -line_length , self.centers[i][1]
- int(line_length * math.tan(math.atan2(2*self.M['mu11'],
self.M['mu20']-self.M['mu02'])/2))),
            (self.centers[i][0] +line_length , self.centers[i][1]
+ int(line_length * math.tan(math.atan2(2*self.M['mu11'],
self.M['mu20']-self.M['mu02'])/2))),
            (255,125,64),2)
            #
cv.line(self.resultIMG,(100,200),(200,300),(255,255,0),10)
        for row in range(self.workIMG.shape[0]):
            for col in range(self.workIMG.shape[1]):
                self.workIMG[row,col] = self.workIMG[row,col] * 50
```

```
robot@robot-GS75-Stealth-10SE:~/Desktop/HW$ /bin/python3 /home/robot/Desktop/HW/hm_3_b.py ./images/er7-4.jpg
Reading File: /home/robot/Desktop/HW/images/er7-4.jpg
Centers [[315 359]
 [102 313]
 [527 253]
 [213 107]]
principal Angles [[-37.08449929]
 [ 36.56504275]
 [ 39.17740366]
 [-64.0709341 ]]
robot@robot-GS75-Stealth-10SE:~/Desktop/HW$
```

The result are:
Input cmd: /bin/python3 /home/robot/Desktop/HW/hm_3_b.py ./images/er7-4.jpg
output:
Reading File: /home/robot/Desktop/HW/images/er7-4.jpg
Centers [[315 359]
 [102 313]
 [527 253]
 [213 107]]
principal Angles [[-37.08449929]
 [36.56504275]
 [39.17740366]
 [-64.0709341]]



The code also can be accessed here:
https://github.com/Runnlion/NTU-Robotic-2021-Fall/tree/main/Team2/HM3

Reference:
[1] 一文帶你搞懂相機内參外參(Intrinsics & Extrinsics)
https://zhuanlan.zhihu.com/p/389653208
[2] Camera Calibration相機校正
https://medium.com/image-processing-and-ml-note/camera-calibration%E7%9B%B8%E6%A9%9F%E6%A0%A1%E6%AD%A3-1d94ffa7cbb4
[3] OpenCV
https://opencv.org/