**BÁO CÁO BÀI TẬP 6**

**Nhóm 18**

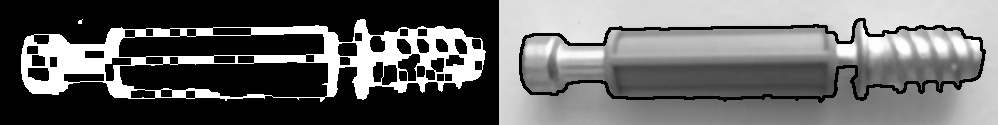
1. Lương Hữu Phú Lộc – 1511844
2. Phạm Ngọc Khôi Nguyên – 1512221
3. Nguyễn Trọng Phúc – 1512534
4. Mai Thiện Quang – 1512640
5. **Yêu cầu**

Robot cần tìm một vật xác định để gắp. Dùng phương pháp Template matching  Hu Moment, matchShapes… tìm tọa độ trong tâm của vật.

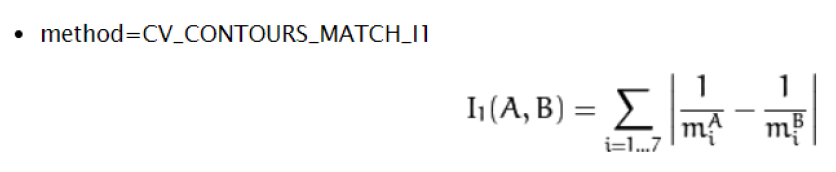


1. **Giải quyết bài toán**
2. **Giải thuật**

* Sử dụng phương pháp matchShape, dựa trên các giá trị Hu Moments để kết quả không bị phụ thuộc bởi các phép quay, tỉ lệ,…
* matchShape đòi hỏi 2 tham số đó là contour của vật mẫu và contour của vật thể cần xác định, do đó trước tiên ta đi tìm contour của vật mẫu
* Nhóm sử dụng phương pháp lấy cạnh Canny, tìm contour vật mẫu (như hình vẽ), sau đó lưu lại giá trị contour này với tên cnt để sử dụng cho hàm matchShape



* Đối với hình chứa vật thể cần tìm, ta cũng tìm contour tất cả các vật thể trong hình. Do ảnh hưởng bởi ánh sáng nên nhóm sử dụng phân ngưỡng thích nghi (adaptive threshold) để chuyển ảnh qua nhị phân trước khi tìm contour.
* Sau đó với mỗi contour, ta tính giá trị của hàm matchShape, sử dụng phương pháp CV\_CONTOURS\_MATCH\_I1



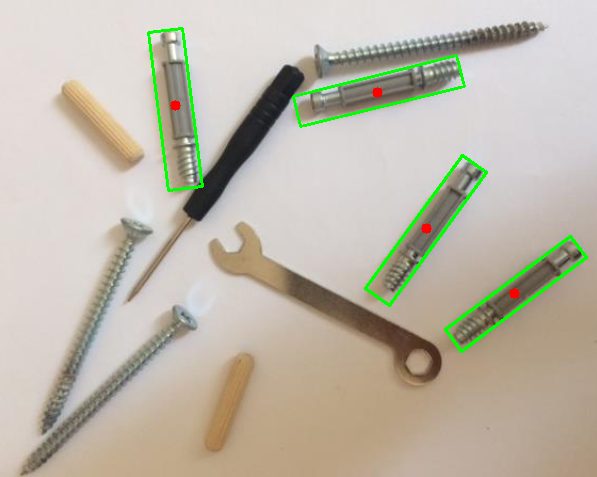
Đặc điểm của phương pháp này là kết quả trả về càng nhỏ thì vật đang xét càng giống với vật mẫu. Tuy nhiên, thực tế vật thể tìm được trong hình và vật mẫu không hoàn toàn giống nhau, cho nên ta sẽ dùng một mức ngưỡng với các giá trị matchShape để xem vật đang xét có giống với vật mẫu hay không. Đối với bài toán này mức ngưỡng nhóm sử dụng là bằng 1, nếu giá trị matchShape bé hơn 1 nghĩa là vật giống với vật mẫu và ngược lại

* Nếu vật đang xét giống với vật mẫu, ta tìm trọng tâm của vật bằng giá trị moment của contour, toạ độ trọng tâm được tính bằng công thức: 

Với M là moment của contour đang xét

1. **Kết quả**

* Trọng tâm của vật ký hiệu là chấm đỏ tương ứng trên hình

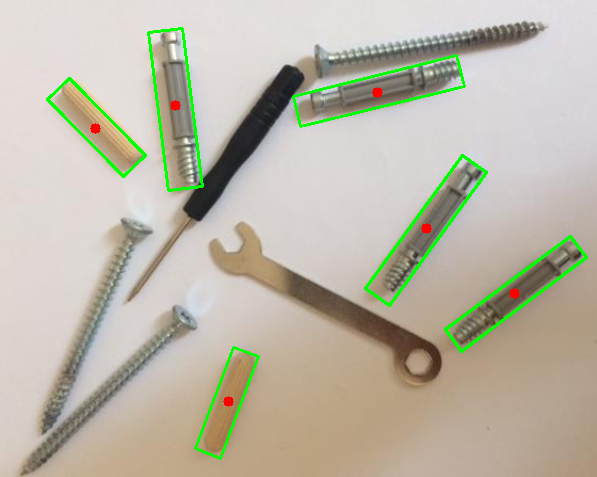
****

1. **Nhận xét**

* Phương pháp thực hiện tương đối đơn giản, tuy nhiên kết quả vẫn có thể chấp nhận được.
* Điểm cần chú ý khi thực hiện giải thuật này là khi chọn mức ngưỡng khi tính giá trị matchShape. Những vật thể dù giống với vật, nhưng bị ảnh hưởng bởi các yếu tố nhiễu, đổ bóng, ánh sáng, thì khi tìm contour giá trị matchShape có thể lớn hơn giá trị ngưỡng đã chọn và ta phải điều chỉnh giá trị ngưỡng này
* Ví dụ giá trị ngưỡng chọn bằng 0.5, ta chỉ tìm được một vật giống với vật mẫu



* Còn với ngưỡng bằng 5, kết quả trả về có thêm một số vật không giống với vật mẫu



* Để giải thuật hiệu quả hơn, ta cần có thêm các bước tiền xử lí như loại bỏ bóng vật thể, cân chỉnh ánh sáng cho hình, vì bóng của vật ảnh hưởng rất nhiều tới quá trình lấy contour vật thể, làm kết quả không chính xác. Nhóm cũng đã tìm cách loại bỏ bóng của vật nhưng chưa thành công nên xin được phép không trình bày trong báo cáo.

1. **Chương trình thực hiện trên python**

**import** **numpy** **as** **np**

**import** **cv2**

*#Read template image*

template = cv2.imread('template.png',0)

template = cv2.medianBlur(template,3)

*# Find canny of template image*

edges = cv2.Canny(template,7,46)

kernel = np.ones((7,7),np.uint8)

edges = cv2.morphologyEx(edges, cv2.MORPH\_CLOSE, kernel)

*# Find contours of object in image*

\_, contours, \_ = cv2.findContours(edges, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

cv2.drawContours(template,contours,0,(0,255,0),2)

res = np.hstack((edges,template))

cv2.imshow('res',res)

cv2.imwrite('res.png',res)

*# Choose biggest contour as object's contour*

cnt = contours[0]

*# Read image which has objects that we need to find*

img = cv2.imread('image.png',1)

gray = cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)

*# Process image to make better result*

gray = cv2.GaussianBlur(gray,(3,3),0)

binary = cv2.adaptiveThreshold(gray,255,cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C,cv2.THRESH\_BINARY\_INV,77,6)

kernel = np.ones((3,3),np.uint8)

binary = cv2.morphologyEx(binary, cv2.MORPH\_CLOSE, kernel)

cv2.imshow('binary',binary)

*# Find contours*

\_, contours, \_ = cv2.findContours(binary, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

*# Find objects*

**for** i **in** range(len(contours)):

**if** cv2.contourArea(contours[i]) > 1000 :

*# Calculate matchShape value*

ret = cv2.matchShapes(contours[i],cnt,1,0.0)

**if** ret < 1: *#Threshold value*

*# Draw circle at center*

M = cv2.moments(contours[i])

cx = int(M['m10']/M['m00'])

cy = int(M['m01']/M['m00'])

cv2.circle(img,(cx,cy),5,[0,0,255],-1)

*# Draw rectangle around object*

rect = cv2.minAreaRect(contours[i])

box = cv2.boxPoints(rect)

box = np.int0(box)

cv2.drawContours(img,[box],0,(0,255,0),2)

cv2.imshow('img',img)

cv2.imwrite('img.png',img)

cv2.waitKey(0)

cv2.destroyAllWindows()