UFPB-CI-DSC
Disciplina: Visão Computacional
Prof. Augusto de Holanda B. M. Tavares
25 de Junho de 2024

11ª Atividade

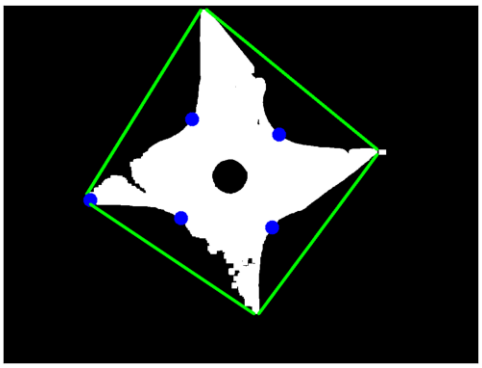
Defeitos de convexidade e identificação de formas

Considerando a imagem abaixo:

Utilizando métodos de visão computacional, realize as seguintes tarefas:

1. Crie uma máscara e obtenha os contornos para a estrela na imagem acima. Isole o contorno que mais se assemelha à forma geral do objeto, como exemplificado abaixo.

2. Obtenha os defeitos de convexidade para este contorno maior, traçando as linhas que formam a casca convexa e marcando os pontos extremos dos defeitos com círculos, como no exemplo abaixo:



3. Rotacione a imagem de exemplo em 90 graus, obtenha o contorno correspondente ao caso original e calcule o grau de similaridade entre as formas nas diferentes imagens.

4. Crie uma imagem preta com as mesmas dimensões da imagem original e desenhe um polígono preenchido a partir dos pontos que formam a casca convexa. Obtenha o contorno deste polígono e calcule o grau de similaridade entre as formas.

5. Repita o processo da tarefa anterior com uma imagem contendo o círculo que envolve o contorno do polígono formado pelos pontos da casca convexa.

Funções de referência

Algumas funções de referência necessárias para implementar as tarefas acima são descritas abaixo:

```
hull = cv.convexHull(cnt,returnPoints = False)
defects = cv.convexityDefects(cnt,hull)
```

- Retorna o vetor dos defeitos $defects$ de convexidade do contorno cnt. Cada elemento $defects[i, 0]$ do vetor irá conter os índices **dos pontos do contorno** de início e fim s e e de um segmento de reta da casca convexa e o índice **do ponto do contorno** mais distante deste segmento f. A *flag returnPoints* está definida em *False* acima para gerar um objeto no formato exigido por *convexityDefects*. No entanto, os pontos da casca convexa podem ser obtidos definindo esta flag em *False*

.

```
img_rot = cv.rotate(img,cv.ROTATE_90_CLOCKWISE)
```

- Retorna a imagem original rotacionada em 90 graus no sentido horário. Há outros parâmetros para rotações distintas.

```
ret = cv.matchShapes(contour1,contour2,method,0.0)
print(ret)
```

- Fornece uma métrica da similaridade entre as formas definidas por $contour1$ e $contour2$. Quanto menor o resultado, maior a similaridade. O parâmetro $method$ define o método de comparação, e para este caso pode ser definido em 1.

```
cv.fillPoly(img, pts=[points], color=(r, g, b))
```

- Traça um polígono preenchido de cor (r, g, b) com os seus vértices definidos por $[points]$ na imagem img.

```
cv.circle(img,(x,y),radius, (r,g,b),thickness)
```

- Traça um círculo de cor (r, g, b) com raio $radius$ e grossura da linha $thickness$ na imagem img ($thickness = -1$ gera um círculo preenchido).