2.2 X:509-692 Y：666-432

2.3 X:539-730 Y: 635-397

2.8 X:513-721 Y:292-597

2.14(wai) (632 273) (825 351) (696 607) (522 517) fq

2\_3.1 X(186 431) Y(380 706) X(928 1101) (296 511)

3.1 X(612 758) Y(356 643) ?? 356-456

3.2 X 535-654 Y 424 591

3.4 X 628- 771 Y 427 624

3.10 X 550 - 665 Y 403 -565 （歪的）

4.1 X 564-657 Y 509-660

4.3 X 347-447 Y 342-473

4.5 X 544- 634 Y 433-567

4.7 X 548-646 Y 378-509

4.10 X 584-679 Y 170-307

5.1 X 570-650 Y 380-498

5.2 X 590-670 Y 391-509

5.9 X 723-805 Y 452-563

5.11 X 612-690 Y 441-557

6.5 X 606-670 Y 357-448

7.2 X 621-683 Y 353-437

7.3 X 953-1019 Y 246-330

8.2 X 599-646 Y 397-471

8.4 X 659-707 Y 305-382

9.2 X 617-652 Y 633-694

10.4 X 610-650 Y 406-462

14 X 599-628 Y 191-231

2.6 X 376-619 Y 307-637

3.8 X 535-672 Y 145-353

3.11 X 559-694 Y 309-506

4.2 X 763-869 Y 235-393

4.4 X 564-659 Y 385-515

4.6 X 725-831 Y 397-553

4.8 X (550,324)-(663,345) Y (550,324)-(534,483)

5.3 X 621-701 Y 431-546

5.4 X 634-723 Y 456-576

5.5 X 575-654 Y 460-570

5.6 X 730-809 Y 433-544

#varYchannelR = np.dot((np.transpose(YchannelR) - meanYchannelR),(YchannelR-meanYchannelR))

#varCrchannelR = np.dot((np.transpose(CrchannelR) - meanCrchannelR),(CrchannelR-meanCrchannelR))

#varCbchannelR = np.dot((np.transpose(CbchannelR) - meanCbchannelR),(CbchannelR-meanCbchannelR))

#varYchannelnR = np.dot((np.transpose(YchannelnR) - meanYchannelnR),(YchannelnR-meanYchannelnR))

#varCrchannelnR = np.dot((np.transpose(CrchannelnR) - meanCrchannelnR),(CrchannelnR-meanCrchannelnR))

#varCbchannelnR = np.dot((np.transpose(CbchannelnR) - meanCbchannelnR),(CbchannelnR-meanCbchannelnR))

#varYchannelY = np.dot((np.transpose(YchannelY) - meanYchannelY),(YchannelY-meanYchannelY))

#varCrchannelY = np.dot((np.transpose(CrchannelY) - meanCrchannelY),(CrchannelY-meanCrchannelY))

#varCbchannelY = np.dot((np.transpose(CbchannelY) - meanCbchannelY),(CbchannelY-meanCbchannelY))

#varYchannelBr = np.dot((np.transpose(YchannelBr) - meanYchannelBr),(YchannelBr-meanYchannelBr))

#varCrchannelBr = np.dot((np.transpose(CrchannelBr) - meanCrchannelBr),(CrchannelBr-meanCrchannelBr))

#varCbchannelBr = np.dot((np.transpose(CbchannelBr) - meanCbchannelBr),(CbchannelBr-meanCbchannelBr))

#probR = scipy.stats.norm(meanYchannelR, varYchannelR).pdf(newimg[i][j][0]) \* scipy.stats.norm(meanCrchannelR, varCrchannelR).pdf(newimg[i][j][1]) \* scipy.stats.norm(meanCbchannelR, varCbchannelR).pdf(newimg[i][j][2]) \* 1/4

#probnR = scipy.stats.norm(meanYchannelnR, varYchannelnR).pdf(newimg[i][j][0]) \* scipy.stats.norm(meanCrchannelnR, varCrchannelnR).pdf(newimg[i][j][1]) \* scipy.stats.norm(meanCbchannelnR, varCbchannelnR).pdf(newimg[i][j][2]) \* 1/4

#probY = scipy.stats.norm(meanYchannelY, varYchannelY).pdf(newimg[i][j][0]) \* scipy.stats.norm(meanCrchannelY, varCrchannelY).pdf(newimg[i][j][1]) \* scipy.stats.norm(meanCbchannelY, varCbchannelY).pdf(newimg[i][j][2]) \* 1/4

#probBr = scipy.stats.norm(meanYchannelBr, varYchannelBr).pdf(newimg[i][j][0]) \* scipy.stats.norm(meanCrchannelBr, varCrchannelBr).pdf(newimg[i][j][1]) \* scipy.stats.norm(meanCbchannelBr, varCbchannelBr).pdf(newimg[i][j][2]) \* ¼

#for a in xR:

# for b in yR:

# YchannelR.append(img2[a,b,0])

# CrchannelR.append(img2[a,b,1])

# CbchannelR.append(img2[a,b,2])

#for c in xnR:

# for d in ynR:

# YchannelnR.append(img2[c,d,0])

# CrchannelnR.append(img2[c,d,1])

# CbchannelnR.append(img2[c,d,2])

#for e in xY:

# for f in yY:

# YchannelY.append(img2[e,f,0])

# CrchannelY.append(img2[e,f,1])

# CbchannelY.append(img2[e,f,2])

#for g in xBr:

# for h in yBr:

# YchannelBr.append(img2[g,h,0])

# CrchannelBr.append(img2[g,h,1])

# CbchannelBr.append(img2[g,h,2])

#pixel\_values = np.array(pixel\_values).reshape((width, height, 3))

#print(pixel\_values.size)

#x, y = xR[1], yR[1]

#print(list(x),list(y))

#b = pixel\_values[x,y,0]

i = 0

cpt = 0

while i < colormap.shape[0]:

j = 0

while j < colormap.shape[1]:

if colormap[i][j] == 0:

cpt += 1

j += 1

i += 1

print(cpt)

i = 0

cpt = 0

while i < binarymap.shape[0]:

j = 0

while j < binarymap.shape[1]:

if binarymap[i][j] == 1:

cpt += 1

j += 1

i += 1

print(cpt)

print(detR)

print(detnR)

print(detY)

print(detBr)

regions = regionprops(newbinary)

for props in regions:

x0,y0 = props.centroid

points = 0

Vxx = 0

Vyy = 0

Vxy = 0

while points < len(newcontour[0]):

Vxx += np.square(x0-newcontour[0][points][0][0])

Vyy += np.square(y0-newcontour[0][points][0][1])

Vxy += (x0-newcontour[0][points][0][0])\*(y0-newcontour[0][points][0][1])

points += 1

avgVxx = Vxx/len(newcontour[0])

avgVyy = Vyy/len(newcontour[0])

avgVxy = Vxy/len(newcontour[0])

Vmatrix = [[avgVxx,avgVxy],[avgVxy,avgVyy]]

eigenvalue,\_ = np.linalg.eig(Vmatrix)

print(eigenvalue)

while numcontour < newarray.shape[0]:

newcontour = newarray[numcontour]

newbinary = cv2.fillPoly(binarymatrix,newcontour,255)

numcontour += 1

regions = regionprops(newbinary)

for props in regions:

x0,y0 = props.centroid

points = 0

Vxx = 0

Vyy = 0

Vxy = 0

while points < len(newcontour):

Vxx += np.square(x0-newcontour[points][0][0])

Vyy += np.square(y0-newcontour[points][0][1])

Vxy += (x0-newcontour[points][0][0])\*(y0-newcontour[points][0][1])

points += 1

avgVxx = Vxx/len(newcontour)

avgVyy = Vyy/len(newcontour)

avgVxy = Vxy/len(newcontour)

Vmatrix = [[avgVxx,avgVxy],[avgVxy,avgVyy]]

eigenvalue,\_ = np.linalg.eig(Vmatrix)

print(eigenvalue)