**import** cv2  
**import** numpy **as** np  
**from** matplotlib **import** pyplot **as** plt  
**import** math  
  
src = cv2.imread(**'./recourse/checkerboard .png'**, cv2.IMREAD\_COLOR) *# 识别图片*img1 = cv2.cvtColor(src, cv2.COLOR\_BGR2GRAY) *# 颜色转换变为灰度图*template = cv2.imread(**'./recourse/white.png'**, cv2.IMREAD\_GRAYSCALE) *# 识别白棋*w, h = template.shape[::-1] *# 模板匹配*res = cv2.matchTemplate(img1, template, cv2.TM\_CCOEFF\_NORMED)  
threshold = 0.9 *# 模型评价指标AUC*loc = np.where(res >= threshold)  
**for** pt **in** zip(\*loc[::-1]): *# 绘制矩形框* cv2.rectangle(src, pt, (pt[0] + w + 4, pt[1] + h + 9), (0, 0, 255), 1)  
template = cv2.imread(**'./recourse/black.png'**, cv2.IMREAD\_GRAYSCALE) *# 识别黑棋*w1, h1 = template.shape[::-1]  
res1 = cv2.matchTemplate(img1, template, cv2.TM\_CCOEFF\_NORMED)  
threshold1 = 0.87  
loc1 = np.where(res1 >= threshold1)  
**for** pt **in** zip(\*loc1[::-1]): *# 绘制矩形框* cv2.rectangle(src, pt, (pt[0] + w1 + 5, pt[1] + h1 + 6), (0, 255, 0), 1)  
  
lan = cv2.imread(**'./recourse/checkerboard .png'**)  
gray = cv2.cvtColor(lan, cv2.COLOR\_BGR2GRAY) *# 转为灰度图*edges = cv2.Canny(gray, 50, 100, apertureSize=3) *# 边缘检测*minlengeth = 100  
maxlineGap = 20  
lines = cv2.HoughLinesP(edges, 1, np.pi / 180, 100, minlengeth, maxlineGap) *# 检测直线***for** each **in** range(len(lines)): *# 将识别到的所有线条画出* **for** x1, y1, x2, y2 **in** lines[each]:  
 cv2.line(src, (x1, y1), (x2, y2), (255, 200, 0), 2)  
  
m = -1  
m1 = -1  
**for** pt1 **in** zip(\*loc[::-1]): *# 模板匹配并翻转图像* **for** pt2 **in** zip(\*loc[::-1]):  
 float(m)  
 a = ((pt1[0] - pt2[0]) \*\* 2 + (pt1[1] - pt2[1]) \*\* 2) \*\* 1 / 2 *# 计算各与模板匹配的对象之间的距离* **if** a > m: *# 寻找最大值* m = a  
 pt1m = (pt1[0] + int(w / 2), pt1[1] + int(h / 2))  
 pt2m = (pt2[0] + int(w / 2), pt2[1] + int(h / 2))  
  
**for** pt3 **in** zip(\*loc1[::-1]): *# 模板匹配并翻转图像* **for** pt4 **in** zip(\*loc1[::-1]):  
 float(m1)  
 a = ((pt3[0] - pt4[0]) \*\* 2 + (pt3[1] - pt4[1]) \*\* 2) \*\* 1 / 2 *# 计算各与模板匹配的对象之间的距离* **if** a > m1: *# 寻找最大值* m1 = a  
 pt3m = (pt3[0] + int(w / 2), pt3[1] + int(h / 2))  
 pt4m = (pt4[0] + int(w / 2), pt4[1] + int(h / 2))  
  
cv2.line(src, pt1m, pt2m, (0, 255, 255), 2) *# 连接距离最大的白子*cv2.line(src, pt3m, pt4m, (255, 0, 255), 2) *# 连接距离最大的黑子*cv2.imshow(**'image'**, src) *# 显示图片*cv2.waitKey(0) *# 按任意键关闭窗口*cv2.destroyAllWindows()

