東南大學 SOUTHEAST UNIVERSITY 人工智能学院

专业技能实训

授课教师：王洪松

邮箱：hongsongwang@seu.edu.cn

图像分割

颜色分离

- 将图像中某种颜色分离出来。

dst = cv2.inRange(src, lowerb, upperb)

src: 输入图像。dst: 输出mask图像，$\texttt{dst}(I) = \texttt{lowerb}(I)_0 \leq \texttt{src}(I)_0 \leq \texttt{upperb}(I)_0$

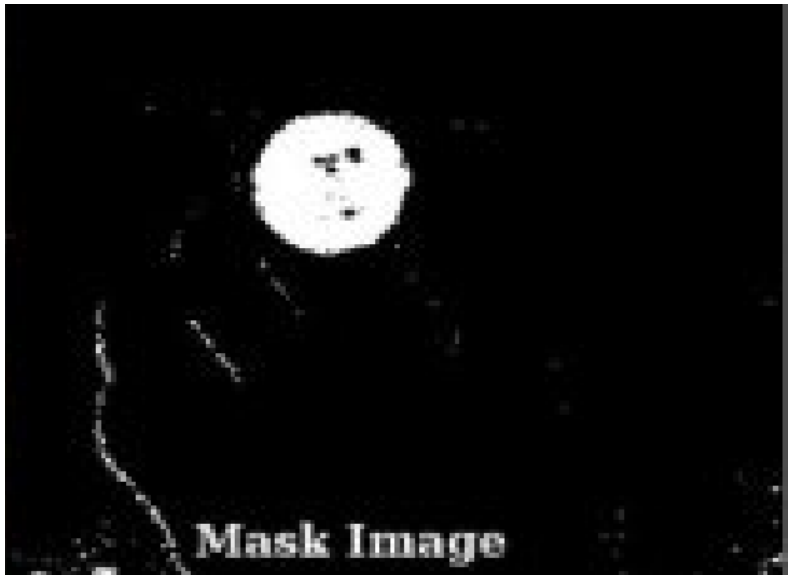
lowerb: 下边界数组或标量。upperb: 上边界数组或标量。

```
hsv = cv2.cvtColor(rgb, cv2.COLOR_RGB2HSV)
# 设定参数lowerb、upperb
lowerb = np.array([...])
upperb = np.array([...])
# 获取mask
mask = cv2.inRange(hsv, lowerb, upperb)
# Bitwise-AND mask and original image
res = cv2.bitwise_and(frame, frame, mask= mask)
```

颜色分离

```
hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
# define range of blue color in HSV
lower_blue = np.array([110,50,50])
upper_blue = np.array([130,255,255])
# Threshold the HSV image to get only blue colors
mask = cv2.inRange(hsv, lower_blue, upper_blue)
```

	黑	灰	白	红		橙	黄	绿	青	蓝	紫
hmin	0	0	0	0	156	11	26	35	78	100	125
hmax	180	180	180	10	180	25	34 ;	77	99	124	155
smin	0	0	0	43		43	43	43	43	43	43
smax	255	43	30	255		255	255	255	255	255	255
vmin	0	46	221	46		46	46	46	46	46	46
vmax	46	220	255	255		255	255	255	255	255	255



阈值化处理

- 阈值处理是指剔除图像内像素值高于一定值或者低于一定值的像素点。

retval, dst = cv2.threshold(src, thresh, maxval, type)

retval 代表返回的阈值。
dst 代表阈值分割结果图像，与原始图像具有相同的大小和类型。
src 代表要进行阈值分割的图像，可以是多通道的。
thresh 代表要设定的阈值。

maxval 代表当type参数为THRESH_BINARY或THRESH_BINARY_INV类型时，需要设定的最大值。
type 代表阈值分割的类型。

阈值化处理

阈值分割类型

类型	定义
cv2.THRESH_BINARY	$\mathrm{dst}(x,y)=\begin{cases}\mathrm{maxval}, & \mathrm{src}(x,y)>\mathrm{thresh}\\ 0, & \text{其他情况}\end{cases}$
cv2.THRESH_BINARY_INV	$\mathrm{dst}(x,y)=\begin{cases}0, & \mathrm{src}(x,y)>\mathrm{thresh}\\ \mathrm{maxval}, & \text{其他情况}\end{cases}$
cv2.THRESH_TRUNC	$\mathrm{dst}(x,y)=\begin{cases}\mathrm{thresh}, & \mathrm{src}(x,y)>\mathrm{thresh}\\ 0, & \text{其他情况}\end{cases}$
cv2.THRESH_TOZERO_INV	$\mathrm{dst}(x,y)=\begin{cases}0, & \mathrm{src}(x,y)>\mathrm{thresh}\\ \mathrm{src}(x,y), & \text{其他情况}\end{cases}$
cv2.THRESH_TOZERO	$\mathrm{dst}(x,y)=\begin{cases}\mathrm{src}(x,y), & \mathrm{src}(x,y)>\mathrm{thresh}\\ 0, & \text{其他情况}\end{cases}$
cv2.THRESH_MASK	掩码
cv2.THRESH_OTSU	标记，使用 Otsu 算法时的可选阈值参数
cv2.THRESH_TRIANGLE	标记，使用 Triangle 算法时的可选阈值参数

Otsu 阈值分割

- Otsu 方法会遍历所有可能阈值，从而找到最佳的阈值。
- 在函数 cv2.threshold()中对参数 type 的类型多传递一个参数“cv2.THRESH_OTSU”，即可实现 Otsu 方式的阈值分割。
- 使用 Otsu 方法时，要把阈值设为 0。函数 cv2.threshold()会自动寻找最优阈值，并将该阈值返回。

```
t1,thd=cv2.threshold(img,127,255,cv2.THRESH_BINARY)
```

```
t2,otsu=cv2.threshold(img,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)
```

从左到右依次为：
原始图像
以 127 为阈值的处理结果
Otsu 阈值分割结果

交互式前景提取

- 交互式前景提取GrabCut算法，原理涉及到K均值聚类、高斯混合模型建模(GMM)、最大流最小割。

  cv2.grabCut(img,mask,rect,bgdModel,fgdModel, iterCount, mode)

img：输入图像。

mask：掩码图像，指定哪些区域是背景、前景或可能的背景/前景等。它由以下标志完成，cv2.GC_BGD、cv2.GC_FGD、cv2.GC_PR_BGD、cv2.GC _PR_FGD，或简单地将0、1、2、3传递到图像。

rect：包含前景目标的长方形坐标 (x,y,w,h)。

bdgModel, fgdModel：算法内部使用的数组。只需创建两个大小为（1, 65）的np.float64类型的零数组。

iterCount：算法迭代次数。

mode：cv2.GC_INIT_WITH_RECT或cv2.GC_INIT_WITH_2MASK或组合，这决定了绘制矩形还是final touchup strokes。

交互式前景提取

- 输入矩形rect

```
img = cv2.imread('LenaRGB.bmp')
mask = np.zeros(img.shape[:2],np.uint8)
bgdModel = np.zeros((1,65),np.float64)
fgdModel = np.zeros((1,65),np.float64)
rect = (50,50,400,450)
cv2.grabCut(img,mask,rect,bgdModel,fgdModel,5,
cv2.GC_INIT_WITH_RECT)
mask2 =
np.where((mask==2)|(mask==0),0,1).astype('uint8')
img2 = img*mask2[:,:,np.newaxis]
```

原始图像

分割结果图像

交互式前景提取

- 输入mask

```
# newmask is the mask image I manually labelled
newmask = cv2.imread(“newmask.png”, 0)

# whereever it is marked white (sure foreground), change mask=1
# whereever it is marked black (sure background), change mask=0
mask[newmask == 0] = 0
mask[newmask == 255] = 1
mask, bgdModel, fgdModel =
cv2.grabCut(img,mask,None,bgdModel,fgdModel,5,cv2.GC_INIT_WITH_
MASK)
mask = np.where((mask==2)|(mask==0),0,1).astype('uint8')
```

图像轮廓

图像轮廓

- 图像轮廓是指将边缘连接起来形成的一个整体。
- 边缘检测虽然能够检测出边缘，但边缘是不连续的，检测到的边缘并不是一个整体。
- 图像轮廓是图像中非常重要的一个特征信息，通过对图像轮廓的操作，我们能够获取目标图像的大小、位置、方向等信息。

查找图像轮廓

```
contours, hierarchy = cv2.findContours( image, mode, method)
```

image：二值图像，可预先使用阈值处理等函数将待查找轮廓的图像处理为二值图像。
mode： 轮廓检索模式。method： 轮廓的近似方法。

图像轮廓

- 参数 mode 决定了轮廓的提取方式，具体有如下：

cv2.RETR_EXTERNAL：只检测外轮廓。

cv2.RETR_LIST：对检测到的轮廓不建立等级关系。

cv2.RETR_CCOMP： 检索所有轮廓并将它们组织成两级层次结构。

cv2.RETR_TREE：建立一个等级树结构的轮廓。

- 参数 method 决定了如何表达轮廓：

cv2.CHAIN_APPROX_NONE：存储所有的轮廓点，相邻两个点的像素位置差不超过 1。

cv2.CHAIN_APPROX_SIMPLE：压缩水平方向、垂直方向、对角线方向的元素，只保留该方向的终点坐标。

cv2.CHAIN_APPROX_TC89_L1：使用 teh-Chinl chain 近似算法的一种风格。

cv2.CHAIN_APPROX_TC89_KCOS：使用 teh-Chinl chain 近似算法的一种风格。

图像轮廓表示

- 轮廓表示：contours 的 type 是 list 类型，list 的每个元素对应图像的一个轮廓，用 Numpy中的 维度为(m, 1, 2)的ndarray 表示，m表示表示轮廓的点的个数。
- 绘制图像轮廓

  image=cv2.drawContours(image, contours, contourIdx, color[, thickness[, lineType[, hierarchy[, maxLevel[, offset]]]]])

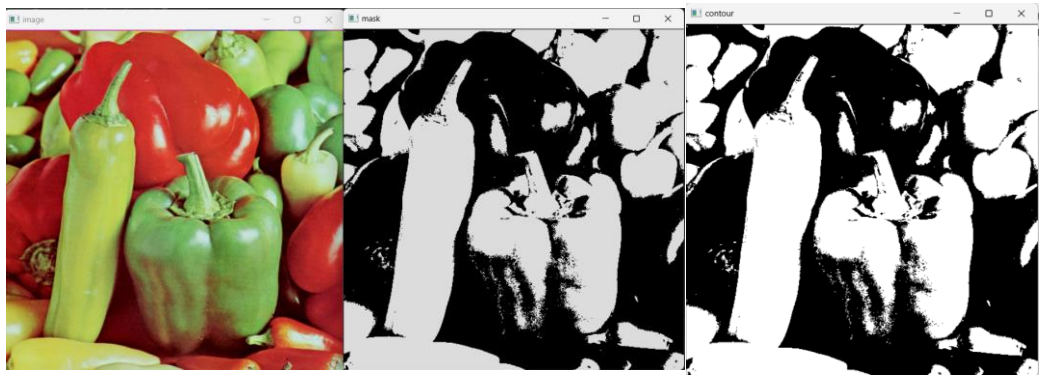
image： 待绘制轮廓的图像。直接在图像 image 上直接绘制轮廓，在函数执行完以后， image 不再是原始图像，而是包含了轮廓的图像。因此，如果图像 image 还有其他用途的话，则需要预先复制一份。
contours：需要绘制的轮廓。
contourIdx：需要绘制的边缘索引。如果该值为-1，则表示绘制全部轮廓。
color：绘制的颜色。 thickness：可选参数，表示绘制轮廓时所用画笔的粗细。如将该值设置为-1，则表示要绘制实心轮廓。

图像轮廓代码

```
image = cv2.imread('PeppersRGB.bmp')
gray = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
ret, binary = cv2.threshold(gray,120,220,cv2.THRESH_BINARY)
contours, hierarchy = cv2.findContours(binary, cv2.RETR_LIST,
cv2.CHAIN_APPROX_SIMPLE)

zero_mask = np.zeros(image.shape,np.uint8)
result = cv2.drawContours(gray, contours,-1,(255,255,255),-1)
contour_image = cv2.drawContours(zero_mask, contours,-1,(255,255,255),-1)

cv2.imshow("mask", binary)
cv2.imshow("contour", contour_image)
cv2.imshow("result",result)
cv2.waitKey()
cv2.destroyAllWindows()
```



原图　　二值图像　　轮廓图像

轮廓拟合

- 在计算轮廓时，可能并不需要实际的轮廓，而仅需要一个接近于轮廓的近似多边形。
- 矩形包围框 retval = cv2.boundingRect(array)

  返回值表示矩形边界的左上角顶点的坐标值及矩形边界的宽度和高度。
- 最小包围矩形框 retval = cv2.boundingRect(array)

  返回值的结构是最小外接矩形的中心(x,y),(宽度,高度), 旋转角度。
- 逼近多边形 approxCurve = cv2.approxPolyDP(curve, epsilon, closed)

返回值 approxCurve 为逼近多边形的点集。curve 是轮廓。

epsilon 为精度，原始轮廓的边界点与逼近多边形边界之间的最大距离。

closed 是布尔型值。该值为 True 时，逼近多边形是封闭的；否则，逼近多边形是不封闭的。

凸包

- 逼近多边形是轮廓的高度近似，但是有时候，我们希望使用一个多边形的凸包来简化它。凸包是物体最外层的“凸” 多边形。

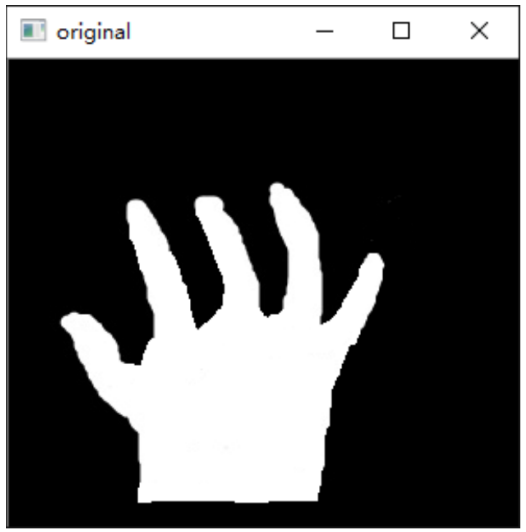
  hull = cv2.convexHull(points[, clockwise[, returnPoints]])

  points：轮廓。
  clockwise：布尔型值。该值为 True 时，凸包角点将按顺时针方向排列；该值为 False 时， 则以逆时针方向排列凸包角点。
  returnPoints：布尔型值。默认值是 True， 函数返回凸包角点的 x/y 轴坐标； 当为 False 时，函数返回轮廓中凸包角点的索引。

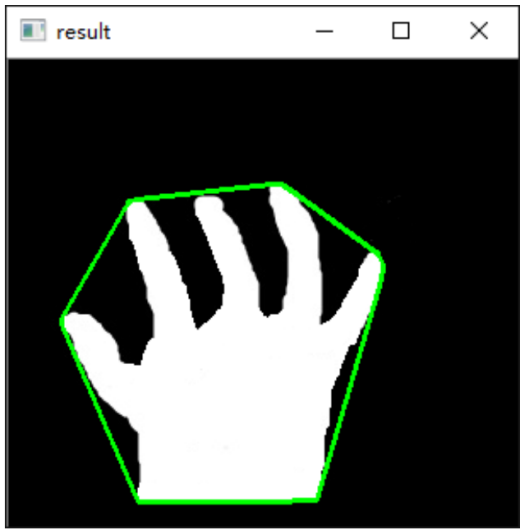
  ```
  hull = cv2.convexHull(contours[0]) # 返回坐标值
  hull2 = cv2.convexHull(contours[0], returnPoints=False) # 返回索引值
  ```

获取轮廓的凸包

```
o = cv2.imread('hand.bmp')
cv2.imshow("original",o)
# --------------提取轮廓------------------
gray = cv2.cvtColor(o,cv2.COLOR_BGR2GRAY)
ret, binary = cv2.threshold(gray,127,255,cv2.THRESH_BINARY)
contours, hierarchy = cv2.findContours(binary, cv2.RETR_LIST,
cv2.CHAIN_APPROX_SIMPLE)
# --------------寻找凸包，得到凸包的角点
hull = cv2.convexHull(contours[0])
# --------------绘制凸包------------------
cv2.polylines(o, [hull], True, (0, 255, 0), 2)
cv2.imshow("result",o)
cv2.waitKey()
cv2.destroyAllWindows()
```



原图



凸包的图像

专业技能实训：练习题

- 用滚动条实现图像颜色分离
  - 用滚动条分别控制输入图像中目标颜色的上边界和下边界。
  - 拖动滚动条，采用颜色分离函数分割得到目标掩码。
  - 绘制目标轮廓。
  - 对比显示原始图像、分割得到的图像和轮廓图像。
- 用滚动条实现图像阈值化处理
  - 用滚动条分别控制阈值和阈值分割类型。
  - 基于所选择的阈值分割类型和阈值，采用阈值化处理分割图像。
  - 绘制目标轮廓。
  - 分别用旋转矩形框和凸包拟合分割得到的目标。
  - 过滤分割结果中的噪声。
  - 对比显示原始图像和分割得到的图像。

专业技能实训：练习题

- 用颜色分离给图像换背景
  - 用颜色分离方法选择图像中的背景区域，用mask表示。
  - 输入新的背景图像，把新图像的大小resize到原图大小。
  - 把原图中的背景像素值设为0。
  - 对mask取反，把背景图像中对mask取反后对应的像素值设为0。
  - 处理后的前景和背景图像相加，得到换背景后的图像。

- 制作证件照
  - 输入人的正面图像。
  - 选择包含人脸和上半身的感兴趣区域，用算法分割出背景或前景。
  - 用一个滚动条控制证件照的背景颜色，比如白色、红色、蓝色等。
  - 拖动滚动条，生成不同背景颜色的人脸证件照。

专业技能实训：练习题

- 图像分割制作透明图像
  - 在原图中拖动鼠标选择矩形框（可用cv2.selectROI()函数） 。
  - 采用GrabCut分割图像，同时显示原始图像和分割得到的图像。
  - 以上过程可以反复迭代，直到得到满意的分割结果 。
  - 按下ESC退出，将原图中除了前景目标之外的其它所有像素设置为透明，保存只包含前景目标的透明图像。

- 交互式图像分割
  - 输入包含某个目标的图像，用鼠标绘制前景目标的mask（掩码）。
  - 绘制完毕，按下Enter键，采GrabCut对输入mask分割图像，同时显示原始图像和分割得到的图像。
  - 以上过程可以反复迭代，直到得到满意的分割结果。
  - 按下ESC退出，保存只包含前景目标的透明图像。