Detecção de Contornos e Aproximação Poligonal

ES235 - Aula 16

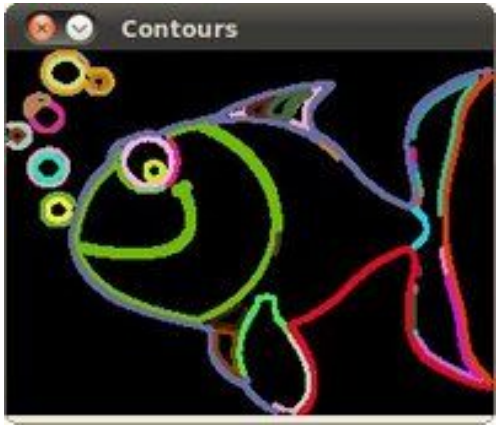
João Marcelo Teixeira
Willams Costa

Detecção de Contornos != Canny

- O cvCanny determina as arestas da imagem, ou seja, o conjunto de todos os pixels que fazem parte dos pixels de aresta
- O findContours retorna conjuntos de pixels conectados, tornando possível diferenciá-los (o drawContours pode ser usado para pintar os contornos com cores distintas)
- Deteção de contornos pode ser usada para:
  - Análise de formas
  - Detecção e reconhecimento de objetos

Detecção de Contornos != Canny

- Para melhor precisão, recomenda-se utilizar como entrada do findContours uma imagem binarizada
- O algoritmo considera a cor de fundo como preto
- É comum utilizar o resultado do canny como entrada da detecção de contornos



Detecção de Contornos != Canny

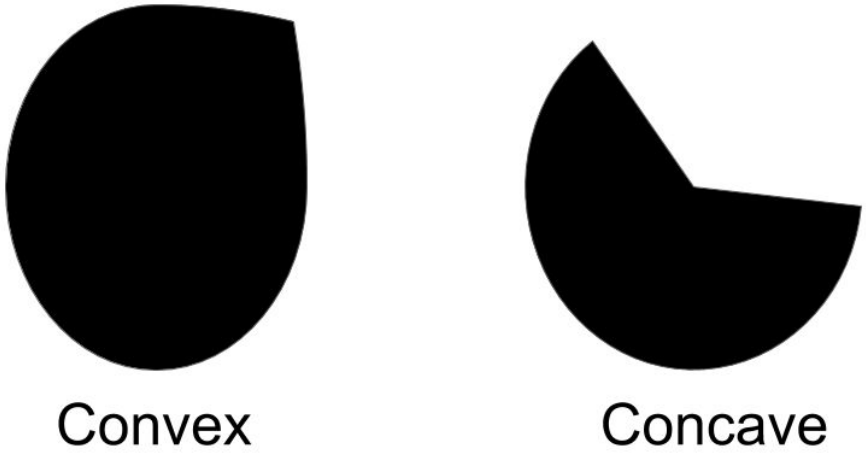
- cv2.findContours(image, mode, method[, contours[, hierarchy[, offset]]])
  - image - image de entrada, 8 bits
  - contours - contornos detectados (cada um é um vetor de pontos)
  - hierarchy - contém informação sobre o nível hierárquico dos contornos da imagem (quem está dentro de quem)
  - mode - CV_RETR_EXTERNAL, CV_RETR_LIST, CV_RETR_CCOMP,CV_RETR_TREE
    method - CV_CHAIN_APPROX_NONE, CV_CHAIN_APPROX_SIMPLE, CV_CHAIN_APPROX_TC89_L1,CV_CHAIN_APPROX_TC89_KCOS
  - offset - usado quando se deseja aplicar um offset às coordenadas dos contornos detectados

Detecção de Contornos != Canny

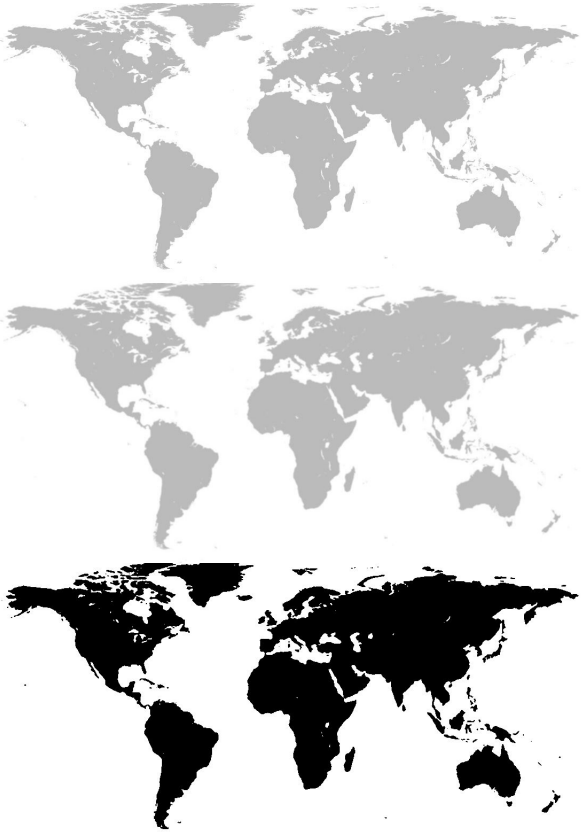
- cv.DrawContours(img, contour, external_color, hole_color, max_level, thickness=1, lineType=8, offset=(0, 0))
  - image - imagem de saída
  - contours - contornos que serão desenhados (obtido com o findContours)
  - contourIdx - ID do contorno a ser desenhado (se negativo, desenha todos)
  - color - cor dos contornos
  - thickness - espessura das linhas do contorno
  - ...

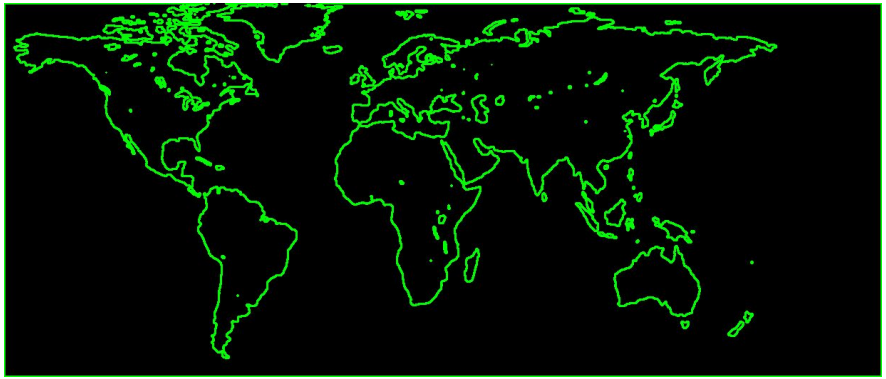
Convex Hull (Fecho Convexo)

- Usado para encontrar o fecho convexo de um contorno



Passo a Passo





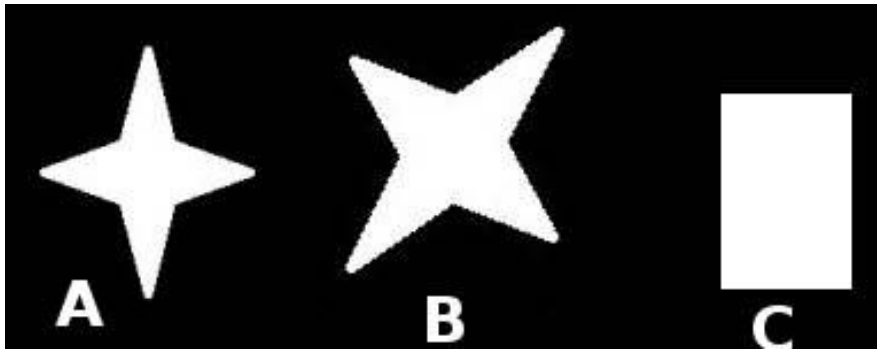
Casamento de Formas (Shape Matching)

- cv.MatchShapes(object1, object2, method, parameter=0)
  - object1 - primeiro contorno a ser comparado
  - object2 - segundo contorno a ser comparado
  - method - tipo de comparação: CV_CONTOURS_MATCH_I1, CV_CONTOURS_MATCH_I2, CV_CONTOURS_MATCH_I3
- Quanto mais próximos são os contornos, mais próximo de zero é o float com o valor retornado

Matching Image A with itself = 0.0

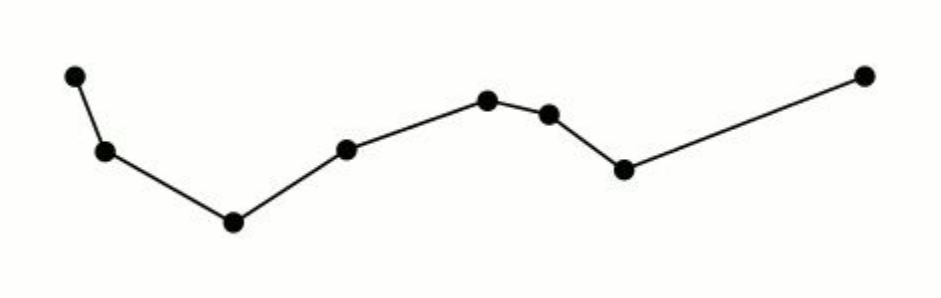
Matching Image A with Image B = 0.001946

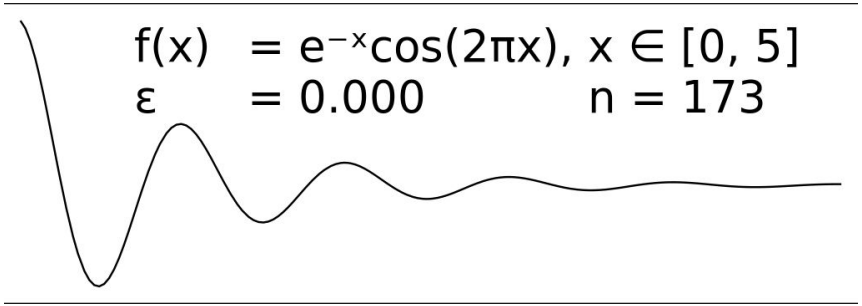
Matching Image A with Image C = 0.326911



Aproximação Poligonal

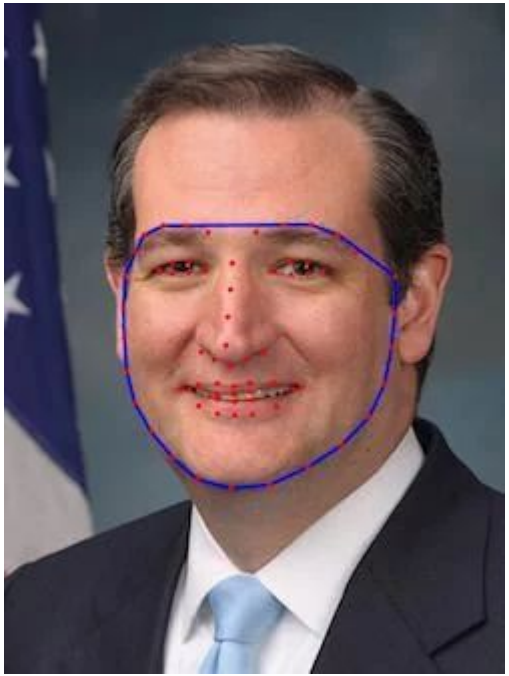
- `cv2.approxPolyDP(curve, epsilon, closed[, approxCurve])`





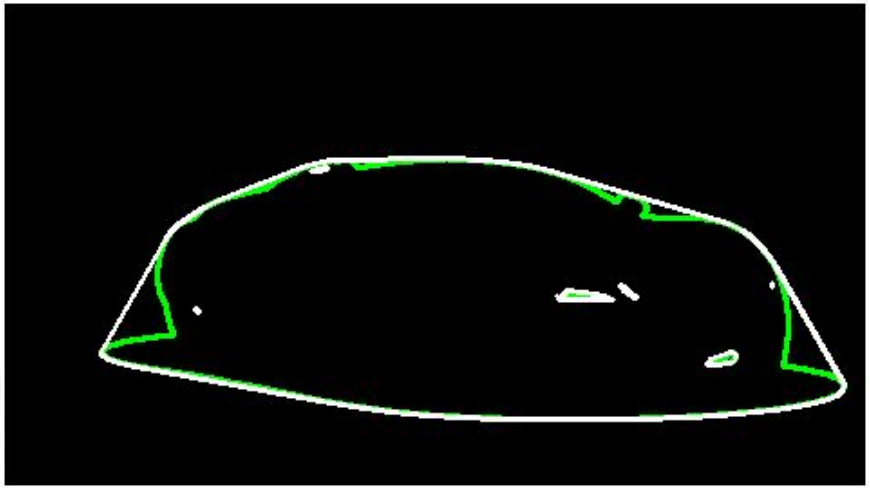
Aplicações

- Boundary from a set of points



Aplicações

- Collision Avoidance



REFERÊNCIAS

Rafael C. Gonzalez and Richard E. Woods. 2006. Digital Image Processing (3rd Edition). Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

https://docs.opencv.org/3.4/d4/d73/tutorial_py_contours_begin.html

https://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html?highlight=findcontours#suzuki85

https://docs.opencv.org/3.4.0/d7/d1d/tutorial_hull.html

https://www.learnopencv.com/convex-hull-using-opencv-in-python-and-c/