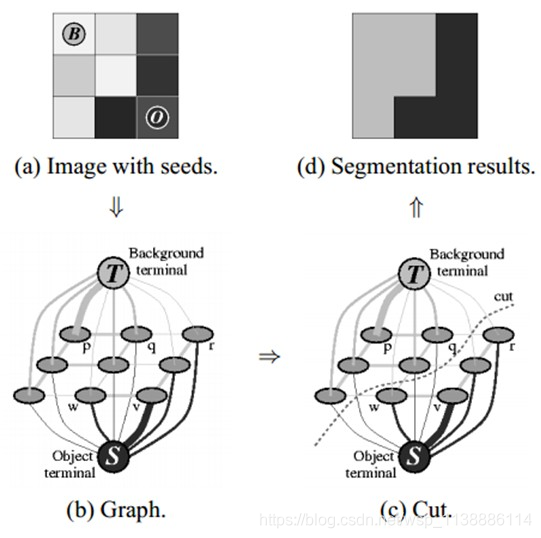
# 一、理论概述

Grabcut是基于图割(graph cut)实现的[图像分割](https://so.csdn.net/so/search?q=%E5%9B%BE%E5%83%8F%E5%88%86%E5%89%B2&spm=1001.2101.3001.7020)算法，它需要用户输入一个bounding box作为分割目标位置，实现对目标与背景的分离/分割，与KMeans与MeanShift等图像分割方法不同。  
**Grabcut分割速度快**，效果好，支持交互操作，因此在很多APP图像分割/背景虚化的软件中可以看到其身影。  
该算法主要基于以下知识：

* k均值聚类
* 高斯混合模型建模(GMM)
* max flow/min cut

关于算法理论：https://blog.csdn.net/kyjl888/article/details/78253829  
GrabCut算法的实现步骤：

* 在图片中定义(一个或者多个)包含物体的矩形。矩形外的区域被自动认为是背景。
* 对于用户定义的矩形区域，可用背景中的数据来区分它里面的前景和背景区域。
* 用高斯混合模型(GMM)来对背景和前景建模，并将未定义的像素标记为可能的前景或者背景。
* 图像中的每一个像素都被看做通过虚拟边与周围像素相连接，而每条边都有一个属于前景或者背景的概率，这是基于它与周边像素颜色上的相似性。
* 每一个像素(即算法中的节点)会与一个前景或背景节点连接。  
  在节点完成连接后(可能与背景或前景连接)，若节点之间的边属于不同终端(即一个节点属于前景，另一个节点属于背景)，则会切断他们之间的边，这就能将图像各部分分割出来。下图能很好的说明该算法：  
  

# 二、OpenCV中grabCut算法函数：

**void grabCut(**

InputArray **img**,  
InputOutputArray **mask**,  
Rect **rect**,  
InputOutputArray **bgdModel**,  
InputOutputArray **fgdModel**,  
int **iterCount**,  
int mode = **GC\_EVAL**  
);

参数说明：

img—待分割的源图像，是8位3通道  
mask—掩码图像，如果使用掩码进行初始化，那么mask保存初始化掩码信息；在执行分割的时候，也可以将用户交互所设定的前景与背景保存到mask中，然后再传入grabCut函数；在处理结束之后，mask中会保存结果。mask只能取以下四种值：(若无标记GCD\_BGD或GCD\_FGD，则结果只有GCD\_PR\_BGD或GCD\_PR\_FGD；)

* GCD\_BGD（=0），背景；
* GCD\_FGD（=1），前景；
* GCD\_PR\_BGD（=2），可能的背景；
* GCD\_PR\_FGD（=3），可能的前景。

rect—限定要进行分割的图像范围  
bgdModel—背景模型，如果为None，函数内部会自动创建一个bgdModel；bgdModel必须是单通道浮点型图像，且行数只能为1，列数只能为13x5；  
fgdModel—前景模型，如果为None，函数内部会自动创建一个fgdModel；fgdModel必须是单通道浮点型图像，且行数只能为1，列数只能为13x5；  
iterCount—迭代次数，必须大于0；  
mode—grabCut函数操作方法，可选：

* GC\_INIT\_WITH\_RECT（=0），用矩形窗初始化GrabCut；
* GC\_INIT\_WITH\_MASK（=1），用掩码图像初始化GrabCut；
* GC\_EVAL（=2），执行分割。

输入：图像、被标记好的前景、背景。  
输出：分割图像  
其中输入的前景、背景指的是一种概率，如果你已经明确某一块区域是背景，那么它属于背景的概率为1；当然如果你觉得它有可能背景，但是没有百分百的肯定，这个时候你就要用到高斯模型，对其进行建模，然后估算概率。现在我以下图为例，用户通过交互输入框选区域，前景位于框选区域内，也就是说矩形区域外的全部属于背景，且概率为百分百。然后方框内可能属于前景，概率需要用高斯混合建模求解。

## 案例：

|  |  |
| --- | --- |
| **import** numpy **as** np **import** cv2   *# 鼠标事件的回调函数* **def** on\_mouse(event, x, y, flag, param):  **global** rect  **global** leftButtonDown  **global** leftButtonUp   *# 鼠标左键按下* **if** event == cv2.EVENT\_LBUTTONDOWN:  rect[0] = x  rect[2] = x  rect[1] = y  rect[3] = y  leftButtonDown = **True** leftButtonUp = **False** *# 移动鼠标事件* **if** event == cv2.EVENT\_MOUSEMOVE:  **if** leftButtonDown **and not** leftButtonUp:  rect[2] = x  rect[3] = y   *# 鼠标左键松开* **if** event == cv2.EVENT\_LBUTTONUP:  **if** leftButtonDown **and not** leftButtonUp:  x\_min = min(rect[0], rect[2])  y\_min = min(rect[1], rect[3])   x\_max = max(rect[0], rect[2])  y\_max = max(rect[1], rect[3])   rect[0] = x\_min  rect[1] = y\_min  rect[2] = x\_max  rect[3] = y\_max  leftButtonDown = **False** leftButtonUp = **True** img = cv2.imread(**'../mydata/angrybird.png'**) mask = np.zeros(img.shape[:2], np.uint8)  bgdModel = np.zeros((1, 65), np.float64) *# 背景模型* fgdModel = np.zeros((1, 65), np.float64) *# 前景模型* rect = [0, 0, 0, 0] *# 设定需要分割的图像范围* leftButtonDown = **False** *# 鼠标左键按下* leftButtonUp = **True** *# 鼠标左键松开* cv2.namedWindow(**'img'**) *# 指定窗口名来创建窗口* cv2.setMouseCallback(**'img'**, on\_mouse) *# 设置鼠标事件回调函数 来获取鼠标输入* cv2.imshow(**'img'**, img) *# 显示图片* **while** cv2.waitKey(2) == -1:  *# 左键按下，画矩阵* **if** leftButtonDown **and not** leftButtonUp:  img\_copy = img.copy()  cv2.rectangle(img\_copy, (rect[0], rect[1]), (rect[2], rect[3]), (0, 255, 0), 2)  cv2.imshow(**'img'**, img\_copy)   *# 左键松开，矩形画好* **elif not** leftButtonDown **and** leftButtonUp **and** rect[2] - rect[0] != 0 **and** rect[3] - rect[1] != 0:  rect[2] = rect[2] - rect[0]  rect[3] = rect[3] - rect[1]  rect\_copy = tuple(rect.copy())  rect = [0, 0, 0, 0]  *# 物体分割* cv2.grabCut(img, mask, rect\_copy, bgdModel, fgdModel, 5, cv2.GC\_INIT\_WITH\_RECT)   mask2 = np.where((mask == 2) | (mask == 0), 0, 1).astype(**'uint8'**)  img\_show = img \* mask2[:, :, np.newaxis]  *# 显示图片分割后结果--显示原图* cv2.imshow(**'grabcut'**, img\_show)  cv2.imshow(**'img'**, img)  cv2.waitKey(0) cv2.destroyAllWindows() |  |

## 效果：在原来图片上面绘制一个矩形就可以切割出矩形框选的部分图片，还会去除背景

|  |
| --- |
|  |

## 在opencv官方示例里面也应该grabcut工具，不过他的图片是从命令行参数里面获取，这个程序有点意思，可以慢慢理解学习

|  |
| --- |
| *#!/usr/bin/env python ''' =============================================================================== Interactive Image Segmentation using GrabCut algorithm.  This sample shows interactive image segmentation using grabcut algorithm.  USAGE:  python grabcut.py <filename>  README FIRST:  Two windows will show up, one for input and one for output.   At first, in input window, draw a rectangle around the object using the right mouse button. Then press 'n' to segment the object (once or a few times) For any finer touch-ups, you can press any of the keys below and draw lines on the areas you want. Then again press 'n' to update the output.  Key '0' - To select areas of sure background Key '1' - To select areas of sure foreground Key '2' - To select areas of probable background Key '3' - To select areas of probable foreground  Key 'n' - To update the segmentation Key 'r' - To reset the setup Key 's' - To save the results =============================================================================== D:\programs\Python\Python36X64\python.exe grabcut.py ../mydata/messi5.jpg '''  # Python 2/3 compatibility* **from** \_\_future\_\_ **import** print\_function  **import** numpy **as** np **import** cv2 **as** cv  **import** sys  **class** App():  BLUE = [255,0,0] *# rectangle color* RED = [0,0,255] *# PR BG* GREEN = [0,255,0] *# PR FG* BLACK = [0,0,0] *# sure BG* WHITE = [255,255,255] *# sure FG* DRAW\_BG = {**'color'** : BLACK, **'val'** : 0}  DRAW\_FG = {**'color'** : WHITE, **'val'** : 1}  DRAW\_PR\_BG = {**'color'** : RED, **'val'** : 2}  DRAW\_PR\_FG = {**'color'** : GREEN, **'val'** : 3}   *# setting up flags* rect = (0,0,1,1)  drawing = **False** *# flag for drawing curves* rectangle = **False** *# flag for drawing rect* rect\_over = **False** *# flag to check if rect drawn* rect\_or\_mask = 100 *# flag for selecting rect or mask mode* value = DRAW\_FG *# drawing initialized to FG* thickness = 3 *# brush thickness* **def** onmouse(self, event, x, y, flags, param):  *# Draw Rectangle* **if** event == cv.EVENT\_RBUTTONDOWN:  self.rectangle = **True** self.ix, self.iy = x,y   **elif** event == cv.EVENT\_MOUSEMOVE:  **if** self.rectangle == **True**:  self.img = self.img2.copy()  cv.rectangle(self.img, (self.ix, self.iy), (x, y), self.BLUE, 2)  self.rect = (min(self.ix, x), min(self.iy, y), abs(self.ix - x), abs(self.iy - y))  self.rect\_or\_mask = 0   **elif** event == cv.EVENT\_RBUTTONUP:  self.rectangle = **False** self.rect\_over = **True** cv.rectangle(self.img, (self.ix, self.iy), (x, y), self.BLUE, 2)  self.rect = (min(self.ix, x), min(self.iy, y), abs(self.ix - x), abs(self.iy - y))  self.rect\_or\_mask = 0  print(**" Now press the key 'n' a few times until no further change \n"**)   *# draw touchup curves* **if** event == cv.EVENT\_LBUTTONDOWN:  **if** self.rect\_over == **False**:  print(**"first draw rectangle \n"**)  **else**:  self.drawing = **True** *# cv.circle(self.img, (x,y), self.thickness, self.value['color'], -1)* cv.circle(self.mask, (x,y), self.thickness, self.value[**'val'**], -1)   **elif** event == cv.EVENT\_MOUSEMOVE:  **if** self.drawing == **True**:  *# cv.circle(self.img, (x, y), self.thickness, self.value['color'], -1)* cv.circle(self.mask, (x, y), self.thickness, self.value[**'val'**], -1)   **elif** event == cv.EVENT\_LBUTTONUP:  **if** self.drawing == **True**:  self.drawing = **False** cv.circle(self.img, (x, y), self.thickness, self.value[**'color'**], -1)  cv.circle(self.mask, (x, y), self.thickness, self.value[**'val'**], -1)   **def** run(self):  *# Loading images* **if** len(sys.argv) == 2:  filename = sys.argv[1] *# for drawing purposes* **else**:  print(**"No input image given, so loading default image, lena.jpg \n"**)  print(**"Correct Usage: python grabcut.py <filename> \n"**)  filename = **'../mydata/lena.jpg'** *# self.img = cv.imread(cv.samples.findFile(filename))* self.img = cv.imread(filename)  self.img2 = self.img.copy() *# a copy of original image* self.mask = np.zeros(self.img.shape[:2], dtype = np.uint8) *# mask initialized to PR\_BG* self.output = np.zeros(self.img.shape, np.uint8) *# output image to be shown   # input and output windows* cv.namedWindow(**'output'**)  cv.namedWindow(**'input'**)  cv.setMouseCallback(**'input'**, self.onmouse)  cv.moveWindow(**'input'**, self.img.shape[1]+10,90)   print(**" Instructions: \n"**)  print(**" Draw a rectangle around the object using right mouse button \n"**)   **while**(1):   cv.imshow(**'output'**, self.output)  cv.imshow(**'input'**, self.img)  k = cv.waitKey(1)   *# key bindings* **if** k == 27: *# esc to exit* **break  elif** k == ord(**'0'**): *# BG drawing* print(**" mark background regions with left mouse button \n"**)  self.value = self.DRAW\_BG  **elif** k == ord(**'1'**): *# FG drawing* print(**" mark foreground regions with left mouse button \n"**)  self.value = self.DRAW\_FG  **elif** k == ord(**'2'**): *# PR\_BG drawing* self.value = self.DRAW\_PR\_BG  **elif** k == ord(**'3'**): *# PR\_FG drawing* self.value = self.DRAW\_PR\_FG  **elif** k == ord(**'s'**): *# save image* bar = np.zeros((self.img.shape[0], 5, 3), np.uint8)  res = np.hstack((self.img2, bar, self.img, bar, self.output))  cv.imwrite(**'grabcut\_output.png'**, res)  print(**" Result saved as image \n"**)  **elif** k == ord(**'r'**): *# reset everything* print(**"resetting \n"**)  self.rect = (0,0,1,1)  self.drawing = **False** self.rectangle = **False** self.rect\_or\_mask = 100  self.rect\_over = **False** self.value = self.DRAW\_FG  self.img = self.img2.copy()  self.mask = np.zeros(self.img.shape[:2], dtype = np.uint8) *# mask initialized to PR\_BG* self.output = np.zeros(self.img.shape, np.uint8) *# output image to be shown* **elif** k == ord(**'n'**): *# segment the image* print(**""" For finer touchups, mark foreground and background after pressing keys 0-3  and again press 'n' \n"""**)  **try**:  bgdmodel = np.zeros((1, 65), np.float64)  fgdmodel = np.zeros((1, 65), np.float64)  **if** (self.rect\_or\_mask == 0): *# grabcut with rect* cv.grabCut(self.img2, self.mask, self.rect, bgdmodel, fgdmodel, 1, cv.GC\_INIT\_WITH\_RECT)  self.rect\_or\_mask = 1  **elif** (self.rect\_or\_mask == 1): *# grabcut with mask* cv.grabCut(self.img2, self.mask, self.rect, bgdmodel, fgdmodel, 1, cv.GC\_INIT\_WITH\_MASK)  **except**:  **import** traceback  traceback.print\_exc()   mask2 = np.where((self.mask==1) + (self.mask==3), 255, 0).astype(**'uint8'**)  self.output = cv.bitwise\_and(self.img2, self.img2, mask=mask2)   print(**'Done'**)   **if** \_\_name\_\_ == **'\_\_main\_\_'**:  print(\_\_doc\_\_)  App().run()  cv.destroyAllWindows() |

### 效果：

|  |
| --- |
|  |