

周围的人都比你厉害，你才会慢慢变强

公告

昵称： 山上有风景  
园龄： 1年11个月  
粉丝： 170  
关注： 19  
+加关注

< 2019年12月 >

日	一	二	三	四	五	六
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4
5	6	7	8	9	10	11

搜索

找我看

常用链接

我的随笔  
我的评论  
我的参与  
最新评论  
我的标签

我的标签

STL(18)  
SDN(15)  
ThinkPHP3.2(1)

积分与排名

积分 - 211109  
排名 - 2008

随笔分类

C/C++(74)  
Html(2)  
Java(33)  
Javascript(19)  
OpenCV(29)  
PHP(2)  
Python(155)  
STL泛型编程(18)  
单片机笔记（复习用）(3)  
计算机网络(32)  
其他知识(14)  
设计模式(27)  
数据结构(57)  
数据库(8)  
算法习题(43)  
算法训练营  
随笔所想(4)

## OpenCV---分水岭算法

### 目录

- 推文：
- OpenCV学习(7) 分水岭算法(1)（原理简介简单明了）
  - OpenCV-Python教程:31.分水岭算法对图像进行分割（步骤讲解不错）
- 使用分水岭算法进行图像分割
- （一）获取灰度图像，二值化图像，进行形态学操作，消除噪点
  - （二）在距离变换前加上一部操作：通过对上面形态学去噪点后的图像，进行膨胀操作，可以得到大部分都是背景的区域（原黑色不是我们需要的部分是背景）
  - （三）使用距离变换distanceTransform获取确定的前景色
- 相关知识补充（重点）
- （四）在获取了背景区域和前景区域（其实前景区域是我们的种子，我们将从这里进行灌水，向四周涨水，但是这个需要在markers中表示）后，这两个区域中有未重合部分（注1）怎么办？首先确定这些区域（寻找种子）
  - 开始获取未知区域unknown（栅栏会创建在这一区域），为下一步获取种子做准备
  - （五）获取了这些区域，我们可以获取种子，这是通过connectedComponents实现,获取masker标签，确定的前景区域会在其中显示为以1开始的数据，这就是我们的种子，会从这里开始漫水
- 重点：
- （六）根据未知区域unknown在markers中设置栅栏，并将背景区域加入种子区域，一起漫水
  - （七）根据种子开始漫水，让水漫起来找到最后的漫出点（栅栏边界），越过这个点后各个山谷中水开始合并。注意watershed会将找到的栅栏在markers中设置为-1
  - （八）结果查看
  - （九）全部代码

图形界面编程  
正则表达式(2)  
转载推文(4)

随笔档案

- 2019年11月(5)
- 2019年10月(32)
- 2019年9月(21)
- 2019年7月(10)
- 2019年5月(8)
- 2019年4月(25)
- 2019年3月(8)
- 2019年2月(1)
- 2019年1月(12)
- 2018年12月(19)
- 2018年9月(5)
- 2018年8月(95)
- 2018年7月(78)
- 2018年6月(26)
- 2018年5月(17)
- 2018年4月(22)
- 2018年3月(111)

最新评论

- 1. Re:python---websocket的使用  
网上的方法都不行，  
换成gbk会报如下的错：  
IndexError: index out of range  
--缘分天空0320
- 2. Re:python---websocket的使用  
@ 缘分天空0320太久没用，忘了。  
应该是字符编码问题吧。这类问题网上应该可以很容易找到方法解决。你看看Python的默认编码和代码是不是一致。一般就是gbk和utf8之间出错...  
--山上有风景
- 3. Re:python---websocket的使用  
您好，用了您的代码，报如下错误，  
麻烦问下如何解决呢？Traceback (most recent call last): File "server3.py", line 101, in <module...  
--缘分天空0320
- 4. Re:数据结构 (六) 查找---线性索引查找  
请问最后倒序排序的那个代码怎么实现的？可以发一下吗？  
--Viki-
- 5. Re:SDN实验---Ryu的源码分析  
@ 山上有风景谢谢！ ...  
--iRoy\_33

阅读排行榜

- 1. python---websocket的使用(17207)
- 2. OpenCV---图像二值化(12490)
- 3. OpenCV---模板匹配matchTemplate(11512)
- 4. OpenCV---直线检测(8193)
- 5. python---基础知识回顾 (九) 图形用户界面-----WxPython(7974)


评论排行榜

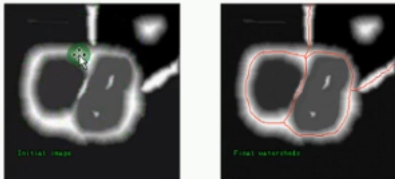
- 1. python---基础知识回顾 (九) 图形用户界面-----Tkinter(4)
- 2. python---websocket的使用(3)

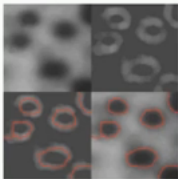
距离变换(Distance Transform)

0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0

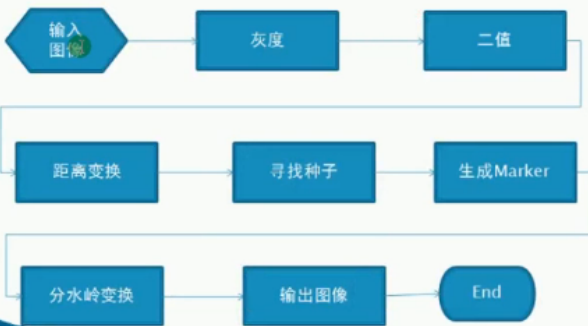
0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	0
0	1	2	2	2	2	2	1
0	1	2	3	3	3	2	1
0	1	2	3	4	3	2	1
0	1	2	3	3	3	2	1
0	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0







基于距离的分水岭分割流程



推文:

OpenCV学习(7) 分水岭算法(1) (原理简介简单明了)

OpenCV-Python教程:31.分水岭算法对图像进行分割 (步骤讲解不错)

使用分水岭算法进行图像分割

(一) 获取灰度图像，二值化图像，进行形态学操作，消除噪点

```
def watershed_demo(image):  
    blur = cv.pyrMeanShiftFiltering(image,10,100)  
    gray = cv.cvtColor(blur,cv.COLOR_BGR2GRAY) #获取灰度图像  
    ret,binary = cv.threshold(gray,0,255,cv.THRESH_BINARY|cv.THRESH_OTSU) #将图像转为黑色和白色部分  
    cv.imshow("binary",binary) #获取二值化图像
```

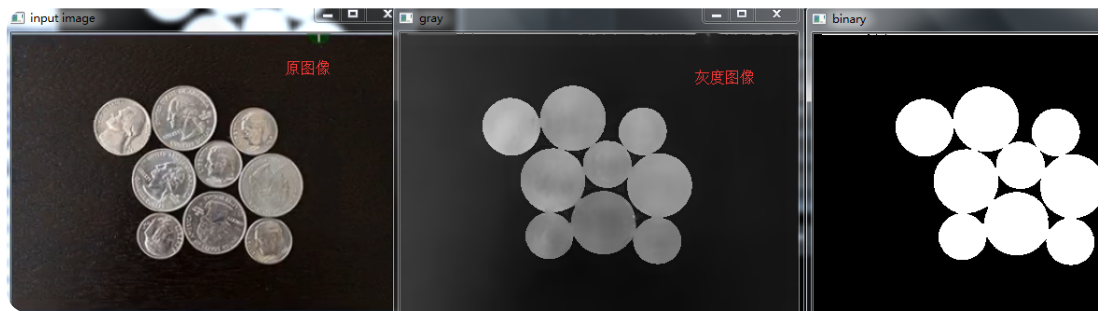
20

关注 | 顶部 | 评论

3. SDN实验---Ryu的源码分析(3)
4. 数据结构 (三) 串---KMP模式匹配算法之获取next数组(2)
5. 数据结构 (四) 树---树的存储结构(1)

#### 推荐排行榜

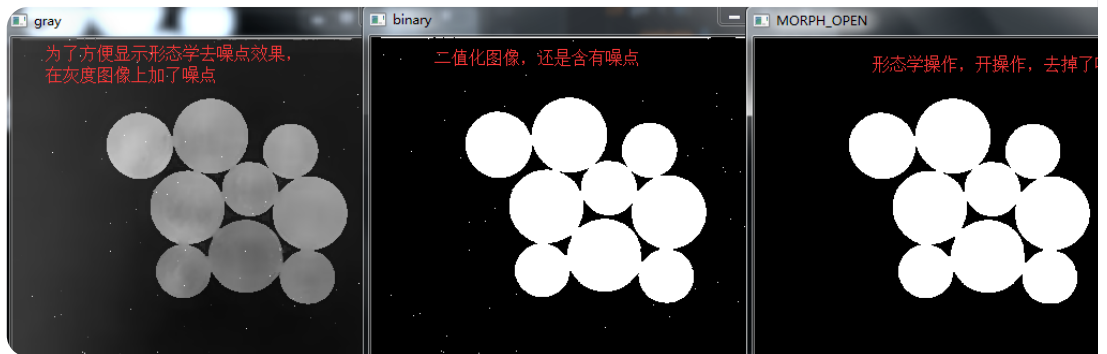
1. 数据结构 (七) 排序---堆排序(11)
2. python---aiohttp的使用(6)
3. python---websocket的使用(4)
4. python---基础知识回顾 (九) 图形用户界面-----WxPython(3)
5. OpenCV---图像金字塔原理(3)



#形态学操作，进一步消除图像中噪点

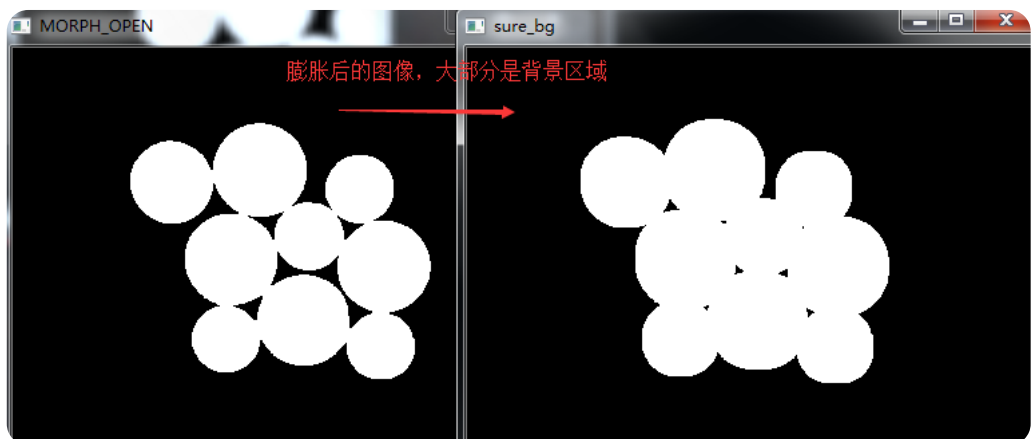
```
kernel = cv.getStructuringElement(cv.MORPH_RECT, (3, 3))
```

```
mb = cv.morphologyEx(binary, cv.MORPH_OPEN, kernel, iterations=2) #iterations连续两次开操作，消除图像的噪点
```



**(二) 在距离变换前加上一步操作：通过对上面形态学去噪点后的图像，进行膨胀操作，可以得到大部分都是背景的区域（原黑色不是我们需要的部分是背景）**

```
sure_bg = cv.dilate(mb, kernel, iterations=3) #3次膨胀，可以获取到大部分都是背景的区域
```



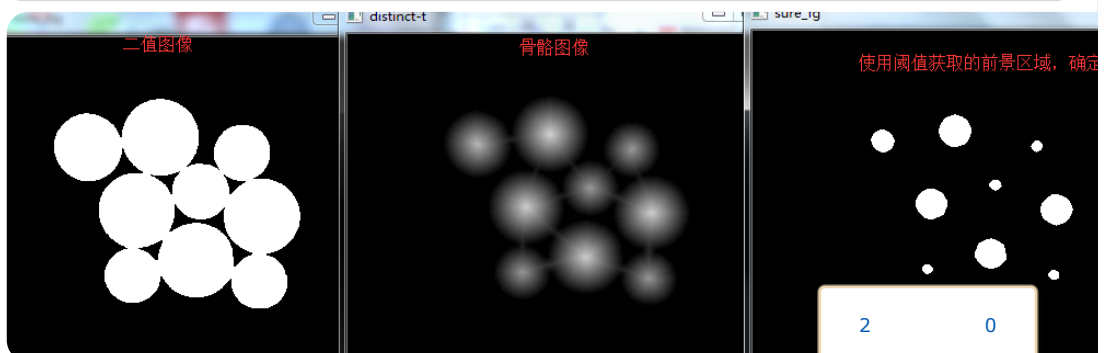
### (三) 使用距离变换distanceTransform获取确定的前景色

根据distanceTransform获取距离背景最小距离的结果（详细看下面相关知识补充）

根据distanceTransform操作的结果，设置一个阈值，使用threshold决定哪些区域是前景，这样得到正确结果的概率很高

```
dist = cv.distanceTransform(mb, cv.DIST_L2, 5) #获取距离数据结果
```

```
ret, sure_fg = cv.threshold(dist, dist.max() * 0.6, 255, cv.THRESH_BINARY) #获取前景色
```



关注 | 顶部 | 评论

## 相关知识补充 (重点)

### (1) 距离变换原理

推文：图像识别中距离变换的原理及作用详解,并附上OpenCV中的distanceTransform实现距离变换的代码！（距离变换的定义讲得不错）

距离变换的处理图像通常都是二值图像，而二值图像其实就是把图像分为两部分，即背景和物体两部分，物体通常又称为前景目标！

通常我们把前景目标的灰度值设为255，即白色

背景的灰度值设为0，即黑色。

所以定义中的非零像素点即为前景目标，零像素点即为背景。

所以图像中前景目标中的像素点距离背景越远，那么距离就越大，如果我们用这个距离值替换像素值，那么新生成的图像中这个点越亮。



再通过设定合理的阈值对距离变换后的图像进行二值化处理，则可得到去除手指的图像（如下图“bidist”窗口图像所示），手掌重心即为该图像的几何中心。



### (2) distanceTransform函数

主要用于计算非零像素到最近零像素点的最短距离。一般用于求解图像的骨骼

```
def distanceTransform(src, distanceType, maskSize, dst=None, dstType=None): # real signature unknown;
    restored from __doc__
```

src: 输入的图像，一般为二值图像

distanceType: 所用的求解距离的类型，有CV\_DIST\_L1, CV\_DIST\_L2, 或 CV\_DIST\_C

mask\_size: 距离变换掩模的大小，可以是 3 或 5。对 CV\_DIST\_L1 或 CV\_DIST\_C 的情况，参数值被强制设定为 3，因为 3×3 mask 给出 5×5 mask 一样的结果，而且速度还更快。

### (3) 若是想骨骼显示（对我们的分水岭流程无影响），我们需要对distanceTransform返回的结果进行归一化处理，使用normalize

因为distanceTransform返回的图像数据是浮点数值，要想在浮点数值表示的颜色空间中，数值范围必须是0-1.0，所以要将其中的数值进行归一化处理

**(重点)** 在整数表示的颜色空间中，数值范围是0-255，但在浮点数值表示的颜色空间中，数值范围是0-1.0，所以要把0-255归一化。

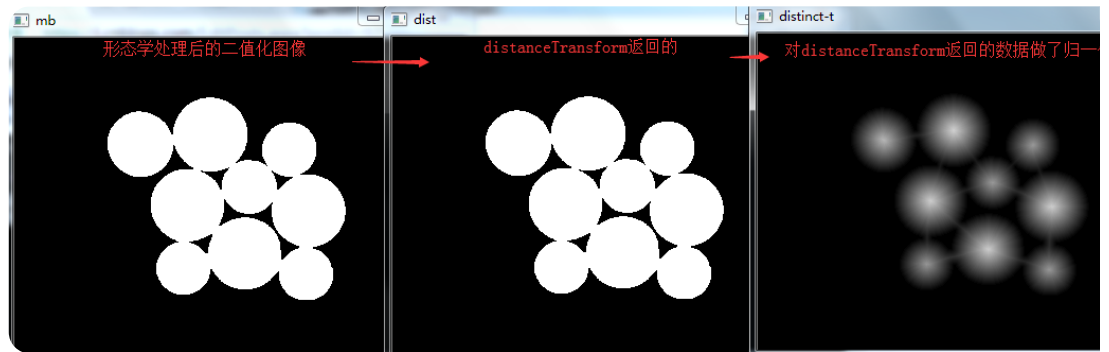
顺便补充：若是不做归一化处理，数值大于1.0的都会变为1.0处理

```
mb = cv.morphologyEx(binary,cv.MORPH_OPEN,kernel,iterations=2) #iterations连续两次开操作
cv.imshow("mb", mb) #这是我们形态学开操作过滤噪点后的图像，暂时可以看做源图像
#距离变换
dist = cv.distanceTransform(mb,cv.DIST_L2,5) #这是我们获取的字段距离数值，对应每个像素都有，所以数组结构和图像数组一致
cv.imshow("dist",dist)
dist_output = cv.normalize(dist,0,1.0,cv.NORM_MINMAX) #归一化的距离图像数组
cv.imshow("distinct-t",dist_output*50)
```

2

0

[关注](#) | [顶部](#) | [评论](#)



发现了似乎distanceTransform返回的图像和源图像一样，似乎出错了

原因：因为distanceTransform返回的是浮点型色彩空间，而dist中存放的数距离0值的最小距离，大多是大于1.0的数值，而上面提到浮点型色彩空间数值范围0~1.0，当数值大于1.0都会被设置为1.0，显示白色，所以和原来的二值化图像一致，我们要想显示骨骼，必须先进行归一化处理

下面是从二值化图像源，distanceTransform距离数组，和归一化距离数组中获取的一段像素数组

```
print(mb[150][120:140])
print(dist[150][120:140])
print(dist_output[150][120:140])
```



整数型色彩空间二值化图像

```
[ 0 0 0 0 0 0 0 0 0 0 0 255 255 255 255 255 255 255 255 255
 255 255]
```

浮点型色彩空间最小距离数组，由于数值大于1.0都会被设置为1.0，所以和上面二值化图像一致

```
[ 0. 0. 0. 0. 0. 0. 0.
 0. 0. 1. 1.4 2.1969 3.1969 4.1969
 5.1969 6.1969 7.1969 8.196899 9.196899 10.187599]
```

浮点型色彩空间归一化数组图像，显示骨骼

```
[0. 0. 0. 0. 0. 0.
 0. 0. 0. 0.00047065 0.0006589 0.00103396
 0.00150461 0.00197525 0.00244589 0.00291654 0.00338719 0.00385783
 0.00432847 0.00479474]
```

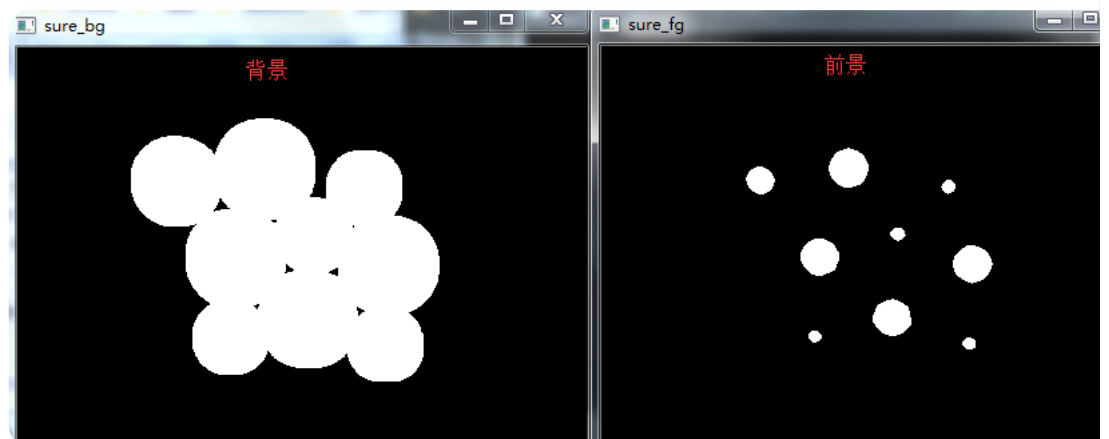


**(四) 在获取了背景区域和前景区域（其实前景区域是我们的种子，我们将从这里进行灌水，向四周涨水，但是这个需要在markers中表示）后，这两个区域中有未重合部分（注1）怎么办？首先确定这些区域（寻找种子）**

注1:

这里是求取硬币偏白色，使用THRESH\_BINARY，所以我们获取对象是白色区域，是获取未重合部分

若是我们求取树叶等偏黑，需要使用THRESH\_BINARY\_INV，此时我们获取的对象是黑色区域，就变为了获取重合部分了



**开始获取未知区域unknown（栅栏会创建在这一区域），为下一步获取种子做准备**

```
surface_fg = np.uint8(sure_fg) #保持色彩空间一致才能进行运算，现在是背景空间为整型空间，前景为浮点型空间，所以进行转换
unknown = cv.subtract(sure_bg, surface_fg)
cv.imshow("unknown", unknown)
```



使用print查看背景前景色彩空间不同

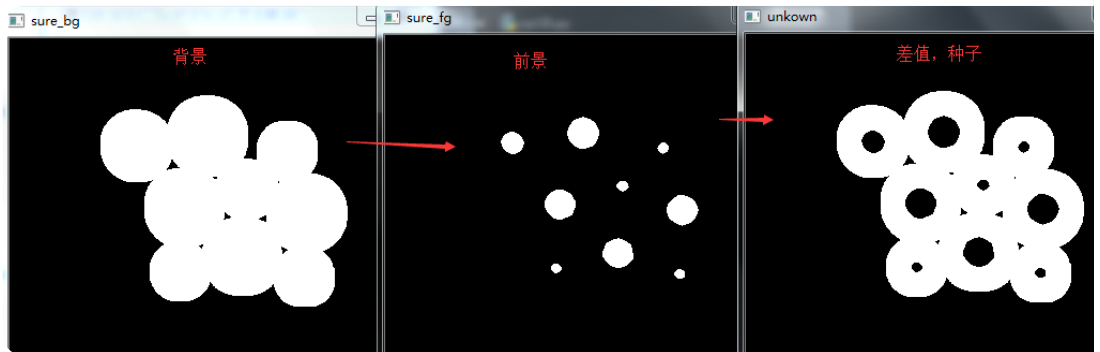
```
print(sure_fg[150][120:140])
```

2 0

关注 | 顶部 | 评论

```
print(sure_bg[150][120:140])
```

```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
[ 0 0 0 0 0 0 255 255 255 255 255 255 255 255 255 255 255
 255 255]
```



(五) 获取了这些区域，我们可以获取种子，这是通过connectedComponents实现，获取masker标签，确定的前景区域会在其中显示为以1开始的数据，这就是我们的种子，会从这里开始漫水

推文: <http://m.imooc.com/article/32675>

## 推文：基于矩阵实现的Connected Components算法

利用connectedComponents求图中的连通图

**重点:**



现在知道了那些是背景那些是硬币（确定的前景区域）了。

那我们就可以**创建标签**（一个与原图像大小相同，数据类型为 `in32` 的数组），并标记其中的区域了。

对我们已经确定分类的区域（无论是前景还是背景）使用不同的正整数标记，对我们不确定的区域（unknown区域）使用 0 标记。

我们可以使用函数 `cv2.connectedComponents()` 来做这件事。

它会把对标签进行操作，将背景标记为 0，其他的对象使用从 1 开始的正整数标记（其实这就是我们的种子，水漫时会从这里漫出）。然后将这个标签返回给我们 markers

但是，我们知道如果背景标记为 0，那分水岭算法就会把它当成未知区域了。（我们要将未知区域标记为 0，所以我们要将背景区域变为其他整数，例如+1）

所以我们想使用不同的整数标记它们。

而对不确定的区域（函数`cv2.connectedComponents` 输出的结果中使用 `unknown` 定义未知区域）标记为 0。



```
#获取mask
```

```
ret, markers = cv.connectedComponents(surface fg)
```

函数原型:

```
def connectedComponents(image, labels=None, connectivity=None, ltype=None): # real signature unknown;
    restored from doc
```

### 参数:

参数image是需要进行连通域处理的二值图像，其他的这里用不到

**返回值:**

`ret`是连通域处理的边缘条数，是上面提到的确定区域（出去背景外的其他确定区域：就是前景），就是种子数，我们会从种子开始向外涨水

`markers`是我们创建的一个标签（一个与原图像大小相同，数据类型为 `in32` 的数组），其中包含有我们原图像的确切区域的数据（前景区域）

**查看部分markers: (0代表的是背景色, )**



```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2  
景区区域，就是我们要找的种子  
2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1]
```

### #0是我们的背景区域

#像这些|

0

[关注](#) | [顶部](#) | [评论](#)





```

blur = cv.pyrMeanShiftFiltering(image,10,100)
gray = cv.cvtColor(blur,cv.COLOR_BGR2GRAY) #获取灰度图像

ret,binary = cv.threshold(gray,0,255,cv.THRESH_BINARY|cv.THRESH_OTSU)
#形态学操作, 进一步消除图像中噪点
kernel = cv.getStructuringElement(cv.MORPH_RECT,(3,3))
mb = cv.morphologyEx(binary,cv.MORPH_OPEN,kernel,iterations=2) #iterations连续两次开操作
sure_bg = cv.dilate(mb,kernel,iterations=3) #3次膨胀,可以获取到大部分都是背景的区域
cv.imshow("sure_bg",sure_bg)

#距离变换
dist = cv.distanceTransform(mb,cv.DIST_L2,5)
cv.imshow("dist",dist)
dist_output = cv.normalize(dist,0,1.0,cv.NORM_MINMAX)
# print(mb[150][120:140])
# print(dist[150][120:140])
# print(dist_output[150][120:140])
cv.imshow("distinct-t",dist_output*50)
ret, sure_fg = cv.threshold(dist,dist.max()*0.6,255,cv.THRESH_BINARY)
cv.imshow("sure_fg",sure_fg)
# print(sure_fg[150][120:140])
# print(sure_bg[150][120:140])
#获取未知区域
surface_fg = np.uint8(sure_fg) #保持色彩空间一致才能进行运算, 现在是背景空间为整型空间, 前景为浮点型空间, 所以进行转换
unknown = cv.subtract(sure_bg,surface_fg)
cv.imshow("unkown",unknown)
#获取maskers,在maskers中含有种子区域
ret,markers = cv.connectedComponents(surface_fg)
#print(ret)

#分水岭变换
markers = markers + 1
markers[unknown==255] = 0

markers = cv.watershed(image,markers=markers)
image[markers==-1] = [0,0,255]

cv.imshow("result",image)

src = cv.imread("./c.png") #读取图片
cv.namedWindow("input image",cv.WINDOW_AUTOSIZE) #创建GUI窗口,形式为自适应
cv.imshow("input image",src) #通过名字将图像和窗口联系

watershed_demo(src)

cv.waitKey(0) #等待用户操作, 里面等待参数是毫秒, 我们填写0, 代表是永远, 等待用户操作
cv.destroyAllWindows() #销毁所有窗口

```

作者: 山上有风景

欢迎任何形式的转载, 但请务必注明出处。

限于本人水平, 如果文章和代码有表述不当之处, 还请不吝赐教。

分类: OpenCV

好文要顶

关注我

收藏该文



山上有风景

关注 - 19

粉丝 - 170

+加关注

« 上一篇: OpenCV---其他形态学操作

» 下一篇: OpenCV---人脸检测

posted @ 2018-07-09 13:32 山上有风景

2

0

收藏

关注 | 顶部 | 评论

评论列表



#1楼 2018-10-07 16:06 元气少女缘结神



博主很厉害。向你学习。

支持(0) 反对(0)

[刷新评论](#) [刷新页面](#) [返回顶部](#)

注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问](#) 网站首页。

Copyright © 2019 山上有风景

Powered by .NET Core 3.1.0 on Linux

2 0

[关注](#) | [顶部](#) | [评论](#)