实验一

import cv2  
import numpy as np  
import matplotlib.pyplot as plt  
  
# 读取原始图像  
image = cv2.imread(r'C:\Users\43215\Desktop\image.jpg')  
  
# 转换为灰度图  
gray\_image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)  
  
# 直方图均衡化  
equalized\_image = cv2.equalizeHist(gray\_image)  
  
# 绘制原图、灰度图和均衡化结果图  
plt.figure(figsize=(12, 4))  
plt.subplot(131)  
plt.title('Original Image')  
plt.imshow(cv2.cvtColor(image, cv2.COLOR\_BGR2RGB))  
plt.axis('off')  
  
plt.subplot(132)  
plt.title('Grayscale Image')  
plt.imshow(gray\_image, cmap='gray')  
plt.axis('off')  
  
plt.subplot(133)  
plt.title('Equalized Image')  
plt.imshow(equalized\_image, cmap='gray')  
plt.axis('off')  
  
# 保存均衡化结果图像  
cv2.imwrite(r'C:\Users\43215\Desktop\compare.jpg', equalized\_image)  
  
# 绘制原图和均衡化结果的直方图  
plt.figure(figsize=(12, 4))  
plt.subplot(121)  
plt.title('Original Histogram')  
plt.hist(gray\_image.ravel(), 256, [0, 256])  
  
plt.subplot(122)  
plt.title('Equalized Histogram')  
plt.hist(equalized\_image.ravel(), 256, [0, 256])  
  
# 保存原图和均衡化结果的直方图  
cv2.imwrite(r'C:\Users\43215\Desktop\before\_after.jpg', equalized\_image)  
  
plt.show()

import cv2  
import numpy as np  
import math  
  
# 读取图像  
img = cv2.imread(r'C:\Users\43215\Desktop\rmb.jpg')  
  
# 转换为灰度图  
gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  
  
# 进行边缘检测，使用Canny算子  
edges = cv2.Canny(gray, 230, 240, apertureSize=3)  
  
# 使用Hough直线变换检测直线  
lines = cv2.HoughLines(edges, 1, np.pi / 90, threshold=100)  
# 存储交点  
intersections = []  
  
# 找到每对直线的交点  
if lines is not None:  
 for i in range(len(lines)):  
 for j in range(i + 1, len(lines)):  
 rho1, theta1 = lines[i][0]  
 rho2, theta2 = lines[j][0]  
  
 # 计算角度差异  
 angle\_diff = np.abs(theta1 - theta2)  
   
 # 设置角度差异的阈值，排除共线情况  
 angle\_threshold = np.pi / 18 # 10度  
   
 if angle\_diff > angle\_threshold:  
 A = np.array([[np.cos(theta1), np.sin(theta1)],[np.cos(theta2), np.sin(theta2)]])  
 B = np.array([rho1, rho2])  
 intersection = np.linalg.solve(A, B)  
  
 if 0 <= intersection[0] < img.shape[1] and 0 <= intersection[1] < img.shape[0]:  
 intersections.append((int(intersection[0]), int(intersection[1])))  
  
# 绘制相邻交点之间的线段  
# 绘制矩形，排除绘制对角线  
if len(intersections) == 4:  
 for i in range(2):  
 cv2.line(img, intersections[i], intersections[i + 2], (0, 0, 255), 2)  
 cv2.line(img, intersections[0], intersections[1], (0, 0, 255), 2)  
 cv2.line(img, intersections[2], intersections[3], (0, 0, 255), 2)  
 x1, y1 = intersections[0]  
 x2, y2 = intersections[2]  
 value1 = math.sqrt((x2 - x1) \*\* 2 + (y2 - y1) \*\* 2)  
   
 x1, y1 = intersections[0]  
 x2, y2 = intersections[1]  
 value2 = math.sqrt((x2 - x1) \*\* 2 + (y2 - y1) \*\* 2)  
  
  
 cash\_length=max(value1,value2)  
 print('cash\_length\_px:',cash\_length,'px')  
 cash\_length\_mm=155  
 length\_per\_px=cash\_length\_mm/cash\_length  
  
  
# 使用Hough圆变换检测圆  
circles = cv2.HoughCircles(gray, cv2.HOUGH\_GRADIENT, dp=1, minDist=1000,  
 param1=300, param2=5, minRadius=0, maxRadius=200)  
  
  
if circles is not None:  
 circles = circles[0, :, :]  
 circles = np.uint16(np.around(circles))  
 for (x, y, r) in circles:  
 # 圆形  
 cv2.circle(img, (x, y), r, (255, 0, 0), 3)  
 # 圆心  
 cv2.circle(img, (x, y), 2, (255, 0, 0), 5)  
 print("center of circle:", x, y)  
 # 半径  
 cv2.line(img, (x, y), (x + r, y), (255, 0, 0), 2)  
  
 radius\_mm = r \* length\_per\_px  
 print('radius(px):', r)  
 print('radius(mm):', radius\_mm)  
 # 保留两位小数输出半径  
 text = "r=" + str(round(radius\_mm, 2)) + "mm"  
 cv2.putText(img, text, (x, y), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (255, 255, 255), 2)  
  
# 输出图像  
cv2.imwrite('output\_image.jpg', img)  
  
# 显示结果图像  
cv2.imshow('Hough Transform Result', img)  
cv2.waitKey(0)  
cv2.destroyAllWindows()

实验三

import cv2 as cv  
import numpy as np  
import math  
  
def seacoastline(frame, minVal=10, maxVal=20, minLineLength=750, maxLineGap=60):  
 frame\_gray = cv.cvtColor(frame, cv.COLOR\_RGB2GRAY)  
 frame\_edges = cv.Canny(frame\_gray, minVal, maxVal)  
  
 lines = cv.HoughLinesP(frame\_edges, 1, np.pi / 180, 20, minLineLength=minLineLength, maxLineGap=maxLineGap)  
 lines = lines[:, 0, :]  
  
 for x1, y1, x2, y2 in lines:  
 cv.line(frame\_edges, (int(x1), int(y1)), (int(x2), int(y2)), (255, 0, 0), 7)  
  
 max\_length = -1  
 max\_x1, min\_x1, max\_y1, min\_y1 = 0, 0, 0, 0  
  
 for line in lines:  
 x1, y1, x2, y2 = line  
 length = abs(y2 - y1)  
 if length > max\_length:  
 max\_length = length  
 max\_x1, min\_x1, max\_y1, min\_y1 = x2, x1, y2, y1  
  
 coastline = [min\_x1, max\_x1, min\_y1, max\_y1]  
 return frame\_edges, coastline  
  
def find\_sea\_line(frame, coastline, max\_distance=65, search\_intervals=7, gap\_distance=10):  
 height, width = frame.shape[0], frame.shape[1]  
 start\_x, end\_x, start\_y, end\_y = coastline  
 delta\_x = end\_x - start\_x  
 delta\_y = start\_y - end\_y  
  
 k = delta\_y / delta\_x  
 k\_ver = (-1) / k  
  
 sea\_line\_segments = []  
 distances = []  
  
 for i in range(int(delta\_y / gap\_distance)):  
 y0 = start\_y - i \* gap\_distance  
 x0 = start\_x + (i / k) \* gap\_distance  
 x\_side = width - x0  
 y\_side = height - y0  
 y\_side /= (-k\_ver)  
  
 side\_length = min(x\_side, y\_side, max\_distance)  
 side\_length = int(side\_length)  
  
 counts = [0] \* (side\_length + 1)  
 size\_x = 1 / math.sqrt(1 + k\_ver \*\* 2)  
 size\_y = (-k\_ver) \* size\_x  
  
 for j in range(side\_length):  
 y = int(y0 + j \* size\_y)  
 x = int(x0 + j \* size\_x)  
  
 if j == 0 and frame[y][x] > 20:  
 counts[j] = 1  
 elif frame[y][x] > 20:  
 counts[j] = counts[j - 1] + 1  
 else:  
 counts[j] = counts[j - 1]  
  
 counts[side\_length] = counts[side\_length - 1]  
 size = int(side\_length / search\_intervals)  
 selected\_point = 0  
 min\_change = 1000  
  
 for t in range(search\_intervals):  
 change = counts[(t + 1) \* size] - counts[t \* size]  
 if change < min\_change:  
 selected\_point = t  
 min\_change = change  
  
 selected\_point \*= size  
 x\_point = selected\_point / math.sqrt(1 + k\_ver \*\* 2)  
 y\_point = (-k\_ver) \* x\_point  
  
 cv.line(frame, (int(x0), int(y0)), (int(x0 + x\_point), int(y0 + y\_point)), (255, 0, 0), 2)  
 sea\_line\_segments.append([int(x0), int(x0 + x\_point), int(y0), int(y0 + y\_point)])  
 distances.append(selected\_point)  
  
 return sea\_line\_segments, distances  
  
def sealine\_draw(frame, line, distance, height\_m):  
 ox, oy = 20, 800  
 sum\_dis = sum(distance) / len(distance)  
 xs, xe, ys, ye = line  
 dx, dy = xe - xs, ys - ye  
 k = dy / dx  
 k1 = (-1) / k  
 size\_x = 1 / math.sqrt(1 + k1 \*\* 2)  
 size\_y = (-k1) \* size\_x  
  
 cv.line(frame, (xs + int(sum\_dis \* size\_x), ys + int(sum\_dis \* size\_y)),  
 (xe + int(sum\_dis \* size\_x), ye + int(sum\_dis \* size\_y)), (0, 0, 255), 2) # 修改颜色为红色 (BGR)  
  
  
 height, width, \_ = frame.shape  
  
 text\_width, text\_height = cv.getTextSize(f"sea\_level\_m={height\_m:.3f}", cv.FONT\_HERSHEY\_SIMPLEX, 1, 2)[0]  
 text\_x = (width - text\_width) // 2  
 text\_y = (height + text\_height) // 2  
  
 cv.putText(frame, f"sea\_level\_m={height\_m:.3f}", (text\_x, text\_y), cv.FONT\_HERSHEY\_SIMPLEX, 1, (0, 255, 0), 2,  
 cv.LINE\_AA, False)  
  
 cv.imwrite('./res.jpg', frame)  
  
 cv.imshow('img', frame)  
 cv.waitKey()  
  
def pixel2m(frame, line, gap=5, threshold\_value=50):  
 gray\_frame = cv.cvtColor(frame, cv.COLOR\_RGB2GRAY)  
 \_, binary\_frame = cv.threshold(gray\_frame, threshold\_value, 255, cv.THRESH\_BINARY)  
  
 xs, xe, ys, ye = line  
 dx, dy = xe - xs, ys - ye  
 k = dy / dx  
 k1 = -1 / k  
 size\_x = 1 / math.sqrt(1 + k1 \*\* 2)  
  
 left\_min = int(abs(xe - xs) / gap + 1)  
  
 for i in range(left\_min, int((xe + 1) / gap)):  
 y\_values = range(frame.shape[0])  
 x\_values = [int(i \* gap + k1 \* y) for y in y\_values]  
 x\_values = np.clip(x\_values, 0, frame.shape[1] - 1)  
  
 pixel\_values = binary\_frame[y\_values, x\_values]  
  
 if np.any(pixel\_values > 20):  
 distance = i \* gap  
 break  
 else:  
 print("未找到目标物体")  
 distance = 0  
  
 distance = xe - distance  
 distance /= size\_x  
 m\_per\_pixel = 1.2 / distance  
  
 return m\_per\_pixel  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 vc = cv.VideoCapture('./3.mp4')  
 t = 4  
 for i in range(t - 1):  
 if vc.isOpened():  
 ret, frame = vc.read()  
 else:  
 ret = False  
 cv.imwrite('./frame.jpg', frame)  
  
 frame\_rec = frame  
 frame, coast\_line\_points = seacoastline(frame, minVal=10, maxVal=20, minLineLength=750, maxLineGap=60)  
 sealine\_points, distance = find\_sea\_line(frame, coast\_line\_points, max\_distance=160, search\_intervals=4,  
 gap\_distance=5)  
 m\_per\_pixel = pixel2m(frame\_rec, coast\_line\_points, gap=6)  
 mean\_dis = sum(distance) / len(distance) \* m\_per\_pixel  
 sealine\_draw(frame\_rec, coast\_line\_points, distance, height\_m=13.8 - mean\_dis)

实验四

import numpy as np  
import cv2  
from skimage import measure  
from imutils import contours  
import imutils  
  
image = cv2.imread(r"C:\Users\43215\Desktop\4.png")  
gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)  
blurred = cv2.GaussianBlur(gray, (11, 11), 0)  
  
# 对图像进行阈值化  
thresh = cv2.threshold(blurred, 212, 255, cv2.THRESH\_BINARY)[1]  
  
# 服饰膨胀消除噪点  
thresh = cv2.erode(thresh, None, iterations=2)  
thresh = cv2.dilate(thresh, None, iterations=4)  
  
labels = measure.label(thresh, connectivity=2, background=0)  
  
mask = np.zeros(thresh.shape, dtype="uint8")  
  
# 调整阈值，过滤小轮廓  
min\_contour\_size = 300  
for label in np.unique(labels):  
 # 如果是噪点或者小轮廓，continue  
 if label == 0 or cv2.countNonZero((labels == label).astype(np.uint8)) < min\_contour\_size:  
 continue  
 # 构造mask  
 labelMask = np.zeros(thresh.shape, dtype="uint8")  
 labelMask[labels == label] = 255  
 mask = cv2.add(mask, labelMask)  
  
cnts = cv2.findContours(mask.copy(), cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)  
cnts = imutils.grab\_contours(cnts)  
cnts = contours.sort\_contours(cnts)[0]  
  
for (i, c) in enumerate(cnts):  
 sumx = 0  
 sumy = 0  
 for j in c:  
 sumx += j[0][0]  
 sumy += j[0][1]  
 avgx = sumx / len(c)  
 avgy = sumy / len(c)  
 (x, y, w, h) = cv2.boundingRect(c)  
 ((cX, cY), radius) = cv2.minEnclosingCircle(c)  
 ((cX, cY), radius) = cv2.minEnclosingCircle(c)  
 cv2.circle(image, (int(cX), int(cY)), int(radius), (0, 0, 255), 2)  
 mask = np.zeros(image.shape, dtype=np.uint8)  
 mask = cv2.fillPoly(mask, c, (255, 255, 255))  
 image[mask] = 0  
 avgx = avgx / 55.56 + 0.5  
 avgy = (avgy - 77.778) / 83.33 + 0.5  
 avgx = format(avgx, '.0f')  
 avgy = format(avgy, '.0f')  
 cv2.putText(image, "({},{})".format(avgx, avgy), (x, y - 15), cv2.FONT\_HERSHEY\_SIMPLEX, 0.85, (255, 255, 0), 3)  
  
# 保存为 "res.jpg"  
cv2.imwrite(r"C:\Users\43215\Desktop\res.jpg", image)