opencv_basic Documentation

发布 *v1.0*

yuanjh

2020 年 10 月 31 日

Contents:

opencv 个人学习笔记，之前并未有视觉经验，且工作中使用场景有限且较为粗浅，只知道大概执行效果即可，并未做深入学习

CHAPTER 1

opencv 小白 01 学习笔记 01

1.1 学习目标

1,opencv 能做什么，不能做什么

2, 阅读代码，知道某种 func 后图片怎么样了

3, 在图片相关机器学习算法预处理阶段，希望通过对图片的简单处理，达到提升训练效果的目的。

1.2 OpenCV-Python 教程:2.Images

https://www.jianshu.com/p/35712839830

· 打开图片，显示，保存图片

· 这些函数：cv2.imread(), cv2.imshow(), cv2.imwrite()

cv2.waitKey() 是一个键盘绑定函数。它的参数是毫秒数，这个函数会等待任意键盘事件指定的毫秒时间。如果你点了任意键，这个程序继续。如果传入 0，它会一直等待按键。它也可以设置成检测指定键，比如如果 a 被按了

cv2.destroyAllWindows() 销毁所有的我们创建的窗口，如果你想销毁指定的窗口，使用函数 cv2.destroyWindow() 你可以传指定窗口的名字作为参数。

如果你使用 64 位的机器，你需要把 k = cv2.waitKey(0) 这行换成：

```
k = cv2.waitKey(0) & 0xFF
# 图集
imgs = np.hstack([img,img2])
# 展示多个
cv.imshow("mutil_pic", imgs)
```

1.3 OpenCV-Python 教程:3. 视频

https://www.jianshu.com/p/562a936512ae

cap = cv2.VideoCapture(0)

cap.isOpened()

可以使用 cap.get(propId) 来访问这个视频的一些属性，propId 是从 0 到 18 的一个数。每个数字代表了视频的一个属性。

比如，我可以通过 cap.get(3) 和 cap.get(4) 来获得这一帧的宽度和高度。它默认会返回 640x480，但是我想把值修改成 320x240. 只需要用 ret = cap.set(3, 320) 和 ret = cap.set(4, 240)

```
fourcc = cv2.VideoWriter_fourcc(*'XVID')
out = cv2.VideoWriter('output.avi', fourcc, 20.0, (640, 480))
out.write(frame)
out.release()
```

1.4 OpenCV-Python 教程:4. 在 OpenCV 里的绘制函数

https://www.jianshu.com/p/b0adf093e15f

cv2.line(), cv2.circle(), cv2.rectangle(), cv2.ellipse(), cv2.putText()

这些函数里，你会发现一些通用的参数:

· img：你要画形状的图片

· color：形状的颜色。对于 BGR，传一个元组进去，比如（255,0,0）是蓝色。对于灰度图，传一个灰度值。

· thickness：线或者圆的粗细。如果传了-1 给一个封闭图形比如圆，它会充满图形。默认的 thickness = 1

· lineType: 线的类型，比如 8-connected，反锯齿等。默认情况下是 8-connected。cv2.LINE_AA 是反锯齿，在曲线时很好看。

```
img = cv2.line(img,(0,0),(511,511),(255,0,0),5)
img = cv2.rectangle(img,(384,0),(510,128),(0,255,0),3)
```

(下页继续)

(续上页)

```
img = cv2.circle(img,(447,63),63,(0,0,255),-1)
img = cv2.ellipse(img,(256,256),(100,50),0,0,180,255,-1)

pts = np.array([[10,5],[20,30],[70,20],[50,10]],np.int32)
pts = pts.reshape((-1,1,2))
img = cv2.polylines(img,[pts],True,(0,255,255))
```

注意:

如果第三个参数是 False，你会得到一个连接所有点的图形，而不是一个封闭图形。

cv2.polylines() 可以被用来画多条线，值需要建一个包含所有线的列表，然后把它传给函数就行了。所有线都会被独立绘制。这比每条线都调用一次 cv2.line() 更快更好的方法。

font = cv2.FONT_HERSHEY_SIMPLEX

cv2.putText(img, ‘OpenCV’, (10,500), font, 4,(255,255,255), 2, cv2.CV_AA)

1.5 OpenCV-Python 教程:5. 鼠标作为画笔

https://www.jianshu.com/p/e261346db440

```
cv2.setMouseCallback('image',draw_circle)
def draw_circle(event, x, y, flags, param):
    global ix, iy, drawing, mode

    if event == cv2.EVENT_LBUTTONDOWN:
        drawing = True
        ix,iy = x,y
    elif event == cv2.EVENT_MOUSEMOVE:
        if drawing == True:
            if mode == True:
                cv2.rectangle(img, (ix, iy), (x, y),(0,255,0),-1)
            else:
                cv2.circle(img,(x,y),5,(0,0,255),-1)
    elif event == cv2.EVENT_LBUTTONUP:
        drawing = False
        if mode == True:
            cv2.rectangle(img,(ix,iy),(x,y),(0,255,0),-1)
        else:
            cv2.circle(img,(x,y),5,(0,0,255),-1)
```

1.6 OpenCV-Python 教程:7. 图片上的基本操作

https://www.jianshu.com/p/80efbe3880dc

```
>>>b,g,r = cv2.split(img)
>>>img=cv2.merge((b,g,r))
>>>b = img[:,:,0]
```

假设你想把所有的红色像素变成 0，你不用这么分割，你可以简单的使用 Numpy 索引，这样更快

```
>>>img[:,:,2]=0
```

cv2.split() 是一个成本很高的操作（执行时间），所以只在必要的时候使用。Numpy 索引要更有效率，能用就用。

1.7 OpenCV-Python 教程:8. 图片的算术运算

https://www.jianshu.com/p/4c4b4e651989

OpenCV 和 Numpy 相加是不同的。OpenCV 相加是一个渗透运算，而 Numpy 的相加是模运算。

```
>>>x = np.uint8([250])
>>>y = np.uint8([10])
>>>printcv2.add(x,y) ? ? ? # 250+10 = 260 => 255
[[255]]
>>>print x+y ? ? ? ?# 250+10 = 260 % 256 = 4
[4]

cv2.addWeighted()
```

【Python——opencv 篇】bitwise_and、bitwise_not 等图像基本运算及掩膜: https://blog.csdn.net/Lily_9/article/details/83143120

```
# set blue thresh
lower_yellow=np.array([11,43,46])
hsv = cv.cvtColor(cp, cv.COLOR_BGR2HSV)
mask = cv.inRange(hsv, lower_yellow, upper_yellow)
cv.imshow('Mask', mask)
res = cv.bitwise_and(cp, cp, mask=mask)
```

1.8 OpenCV-Python 教程:9. 性能测量和改进技术

https://www.jianshu.com/p/205a7514641a

cv2.getTickCount 函数返回从一个参考时间（比如机器开机的时间）开始到这个函数被调用的时间之间的时钟循环数量。所以如果你在函数执行前调用一次，函数执行完调用一次，你就能得到函数执行用掉的时钟循环。

cv2.getTickFrequency 函数返回时钟频率或者每秒钟的时钟循环数。所以要得到函数执行了多少秒

OpenCV 的默认优化

很多 OpenCV 函数对 SSE2, AVX 等做了优化。当然也有未优化的代码。所以如果我们的系统支持这些特性，我们应该利用他们（基本上现在的主流处理器都支持）。在编译的时候是自动启用的。所以如果启用的话 OpenCV 执行的是优化的代码，你可以用 cv2.useOptimized() 来检查是否启用了，用 cv2.setUseOptimized() 来启用/禁用

Python 标量运算时比 Numpy 标量运算要快的。所以对于包含 1 到两个元素的运算，Python 标量要比 Numpy 数组要快。Numpy 在数组尺寸有点大的时候占优势。

注意:

一般来说，OpenCV 函数比 Numpy 函数要快，所以对于相同的运算，推荐优先使用 OpenCV 函数。但是，也有例外，特别是当 Numpy 操作 views 而不是复制的时候。

1.9 OpenCV-Python 教程:10. 更改颜色空间

https://www.jianshu.com/p/65c1fbd8ae2a、

了解下列函数：cv2.cvtColor(), cv2.inRange()

变更色彩空间

```
# 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)

# Bitwise-AND mask and original image
res = cv2.bitwise_and(frame,frame,mask=mask)

>>>green = np.uint8([[[0,255,0]]])
>>>hsv_green = cv2.cvtColor(green,cv2.COLOR_BGR2HSV)
```

(下页继续)

(续上页)

```
>>>print hsv_green
[[[ 60 255 255]]]
```

CHAPTER 2

opencv 小白 02 学习笔记 02

2.1 OpenCV-Python 教程:11. 图片阈值

https://www.jianshu.com/p/267a32ad0a23

cv2 阈值处理:https://blog.csdn.net/u011070767/article/details/80639556

一、全局阈值

为整个图片指定一个阈值，函数为 cv2.threshold(src, thresh, maxval, type, dst=None)

```
src: 原图 (灰图)
thresh: 阈值
maxval: 给#THRESH_BINARY and #THRESH_BINARY_INV 模式使用的最大值
 type: 二值化的类型
cv2.THRESH_BINARY           超过阈值部分取 maxval (最大值), 否则取 0
cv2.THRESH_BINARY_INV    THRESH_BINARY 的反转
cv2.THRESH_TRUNC            大于阈值部分设为阈值, 否则不变
cv2.THRESH_TOZERO          大于阈值部分不改变, 否则设为 0
cv2.THRESH_TOZERO_INV  THRESH_TOZERO 的反转
```

二、自适应阈值

前面介绍的是全局性的阈值，整个图像的像素都以此阈值为基准。而自适应阈值可以看成是一种局部性的阈值，指定一个区域大小，此区域内的阈值为区域里面像素的平均值（或加权和）减去第六个参数 C

```
adaptiveThreshold(src, maxValue, adaptiveMethod, thresholdType, blockSize, C, dst=None):
src: 原图（灰图）
maxValue: 像素值上限
adaptiveMethod: 自适应方法
 cv2.ADAPTIVE_THRESH_MEAN_C ：领域内均值
 cv2.ADAPTIVE_THRESH_GAUSSIAN_C ：领域内像素点加权和，权重为一个高斯窗口
thresholdType: 只有两个 cv2.THRESH_BINARY 和 cv2.THRESH_BINARY_INV
blockSize:  规定正方形领域的大小
C: 常熟 C，阈值等于指定正方形领域的均值或加权和减去这个常熟
```

2.2 OpenCV-Python 教程:12. 图片的几何转换

https://www.jianshu.com/p/1c6512d475cc

OpenCV 提供了两个转换函数，cv2.warpAffine 和 cv2.warpPerspective，通过他们你可以进行各种转换，cv2.warpAffine 接受 2x3 的转换矩阵二 cv2.warpPerspective 接受 3x3 的转换矩阵做为输入。

OpenCV 有一个函数 cv2.resize() 来干这个，图片的大小可以人工指定，或者你可以指定缩放因子。有不同的差值方式可以使用，推荐的插值方法是缩小时用 cv2.INTER_AREA，放大用 cv2.INTER_CUBIC(慢) 和 cv2.INTER_LINEAR。默认情况下差值使用 cv2.INTER_LINEAR

```
res = cv2.resize(img, None, fx=2, fy=2, interpolation=cv2.INTER_CUBIC)

#OR

height, width = img.shape[:2]

res = cv2.resize(img, (2*width, 2*height), interpolation=cv2.INTER_CUBIC)
```

平移是改变物体的位置。你可以把它变成 Numpy 的数组，类型是 np.float32 的，然后把它传给 cv2.warpAffine() 函数

```
M = np.float32([[1,0,100],[0,1,50]])

dst = cv2.warpAffine(img, M, (cols,rows))
```

旋转

```
M = cv2.getRotationMatrix2D((cols/2,rows/2), 90, 1)

dst = cv2.warpAffine(img, M, (cols, rows))
```

仿射变换

```
M = cv2.getAffineTransform(pts1, pts2)

dst = cv2.warpAffine(img,M,(cols, rows))
```

透视变换

对于透视变换，你需要一个 3x3 的转换矩阵。转换后直线仍然保持直线

cv2.getPerspectiveTransform 函数就能得到转换矩阵了，再用 cv2.warpPerspective 来接收这个 3x3 的转换矩阵。

```
pts1 = np.float32([[56,65],[368,52],[28,387],[389,390]])
pts2 = np.float32([[0,0],[300,0],[0,300],[300,300]])
M = cv2.getPerspectiveTransform(pts1,pts2)
dst = cv2.warpPerspective(img, M,(300,300))
```

2.3 OpenCV-Python 教程:13. 平滑图片

https://www.jianshu.com/p/451c52a74ddb

```
kernel = np.ones((5,5), np.float32)/25

dst = cv2.filter2D(img, -1, kernel)
```

图片模糊 (图片平滑)

1. 平均

这个方法是用一个标准化的箱式过滤器来卷积。它简单的把核区域周围的像素的平均替换中心元素。

这个是用的 cv2.blur() 或者 cv2.boxFilter()。

如果你不想用标准化箱式过滤器，使用 cv2.boxFilter() 然后传参数 normalize=False 给函数。

blur = cv2.blur(img,(5,5))

2. 高斯滤波

在这个方法里，使用一个高斯核。函数是 cv2.GaussianBlur()。

blur = cv2.GaussianBlur(img,(5,5),0)

3. 中值滤波

这里函数 cv2.medianBlur() 计算核窗口下的所有像素的中值来替换中心像素点。这个特别适合去除椒盐噪点

在高斯和箱式过滤中，给中心点用的过滤的值可以是在原图中没有的值，但是在中值滤波中不同，因为中心元素总是被图片里的某个像素值替代。这个方法在去除噪音上很高效。核的大小必须是正奇数。

median = cv2.medianBlur(img,5)

4. 双边滤波

我们之前展示过的滤波器都是倾向于模糊边界的。但是双边滤波不是这样，cv2.bilateralFilter()，在保持边界的情况下去除噪点非常有效。但是这个操作比其他滤波器都慢一些。

blur = cv2.bilateralFilter(img,9,75,75)

2.4 OpenCV-Python 教程:14. 形态变换

https://www.jianshu.com/p/dcecaf62da71

1. 腐蚀

腐蚀的基本理念就和土壤腐蚀一样，它会腐蚀掉前景的边缘（所以前景应该用白色）。

erosion = cv2.erode(img,kernel,iterations=1)

2. 膨胀

这个就是腐蚀的反义词，在核下只要有至少一个像素是 1，像素的值就是 1. 所以它会增加图片上白色区域的范围或者前景物体的大小。

dilation = cv2.dilate(img,kernel,iterations=1)

3. 开

开就是腐蚀之后再膨胀的另一个名字。我们使用函数 cv2.morphologyEx()

opening = cv2.morphologyEx(img,cv2.MORPH_OPEN,kernel)

4. 闭

闭是开的反义词，膨胀之后再腐蚀，在用来关闭前景对象里的小洞或小黑点很有用。

closing = cv2.morphologyEx(img,cv2.MORPH_CLOSE,kernel)

5. 形态梯度

这个和腐蚀以及膨胀不同，结果看上去像是物体的轮廓。

gradient = cv2.morphologyEx(img,cv2.MORPH_GRADIENT,kernel)

6. 顶帽

这个是输入图片和图片的开运算结果的差别，下面是 9x9 的核的

tophat=cv2.morphologyEx(img,cv2.MORPH_TOPHAT,kernel)

7. 黑帽

这是输入图片的闭的结果和输入图片的差别。

blackhat=cv2.morphologyEx(img,cv2.MORPH_BLACKHAT,kernel)

2.5 OpenCV-Python 教程:15. 图片梯度

https://www.jianshu.com/p/e7d466446a06

图像梯度的基本原理：https://blog.csdn.net/saltriver/article/details/78987096，

当用均值滤波器降低图像噪声的时候，会带来图像模糊的副作用。我们当然希望看到的是清晰图像。那么，清晰图像和模糊图像之间的差别在哪里呢？从逻辑上考虑，图像模糊是因为图像中物体的轮廓不明显，轮廓边缘灰度变化不强烈，层次感不强造成的，那么反过来考虑，**轮廓边缘灰度变化明显些，层次感强些是不是图像就更清晰**些呢。

sobel 算子，scharr 算子，Laplacian 算子:https://blog.csdn.net/naruhina/article/details/104710805

2.6 OpenCV-Python 教程:16.Canny 边缘检测

https://www.jianshu.com/p/f91a7b8e5285

Canny 边缘检测是一个很流行的边缘检测算法。由 John F.Canny 在 1986 年开发。这是一个多步骤的算法。

```
1.降噪
2.找到图片中的亮度梯度
3.非最大值抑制
4.滞后阈值
```

OpenCV 把所有这些放在一个函数里，edges = cv2.Canny(img,100,200)。我们来看看怎么用它，第一个参数是输入图片，第二个和第三个参数是我们的 minVal 和 maxVal。aperture_size 参数是索贝尔核的大小，用来找图片的梯度。默认是 3，最后一个参数是 L2gradient，用来指定寻找梯度幅值的公式。如果为 True，会使用上面提到的更准确的公式，否则会用下面这个函数，默认情况下为 False:
$Edge\_Gradient\ (G) = |G_x| + |G_y|$

2.7 OpenCV-Python 教程:17. 图像金字塔

https://www.jianshu.com/p/a06e38691dca

处理一个图像的不同分辨率的图片。比如在**搜索图像里的某些元素的时候**，比如脸，我们并不确认目标在图片里的大小。在这种情况下，我们可能需要创建一系列的不同分辨率的图片来在其中寻找目标。这些不同分辨率的图片叫做图片金字塔（因为他们从小到大堆在一起的时候像个金字塔）有两种图像金字塔 1）高斯金字塔 2）拉普拉斯金字塔 **高斯金字塔的高级（低分辨率）是从低级别（高分辨率）的图像里移除连续的行和列来形成的**。高级别理的每个像素是下级 5 个高斯权重的像素得到的。**拉普拉斯金字塔式从高斯金字塔得到的**，没有单独的函数。

```
img = cv2.imread('messi5.jpg')
lower_reso = cv2.pyrDown(higher_reso)
higher_reso2 = cv2.pyrUp(lower_reso)
```

2.8 OpenCV-Python 教程:18. 图像轮廓

https://www.jianshu.com/p/4f790fb18691

轮廓可以被简单解释为一个**连接所有连续点的曲线（沿边界），有同样的颜色和亮度**。轮廓在做**形状分析和目标检测与识别**都很有用。

```
为了更好地额准确率，使用二进制图像，所以在找轮廓前，使用阈值或者 canny 边缘检测。
findContours 函数修改原图。所以如果你想在找轮廓后还需要原图，把它存到别的变量里。
在 OpenCV 里，找轮廓和在黑色背景里找白色目标一样，所以记住，目标应该是白的而背景是黑色的。

im = cv2.imread('test.jpg')
imgray = cv2.cvtColor(im,cv2.COLOR_BGR2GRAY)
ret,thresh = cv2.threshold(imgray,127,255,0)
image, contours, hierarchy = cv2.findContours(thresh,cv2.RETR_TREE,cv2.CHAIN_APPROX_
↪SIMPLE)
```

要绘制轮廓，可以用 cv2.drawContours 函数。如果你有图形的边界点，也可以用来绘制任何形状。

要画一个图像的所有轮廓：

```
img=cv2.drawContours(img,contours,-1,(0,255,0),3)
```

要画第四级轮廓：

```
img=cv2.drawContours(img,contours,3,(0,255,0),3)
```

但大多数时候，下面的更有用：

```
cnt=contours[4]img=cv2.drawContours(img,[cnt],0,(0,255,0),3)
```

Contour 近似方法

这是 cv2.findContours 函数的参数，它实际是指什么呢?

轮廓是图形的边界。它存了边界坐标 (x,y)，但是它存了所有坐标么？这个就是轮廓近似方法指定的。

如果你传 cv2.CHAIN_APPROX_NONE，所有的边界点都会存下来。但是实际上我们需要所有的点么？比如说，你发现一个直线的轮廓，你需要这线上的所有点来表示这个线么？不需要，我们只需要两个端点就够了。**这就是 cv2.CHAIN_APPROX_SIMPLE 要做的。它会去掉所有冗余点来压缩轮廓，节省内存。**

2.9 OpenCV-Python 教程:19. 轮廓属性

https://www.jianshu.com/p/6bde79df3f9d

1 图像矩

帮你计算一些属性，比如重心，面积等。

函数 cv2.moments() 会给你一个字典，包含所有矩值

2. 轮廓面积

轮廓面积由函数 cv2.contourArea() 得到或者从矩里得到 M[‘m00’]

3. 轮廓周长

可以用 cv2.arcLength() 函数得到。第二个参数指定形状是否是闭合的轮廓（如果传 True)。或者只是一个曲线。

4. 轮廓近似

这会把轮廓形状近似成别的边数少的形状，边数由我们指定的精确度决定。这是 Douglas-Peucker 算法的实现。

```
approx = cv2.approxPolyDP(cnt,epsilon,True)
```

5. 凸形外壳

凸形外壳和轮廓近似类似，但是还不一样（某些情况下两个甚至提供了同样的结果)。

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

6. 检查凸面

有一个函数用来检查是否曲线是凸面, cv2.isContourConvex(). 它返回 True 或 False。

```
k=cv2.isContourConvex(cnt)
```

7. 边界矩形

有两种边界矩形

7.a. 正边界矩形

```
x,y,w,h = cv2.boundingRect(cnt)
img = cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)
```

7.b. 渲染矩形

```
rect = cv2.minAreaRect(cnt)
box = cv2.boxPoints(rect)
box = np.int0(box)
im = cv2.drawContours(im,[box],0,(0,0,255),2)
```

这个边界矩形是用最小面积画出来的，所以要考虑旋转。函数是 cv2.minAreaRect()。它返回一个 Box2D 结构，包含了 (左上角 (x,y)，(width, height)，旋转角度)。但是要画这个矩形我们需要 4 个角。这四个角用函数 cv2.boxPoints() 得到

8. 最小闭包圆

我们找一个目标的外接圆可以用函数 cv2.minEnclosingCircle(). 这个圆用最小面积完全包围目标。

```
(x,y),radius = cv2.minEnclosingCircle(cnt)
center = (int(x),int(y))
radius = int(radius)
img = cv2.circle(img,center,radius,(0,255,0),2)
```

9. 椭圆

用一个椭圆来匹配目标。它返回一个旋转了的矩形的内接椭圆

```
ellipse=cv2.fitEllipse(cnt)
im=cv2.ellipse(im,ellipse,(0,255,0),2)
```

1. 直线

类似的我们可以匹配一根直线，下面的图像包含一系列的白色点，我们可以给它一条近似的直线。

```
rows,cols = img.shape[:2]
[vx,vy,x,y] = cv2.fitLine(cnt, cv2.DIST_L2,0,0.01,0.01)
lefty = int((-x*vy/vx) + y)
righty = int(((cols-x)*vy/vx)+y)
img = cv2.line(img,(cols-1,righty),(0,lefty),(0,255,0),2)
```

2.10 OpenCV-Python 教程:20. 轮廓属性

https://www.jianshu.com/p/0d5d357840e6

我们要得到一些目标有用的属性，比如当量直径

1. 高宽比

这是目标的边界矩形的宽高比

```
x,y,w,h=cv2.boundingRect(cnt)
aspect_ratio=float(w)/h
```

2.Extent

Extent 是轮廓面积和边界矩形面积的比率

3.Solidity

是轮廓面积和凸形外壳面积的比率

4. 等价半径

是面积和轮廓面积一样的圆的半径

```
area=cv2.contourArea(cnt)
equi_diameter=np.sqrt(4*area/np.pi)
```

5. 方向

目标的方向角度。下面的方法可以得到长轴和短轴长度

```
(x,y),(MA,ma),angle=cv2.fitEllipse(cnt)
```

1.

在某些情况下，我们可能需要构成目标的所有点。

```
mask = np.zeros(imgray.shape,np.uint8)
cv2.drawContours(mask,[cnt],0,255,-1)
pixelpoints = np.transpose(np.nonzero(mask))
#pixelpoints = cv2.findNonZero(mask)
```

这里，两个方法，一个使用 Numpy 函数，另一个使用 OpenCV 函数（最后的注释行）达到同样目的。结果也是相同的。不同的一点是 Numpy 给的坐标是 (row, column) 格式，而 OpenCV 给的坐标是 (x, y) 格式，所以基本上结果可以互相转换。row = x ，column = y

7. 最大值，最小值以及他们的位置

```
min_val,max_val,min_loc,max_loc=cv2.minMaxLoc(imgray,mask=mask)
```

8. 平均颜色和平均强度

我们可以得到目标的平均颜色。或者是灰度模式下的平均亮度。再次使用了 mask image

```
mean_val=cv2.mean(im,mask=mask)
```

9. 端点

端点表示最高点，最低点，最左和最右点。

```
leftmost = tuple(cnt[cnt[:,:,0].argmin()][0])
rightmost=tuple(cnt[cnt[:,:,0].argmax()][0])
topmost=tuple(cnt[cnt[:,:,1].argmin()][0])
bottommost=tuple(cnt[cnt[:,:,1].argmax()][0])
```

CHAPTER 3

opencv 小白 03 学习笔记 03

3.1 OpenCV-Python 教程:21. 轮廓：更多函数

https://www.jianshu.com/p/07312149e60a

1. 凸面缺陷

OpenCV 提供了现成的函数来做这个，cv2.convexityDefects().

2.Point Polygon Test

这个函数找到图像里的点和轮廓之间的最短距离。它返回的距离当点在轮廓外的时候是负值，当点在轮廓内是正值，如果在轮廓上是 0

3. 匹配形状

OpenCV 提供一个函数 cv2.matchShapes() 来让我们可以比较两个形状，或者两个轮廓来返回一个量表示相似度。结果越低，越相似，它是根据 hu 矩来计算的。不同的计算方法在文档里有介绍。

3.2 OpenCV-Python 教程:22. 轮廓层级

https://www.jianshu.com/p/4daf1f7e69e0

什么是层级?

一般来说我们用 cv2.findContours() 函数来检测图像里的目标，有时候目标在不同的地方，但是在有些情况下，有些图形在别的图形里面，就像图形嵌套，在这种情况下，我们把外面那层图形叫做 parent，里面的叫

child。这样图形里的轮廓之间就有了关系。我们可以指定一个轮廓和其他之间的是如何连接的，这种关系就是层级。

OpenCV 里的层级表示

每个轮廓有他自己的关于层级的信息，谁是他的孩子，谁是他的父亲等。OpenCV 用一个包含四个值得数组来表示：[Next, Previous, First_Child, Parent]

我们知道了层级，现在来看 OpenCV 里的轮廓获取模式，四个标志 cv2.RETR_LIST, cv2.RETR_TREE, cv2.RETR_CCOMP, cv2.RETR_EXTERNAL 表示啥?

轮廓获取模式

1.RETR_LIST

这是最简单的一个，它获取所有轮廓，但是不建立父子关系，他们都是一个层级。所以，层级属性第三个和第四个字段（父子）都是-1，但是 Next 和 Previous 还是有对应值。

2.RETR_EXTERNAL

如果用这个模式，它返回最外层的。所有孩子轮廓都不要，我们可以说在这种情况下，只有家族里最老的会被照顾，其他都不管。所以在我们的图像里，有多少最外层的轮廓呢，有 3 个，contours 0,1,2

3.RETR_CCOMP

这个模式获取所有轮廓并且把他们组织到一个 2 层结构里，对象的轮廓外边界在等级 1 里，轮廓内沿（如果有的话）放在层级 2 里。如果别的对象在它里面，里面的对象轮廓还是放在层级 1 里，它的内沿在层级 2.

4.RETR_TREE

最后，Mr.Perfect。它取回所有的轮廓并且创建完整的家族层级列表，它甚至能告诉你谁是祖父，父亲，儿子，孙子。。

3.3 OpenCV-Python 教程:23.histogram

https://www.jianshu.com/p/c3f414646a50

什么是 histogram? 它可以给出图像的密度分布的总体概念，它的 x 轴是像素值（0 到 255）y 轴是对应的像素在图像里的数量。

看 histogram 你可以得到对比度，亮度，密度分布等直观信息。

1.OpenCV Histogram 计算

2.Numpy 里的 Histogram 计算

OpenCV 函数要比 np.histogram() 要快很多（40x）。所以还是用 OpenCV 函数。

使用 Mask

我们使用 cv2.calcHist() 来找整个图的 histogram。如果你想找某个区域的 histogram，就创建一个你想要的区域是白色而其他地方是黑色的 mask 图像。

3.4 OpenCV-Python 教程:24.histogram-2:histogram 均衡

https://www.jianshu.com/p/7494f40e722b

一张像素值被限制在一个特定值范围内的图像，比如，亮图被限制所有像素都是亮值。但是一个好的图片应该是有所有范围的像素。所以我们需要把 histogram 拉伸到两端，这就是 histogram 均衡。这个一般是用来提升图片的对比度。

在面部识别里，在训练面部数据之前，面部的图片会做 histogram 均衡以让他们所有都是同样的亮度条件。

OpenCV 里的 Histograms 均衡

OpenCV 有一个函数可以做这个，cv2.equalizeHist()。它的输入是灰度图像，输出时我们的 histogram 均衡图像。

```
img = cv2.imread('wiki.jpg',0)
equ = cv2.equalizeHist(img)
```

histogram 均衡在图片的 histogram 被限制在一个范围内时很有用，它在 histogram 覆盖大范围时并不好用。

对比度受限自适应直方图均衡

我们第一个 histogram 均衡考虑的是图片全部对比度，在很多情况下，这不是好主意，

背景对比度在均衡后确实提高了。但是对比两张图里的雕像，我们由于过度亮而导致失去了大部分面部信息。这是由于图像的 histogram 并不是像之前的图片那样限制在特定范围内。

要解决这个问题，要用适应性 histogram 均衡。在这里，图像被分成小块，这些小块叫做瓷砖（瓷砖的大小在 OpenCV 里默认是 8x8）。然后这些小块还和平常一样做均衡，所以在小块里，histogram 是在小范围内的，如果有噪点，会被放大。要避免这个，要应用对比度限制。如果任何 histogram 高于特定的对比度限制（OpenCV 里默认是 40），那些像素会被修剪掉并被无变化的放到其他然后再做 histogram 均衡。均衡后，要移除瓷砖边界的人工因素，要应用双线性插值。

下面的代码显示了 OpenCV 里的 CLAHE：

```
# create a CLAHE object (Arguments are optional).
clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
cl1 = clahe.apply(img)
cv2.imwrite('clahe_2.jpg',cl1)
```

3.5 OpenCV-Python 教程:25.Histograms-3:2d Histograms

https://www.jianshu.com/p/2f3fcabad43a

在第一节里，我们计算和绘制了一维的 histogram，它被叫做一维 histogram 是因为我们只拿了一个属性出来，像素的灰度强度值。而如果是二维 histogram，你就要考虑两个属性了。一般来说对于彩色 histogram 两个属性是色调和饱和度的值。

OpenCV 里的 2D histogram

用 cv2.calcHist() 很简单，对于彩色 histogram 我们需要把图像从 BGR 转换成 HSV。（记住，对于 1 维 histogram，我们从 BGR 转成灰度），

Numpy 里的 2D histogram

Numpy 也提供了函数 np.histogram2d()。（记住，对于 1 维的是 np.histogram()）

3.6 OpenCV-Python 教程:26.Histogram 4 Histogram 向后投影

https://www.jianshu.com/p/31cd06b0bd6f

用来在图片分割或者在图像里找感兴趣的目标，简单说来，它创建了一个和输入图像一样大小的图像，但是是单通道的。输出图像会有我们感兴趣的目标，但是它比其他部分更白。

3.7 OpenCV-Python 教程:27. 图像转换

https://www.jianshu.com/p/58c39dce2a7a

傅里叶变换

图像处理（5）–图像的傅里叶变换：https://blog.csdn.net/qq_33208851/article/details/94834614

信号在频率域的表现

在频域中，频率越大说明原始信号变化速度越快；频率越小说明原始信号越平缓。当频率为 0 时，表示直流信号，没有变化。因此，频率的大小反应了信号的变化快慢。高频分量解释信号的突变部分，而低频分量决定信号的整体形象。

在图像处理中，频域反应了图像在空域灰度变化剧烈程度，也就是图像灰度的变化速度，也就是图像的梯度大小。对图像而言，图像的边缘部分是突变部分，变化较快，因此反应在频域上是高频分量；图像的噪声大部分情况下是高频部分；图像平缓变化部分则为低频分量。也就是说，傅立叶变换提供另外一个角度来观察图像，可以将图像从灰度分布转化到频率分布上来观察图像的特征。书面一点说就是，傅里叶变换提供了一条从空域到频率自由转换的途径。对图像处理而言，以下概念非常的重要：

图像高频分量：图像突变部分；在某些情况下指图像边缘信息，某些情况下指噪声，更多是两者的混合；

低频分量：图像变化平缓的部分，也就是图像轮廓信息

高通滤波器：让图像使低频分量抑制，高频分量通过

低通滤波器：与高通相反，让图像使高频分量抑制，低频分量通过

带通滤波器：使图像在某一部分的频率信息通过，其他过低或过高都抑制

还有个带阻滤波器，是带通的反。

3.8 OpenCV-Python 教程:28. 模板匹配

https://www.jianshu.com/p/53ef74b02f6a

模板匹配是在一个大图里搜索和找模板图像位置的方法。OpenCV 有个函数 cv2.matchTemplate() 来做这个。它吧模板图像在输入图像上滑动，对比模板和在模板图像下的输入图像块。它返回了一个灰度图像，每个像素表示那个像素周围和模板匹配的情况。

如果输入图像大小是 WxH 而模板图像大小是 wxh，输出图像的大小是（W-w+1，H-h+1）。当你得到了结果，你可以用 cv2.minMaxLoc() 函数来找最大最小值。把它作为矩形左上角，w，h 作为矩形的宽和高。矩形是你的模板区域。

注意:

如果你用 cv2.TM_SQDIFF 作为比较方法，最小值是最匹配。

尝试所有的比较方法

methods = ['cv2.TM_CCOEFF' , 'cv2.TM_CCOEFF_NORMED' , 'cv2.TM_CCORR' ,' cv2.TM_CCORR_NORMED' , 'cv2.TM_SQDIFF' , 'cv2.TM_SQDIFF_NORMED']

模板匹配多个目标

在前面我们搜索了 messi 的脸，目标只在图像里出现了一次，假设你要搜的东西在图像里出现多次，cv2.minMaxLoc() 不会给你所有的位置。在这种情况下，我们会使用阈值，在这个例子里，我们使用超级玛丽的截图来找金币。

```
img_rgb = cv2.imread('mario.png')
img_gray = cv2.cvtColor(img_rgb, cv2.COLOR_BGR2GRAY)
template = cv2.imread('mario_coin.png',0)
w, h = template.shape[::-1]

res = cv2.matchTemplate(img_gray,template,cv2.TM_CCOEFF_NORMED)
threshold = 0.8
loc = np.where( res >= threshold)
for pt in zip(*loc[::-1]):
    cv2.rectangle(img_rgb, pt, (pt[0] + w, pt[1] + h), (0,0,255), 2)

cv2.imwrite('res.png',img_rgb)
```

3.9 OpenCV-Python 教程:29. 霍夫线变换

https://www.jianshu.com/p/722b5ac8773d

霍夫变换（主要说明检测直线及圆的原理）:https://blog.csdn.net/weixin_40196271/article/details/83346442

霍夫变换是图像处理中从图像中识别几何形状的基本方法之一，应用很广泛，也有很多改进算法。主要用来从图像中分离出具有某种相同特征的几何形状（如，直线，圆等）。最基本的霍夫变换是从黑白图像中检测直线 (线段)。

函数 cv2.HoughLines()

CHAPTER 4

opencv 小白 04 学习笔记 04

4.1 OpenCV-Python 教程:30. 霍夫圆变换

https://www.jianshu.com/p/a795171f8092

函数是 cv2.HoughCircles()

4.2 OpenCV-Python 教程:31. 分水岭算法对图像进行分割

https://www.jianshu.com/p/de81d6029235

4.3 OpenCV-Python 教程:32. 使用 GrabCut 算法分割前景

https://www.jianshu.com/p/117f66320589

开始用户画一个矩形方块把前景图圈起来，前景区域应该完全在矩形内，然后算法反复进行分割以达到最好效果。但是有些情况下，分割的不是很好，比如把前景给标称背景了等。在这种情况下用户需要再润色，就在图像上有缺陷的点给几笔。这几笔的意思是说“嘿，这个区域应该是前景，你把它标成背景了，下次迭代改过来”或者是反过来。那么在下次迭代，结果会更好。

4.4 OpenCV-Python 教程:33. 特征检测和描述

https://www.jianshu.com/p/f222200a5769

4.5 OpenCV-Python 教程:34.Harris 角点检测

https://www.jianshu.com/p/9c7cf8ea183f

cv2.cornerHarris(gray,2,3,0.04)

有时候，你可能需要更准确的找到角，OpenCV 用函数 cv2.cornerSubPix() 通过亚像素精度精炼角点检测。下面是例子，我们需要先找到 harris 角，然后我们把这些角的质心传进去（可能有一堆点在角上，我们找他们的质心）来精炼他们。

```
img = cv2.imread(filename)
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

# find Harris corners
gray = np.float32(gray)
dst = cv2.cornerHarris(gray,2,3,0.04)
dst = cv2.dilate(dst,None)
ret, dst = cv2.threshold(dst,0.01*dst.max(),255,0)
dst = np.uint8(dst)

# find centroids
ret, labels, stats, centroids = cv2.connectedComponentsWithStats(dst)

# define the criteria to stop and refine the corners
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 100, 0.001)
corners = cv2.cornerSubPix(gray,np.float32(centroids),(5,5),(-1,-1),criteria)
```

4.6 OpenCV-Python 教程:35.Shi-Tomasi 角点检测和特征跟踪

https://www.jianshu.com/p/163ff90e35a9

OpenCV 有个函数 cv2.goodFeaturesToTrack()。它会用 Shi-Tomasi 方法（或者 Harris 角点检测，你可以指定）找到 N 个最强的角。输入图像仍然应该是灰度图。然后你指定你想找到的角的数量，接着指定质量级别，值介于 0 和 1 之间，指明了角的最小质量。之后我们提供角之间的最小欧几里得距离。

通过所有这些信息，函数可以在图像里找角。所有低于质量级别的角被拒绝。然后它会根据质量降序对剩下的角排序。然后函数取第一个最强的角，把周围的最小距离内的所有角都扔掉，然后返回 N 个最强的角。

4.7 OpenCV-Python 教程:36.SIFT（尺度不变特征变换）

https://www.jianshu.com/p/c0379c931e74

在 SIFT 算法里主要有四步：

```
1.尺度空间极值检测
2.关键点本地化
3.方向分配
4.关键点描述
5.关键点匹配
```

OpenCV 里的 SIFT

sift.detect() 函数找到图像的关键点，你可以传一个掩图给它，如果你只想在图像的一个部分内搜索的话。每个关键点是一个特殊的结构，这些结构有很多属性，比如 (x,y) 坐标，有意义的邻居的大小，指定它们方向的角度，指定关键点力量的响应等。

OpenCV 也提供了 cv2.drawKeyPoints() 函数来在关键点的位置画上小圆圈，如果你传入一个标志位，cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS，它会画一个和关键点大小一样的圆，还会显示出它的方向。

计算描述，OpenCV 提供了两个方法：

1. 由于你已经找到关键点了，你可以调用 sift.computer() 来计算我们找到的关键点的描述，比如：kp, des = sift.compute(gray, kp)

2. 如果你没找关键点，直接用 sift.detectAndCompute() 一次直接找到关键点和描述。

```
sift=cv2.SIFT()
kp,des=sift.detectAndCompute(gray,None)
```

这里 kp 是关键点的列表，des 是形状数组

4.8 OpenCV-Python 教程:37.SURF(加速稳健特征)

https://www.jianshu.com/p/6aee43c6f2ac

```
# Again compute keypoints and check its number.
>>> kp, des = surf.detectAndCompute(img,None)
>>> print len(kp)
47
```

这比 50 少了，我们画出它来

```
>>>img2=cv2.drawKeypoints(img,kp,None,(255,0,0),4)
>>>plt.imshow(img2),plt.show()
```

4.9 OpenCV-Python 教程:38.FAST 角点检测算法

https://www.jianshu.com/p/d778e3be32ff

我们看到了一些特征检测算法，他们很多都不错，但是从实时应用的角度看，他们都不够快，一个最好的例子是 SLAM（同步定位与地图创建）移动机器人没有足够的计算能力。

作为解决方案，**FAST(加速切片测试特征）算法**被提出。

4.10 OpenCV-Python 教程:39.BRIEF

https://www.jianshu.com/p/a22b82f1df5f

BRIEF 在这个时候出现了，它提供了直接找到二进制字符串而不找描述子的简便办法。它取被平滑过的图像块，选择 nd(x,y) 集合位置对，然后在这些位置对上做像素强度对比，比如，设第一个位置对为 p 和 q，如果 I(p) < I(q)，那么它的结果是 1, 否则是 0，这用在所有 nd 个位置对，得到 nd 维的位串。

这里 nd 可以是 128,256 或者 512.OpenCV 支持所有这些，但是默认是 256（OpenCV 用字节表示，所以就是 16，32 和 64 字节)。当你得到这个，你可以使用 Hamming 距离来匹配这些描述子

一个重要的点是 **BRIEF 是一个特征描述子，它不提供任何方法来找特征**，所以你还得使用别的特征描述子比如 SIFT，SURF 等，论文推荐使用 CenSurE，是个快速检测器，BRIEF 甚至和 CenSurE 比 SURF 还好点。

简单说，**BRIEF 是一个快速的特征描述子计算和匹配方法。它提供了高识别率**，除非是大的平面旋转

CHAPTER 5

opencv 小白 05 学习笔记 05

5.1 OpenCV-Python 教程:40.ORB

https://www.jianshu.com/p/49a84ddef11d

ORB 最重要的事情是它是 OpenCV 实验室出来的，它在计算成本，匹配性能上是 SIFT 和 SURF 很好的替代品，还有最主要的，专利，对，**SIFT 和 SURF 都是有专利的，你得付费试用，但是 ORB 不是**。

ORB 基本上是一个 FAST 关键点检测和 BRIEF 描述子的融合，同时做了很多修改提高了性能。**首先它使用 FAST 来找关键点，然后用 Harris 角点测量来找到头 N 个点。还使用金字塔来产生多层级特征**，但是问题是 FAST 不计算方向，所以旋转不变呢？作者做了如下修改。

5.2 OpenCV-Python 教程:41. 特征匹配

https://www.jianshu.com/p/ed57ee1056ab

Brute-Force 匹配器很简单，它取第一个集合里一个特征的描述子并用第二个集合里所有其他的特征和他通过一些距离计算进行匹配。最近的返回。

用 cv2.BFMatcher() 创建 BF 匹配器对象. 它取两个可选参数，第一个是 normType。它指定要使用的距离量度。默认是 cv2.NORM_L2。对于 SIFT,SURF 很好。（还有 cv2.NORM_L1）。对于二进制字符串的描述子，比如 ORB，BRIEF，BRISK 等，应该用 cv2.NORM_HAMMING。使用 Hamming 距离度量，如果 ORB 使用 VTA_K == 3 或者 4，应该用 cv2.NORM_HAMMING2

第二个参数是布尔变量，crossCheck 模式是 false，如果它是 true，匹配器返回那些和 (i, j) 匹配的，这样集合 A 里的第 i 个描述子和集合 B 里的第 j 个描述子最匹配。两个集合里的两个特征应该互相匹配，它提供

了连续的结果，

匹配方法：BFMatcher.match() 和 BFMatcher.knnMatch()。第一个返回最匹配的，第二个方法返回 k 个最匹配的，k 由用户指定。当我们需要多个的时候很有用。

匹配结果绘图：想我们用 cv2.drawKeypoints() 来画关键点一样，cv2.drawMatches() 帮我们画匹配的结果，它把两个图像水平堆叠并且从第一个图像画线到第二个图像来显示匹配。还有一个 cv2.drawMatchesKnn 来画 k 个最匹配的。如果 k=2，它会给每个关键点画两根匹配线。所以我们得传一个掩图，如果我们想选择性的画的话。

5.3 OpenCV-Python 教程:42. 特征匹配 +Homography 找目标

https://www.jianshu.com/p/d835f1a4717c

使用一个 calib3d 模块里的函数，cv2.findHomography(). 如果我们传了两个图像里的点集合，它会找到那个目标的透视转换。然后我们可以使用 cv2.perspectiveTransform() 来找目标，它需要至少 4 个正确的点来找变换。

我们看过可能会有一些匹配是的错误而影响结果。哟啊解决这个问题，算法使用了 RANSAC 或者 LEAST_MEDIAN（由标志决定）。提供正确估计的好的匹配被叫做 inliers，而其他的叫做 outliers。cv2.findHomography() 返回一个掩图来指定 inlier 和 outlier。

5.4 OpenCV-Python 教程:43.meanshift 和 camshift

https://www.jianshu.com/p/4de0facae74a

5.5 OpenCV-Python 教程:44. 光流

https://www.jianshu.com/p/87934b0cdd65

光流是**物体或者摄像头的运动导致的两个连续帧之间的图像对象的视觉运动的模式**。它是一个向量场，**每个向量是一个位移矢量**，显示了从第一帧到第二帧的点的移动。

光流在很多领域有应用：

- 从移动构建
- 视频压缩
- 视频稳定

光流在很多假设下有效：

物体像素强度在连续帧之间不变化
邻居像素有相似运动

OpenCV 通过函数 cv2.calcOpticalFlowPyrLK() 提供了所有这些。这里，我们创建一个简单的应用来跟踪视频里的一些点。我们用 cv2.goodFeaturesToTrack() 来决定点。先取第一帧，检测一些 Shi-Tomasi 角点，然后我们用 Lucas-Kanade 光流法迭代跟踪那些点。对于函数 cv2.calcOpticalFlowPyrLK() 我们传前一帧，前面的那些点和下一帧。它会返回下一帧的点和一些状态值，如果下一次的点被找到了这些值就为 1，如果没找到就是 0. 我们在下一步把这些点再作为上一次的点传进去继续迭代。

OpenCV 里的密集光流

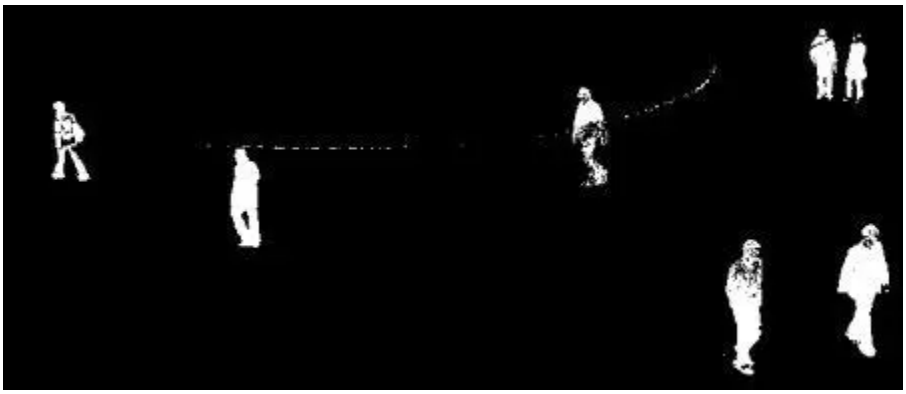
Lucas-Kanade 方法计算稀疏特征集的光流（在我们的例子里，角点检测使用 Shi-Tomasi 算法）。OpenCV 提供了另一个算法来找密集光流。它计算帧里的所有点的光流。它基于 Gunner Farneback 的算法。

5.6 OpenCV-Python 教程:45. 背景去除

https://www.jianshu.com/p/a8a9bc22ebca

OpenCV 实现了三个这样的算法，很易用，我们一个个看一下。

BackgroundSubtractorMOG



BackgroundSubtractorMOG2

BackgroundSubtractorGMG

5.7 OpenCV-Python 教程:46. 摄像头标定

https://www.jianshu.com/p/df78749b4318

5.8 OpenCV-Python 教程:47. 姿态估计

https://www.jianshu.com/p/fb4e574f5574

5.9 OpenCV-Python 教程:48. 核面几何

https://www.jianshu.com/p/0ddde4b7730f

5.10 OpenCV-Python 教程:49. 立体图像的深度图

https://www.jianshu.com/p/4a31a3d883f1

CHAPTER 6

opencv 小白 06 学习笔记 06

6.1 OpenCV-Python 教程:50. 理解 k-近邻

https://www.jianshu.com/p/404e329e4e80

OpenCV 里的 kNN

```
newcomer = np.random.randint(0,100,(1,2)).astype(np.float32)
plt.scatter(newcomer[:,0],newcomer[:,1],80,'g','o')

knn = cv2.KNearest()
knn.train(trainData,responses)
ret, results, neighbours ,dist = knn.find_nearest(newcomer, 3)
```

6.2 OpenCV-Python 教程:51. 使用 kNN 做手写数据 OCR

https://www.jianshu.com/p/af02293c5a7e

```
knn = cv2.KNearest()
knn.train(train,train_labels)
ret,result,neighbours,dist = knn.find_nearest(test,k=5)
```

6.3 OpenCV-Python 教程:52. 理解 SVM

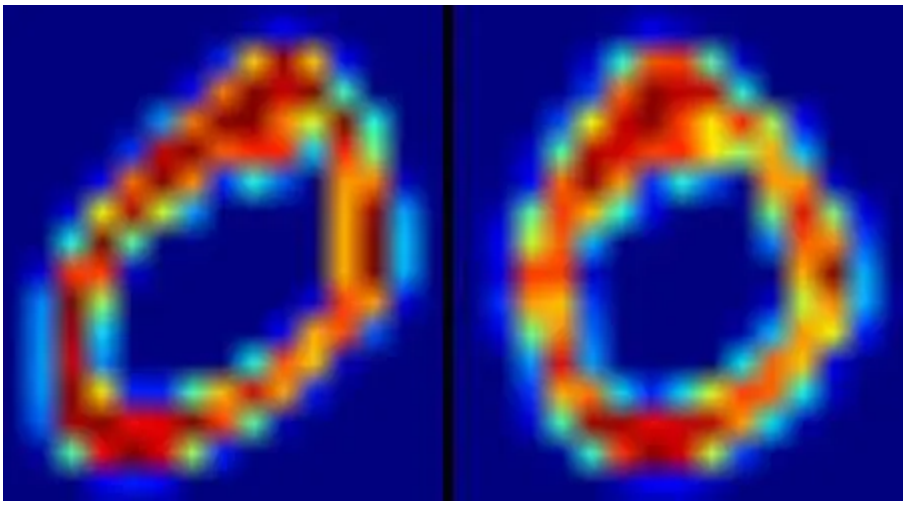
https://www.jianshu.com/p/03cfa441ca64

6.4 OpenCV-Python 教程:53. 使用 SVM 进行手写数据的 OCR

https://www.jianshu.com/p/0fd323eb0a99

使用图像的二阶矩模型来抗色偏。所以我们首先定义一个函数 deskew() 取一个数字图像并对他抗色偏。下面是 deskew() 函数:

```
def deskew(img):
    m = cv2.moments(img)
    if abs(m['mu02']) < 1e-2:
        return img.copy()
    skew = m['mu11']/m['mu02']
    M = np.float32([[1, skew, -0.5*SZ*skew], [0, 1, 0]])
    img = cv2.warpAffine(img,M,(SZ, SZ),flags=affine_flags)
    return img
```



```
svm = cv2.SVM()
svm.train(trainData,responses, params=svm_params)
svm.save('svm_data.dat')

######    Now testing      ########################

deskewed = [map(deskew,row) for row in test_cells]
hogdata = [map(hog,row) for row in deskewed]
```

(下页继续)

(续上页)

```
testData = np.float32(hogdata).reshape(-1,bin_n*4)
result = svm.predict_all(testData)
```

6.5 OpenCV-Python 教程:54.K-Means 集群

https://www.jianshu.com/p/53a1ae1651c0

6.6 OpenCV-Python 教程:55.OpenCV 里的 K-Means 聚类

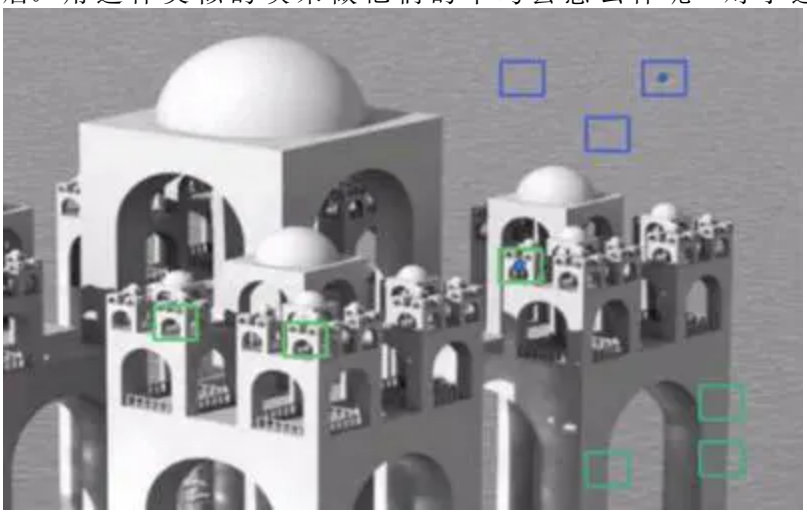
https://www.jianshu.com/p/b24086aab3fc

6.7 OpenCV-Python 教程:56. 图像去噪

https://www.jianshu.com/p/26e29adf7429

保持一个静止的摄像机对准一个位置多呆几秒，这会给你很多帧，或者是对一个场景的很多图像。然后写一些代码来找到视频里所有帧的平均值。比较最终的结果和第一帧。你可以看到噪点被去掉了。不幸的是这个简单的方法对于摄像机和场景的运动来说就不健壮了。而且经常你也只有一个噪音图像可用。

所以思路很简单，我们需要一套类似的图像来平均去掉噪点，假设图像上一个小窗口（比如 5x5 的窗口）。很有可能在图像里的某处还有一个相同的块。有时候是在它附近的邻居。用这样类似的块来做他们的平均会怎么样呢? 对于这个特定的窗口，看下面的例子:



对于彩色图像，图像先要转换成 CIELAB 颜色空间然后再分成 L 去噪和 AB 部分。

OpenCV 里的图像去噪

OpenCV 提供了这个技术的四个变形:

1.cv2.fastNlMeansDenoising() - 对于一个灰度图像的
2.cv2.fastNlMeansDenoisingColored() - 对于彩色图像的
3.cv2.fastNlMeansDenoisingMulti() - 对于短时间内拍摄的一序列图像的（灰度图像）
4.cv2.fastNlMeansDenoisingColoredMulti() - 和上面一眼个，不过是彩色图像。

通用参数如下：

- h: 决定过滤器强度的参数。更高的 h 值能够更好去噪，但是会去掉更多图像细节（10 就 ok）
- hForColorComponents: 和 h 一样，不过只是针对彩色图像的（一般和 h 一样）
- templateWindowSize: 应该是奇数（推荐 7）
- searchWindowSize: 应该是奇数（推荐 21）

6.8 OpenCV-Python 教程:57. 图像修复

https://www.jianshu.com/p/af5beb643d38

一些老照片已经有黑点啊，划痕啊等。你有想过修复它们么？我们不能简单的在绘图工具里把他们擦除了就完了。因为这样只是把黑色的东西变成白色的而已，实际上没用。在这种情况下，会用到一种技术叫图像修复。基本的思想很简单：用周围的像素替换坏掉的像素，这样看上去就和周围一样了。

OpenCV 提供了两个，可以用同一个函数来访问: cv2.inpaint()

编码

我们需要创建和输入图像相同大小的掩图，需要修复的区域对应的像素要非 0. 剩下的就简单了。我的图像被一些黑色划痕给破坏了 (实际上是我自己加的)。我用绘图工具对应的标记出来。

```
import numpy as np
import cv2

img = cv2.imread('messi_2.jpg')
mask = cv2.imread('mask2.png',0)
```

(下页继续)

(续上页)

```
dst = cv2.inpaint(img,mask,3,cv2.INPAINT_TELEA)

cv2.imshow('dst',dst)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

看下面的结果。第一个图片是输入图像，第二个是掩图，第三个是用第一种算法的结果，最后一张是第二种算法的结果。

6.9 OpenCV-Python 教程:58. 使用 Haar Cascades 面部识别

https://www.jianshu.com/p/e5f18ef0e5a8

OpenCV 里的 Haar-cascade 检测

OpenCV 提供了检测器和训练器。如果你想训练你自己的分类器来识别诸如汽车啊，飞机什么的，你可以使用 OpenCV 来创建一个。详细内容见：Cascade Classifier Training

这里我们只看检测器，OpenCV 已经包含了很多训练过的分类器，面部的，眼睛的，笑容的等。那些 XML 文件存在 opencv/data/haarcascades/目录。让我们创建一张脸和眼的检测器吧。

首先我们需要加载必须的 XML 分类器，然后用灰度模式加载我们的输入图像（或者视频）。

```
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')
```

(下页继续)

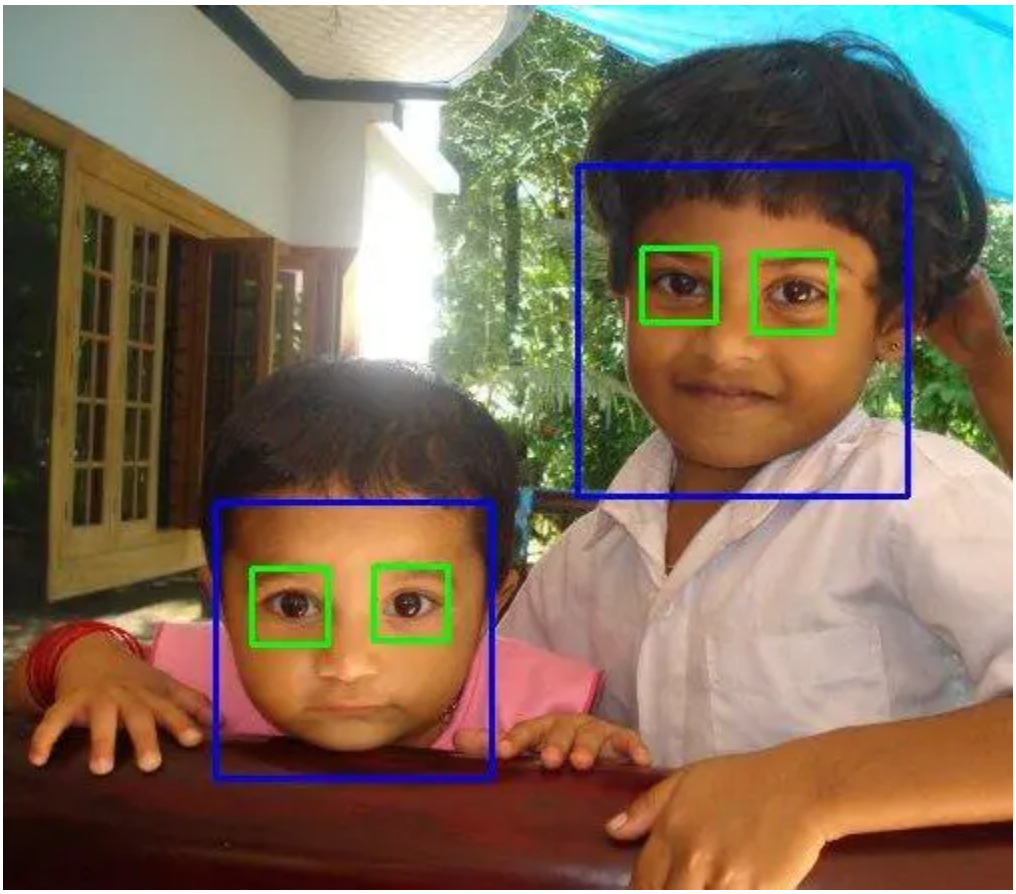
(续上页)

```
img = cv2.imread('sachin.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

现在我们在图像里找到脸。如果找到脸，它会返回检测到的脸的位置 (x, y, w,h)。当我们得到这些位置，我们可以为脸创建一个 ROI 然后在这个 ROI 上应用眼睛检测（因为眼睛总是在脸上的!）

```
faces = face_cascade.detectMultiScale(gray, 1.3, 5)
for (x,y,w,h) in faces:
    img = cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
    roi_gray = gray[y:y+h, x:x+w]
    roi_color = img[y:y+h, x:x+w]
    eyes = eye_cascade.detectMultiScale(roi_gray)
    for (ex,ey,ew,eh) in eyes:
        cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)

cv2.imshow('img',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



6.10 OpenCV-Python 教程:59.OpenCV-Python 是如何工作的

https://www.jianshu.com/p/f17ee7a100ef

python 调用 c 简介

6.11 OpenCV-Python 教程:60.scikit-learn

https://www.jianshu.com/p/aedaec60f67b

scikit-learn 简介，scikit-learn 每个数据挖掘专业必备入门工具，没啥可说的，深度学习火之前大家都是靠 scikit-learn 过日子的

6.12 OpenCV-Python 教程:61. 一元线性回归

https://www.jianshu.com/p/1710ded8ff84

使用 python 的 matlibplot 画图，体现一元线性回归的相关性关系。

CHAPTER 7

opencv 小白 07PracticalExercise 学习笔记 01

资料:OpenCV-Practical-Exercise:https://github.com/luohenyueji/OpenCV-Practical-Exercise

7.1 学习目的

1,OpenCV 用法

2, 各种机器学习场景涉及的 opencv 方法，问题解决思路等

3, 各机器学习算法使用场景和特点

7.2 1 基于深度学习识别人脸性别和年龄

核心代码

```
faceNet = cv.dnn.readNet(faceModel, faceProto)
frameFace, bboxes = getFaceBox(faceNet, frame)
    blob = cv.dnn.blobFromImage(frameOpencvDnn, 1.0, (300, 300), [104, 117, 123], True,␣
↪False)
    net.setInput(blob)
    detections = net.forward()

    gender = genderList[genderPreds[0].argmax()]
```

7.3 2 人脸识别算法对比

核心代码

```
faceCascade = cv2.CascadeClassifier('./model/haarcascade_frontalface_default.xml')
outOpencvHaar, bboxes = detectFaceOpenCVHaar(faceCascade, frame)
    faces = faceCascade.detectMultiScale(frameGray)
```

7.4 3 透明斗篷

核心代码

```
img = np.flip(img,axis=1)# 翻转

mask2 = cv2.inRange(hsv,lower_red,upper_red)# 0,1 点阵

mask1 = mask1+mask2

# Refining the mask corresponding to the detected red color
mask1 = cv2.morphologyEx(mask1, cv2.MORPH_OPEN, np.ones((3,3),np.uint8),iterations=2)
mask1 = cv2.dilate(mask1,np.ones((3,3),np.uint8),iterations = 1)# 去除噪音
mask2 = cv2.bitwise_not(mask1)# 取反

# Generating the final output
res1 = cv2.bitwise_and(background,background,mask=mask1)# 从背景中获取布的部分
res2 = cv2.bitwise_and(img,img,mask=mask2)# 从前景中获取布之外部分
final_output = cv2.addWeighted(res1,1,res2,1,0)# 合并，布以外 = 前景，布内部=背景，效果就
是布变成透视的（隐身效果的布）
```

7.5 4OpenCV 中的颜色空间

7.6 5 基于深度学习的文本检测 (略)

7.7 6 基于特征点匹配的视频稳像 (略)

7.8 7 使用 YOLOv3 和 OpenCV 进行基于深度学习的目标检测 Model

YOLO 目标检测器首先，它将图像划分为 13×13 的单元格。这 169 个单元的大小取决于输入的大小。对于我们在实验中使用的 416×416 输入尺寸，单元尺寸为 32×32。然后每个单元格作为一个边界框进行一次检测。对于每个边界框，网络还预测边界框实际包围对象的置信度，以分类的概率。大多数这些边界框都被消除了，因为它们的置信度很低，或者因为它们与另一个具有非常高置信度得分的边界框包围相同的对象。该技术称为非极大值抑制。

YOLOv3 作者使 YOLOv3 比以前的作品 YOLOv2 更快，更准确。YOLOv3 可以更好地进行多个尺度检测。他们还通过增加网络来改进网络。

```
modelConfiguration = "yolov3.cfg";
modelWeights = "yolov3.weights";

net = cv.dnn.readNetFromDarknet(modelConfiguration, modelWeights)
net.setPreferableBackend(cv.dnn.DNN_BACKEND_OPENCV)
net.setPreferableTarget(cv.dnn.DNN_TARGET_CPU)
```

第一个设置，假如设置 DEFAULT，默认设置的话，必须设置一个环境变量，并且变量的路径要是磁盘上一个文件夹，文件夹要存在，否则会警告或者报错。假如设置成 OPENCV，会在用户名一个临时文件夹生成一些 OPENCL 的文件。建议设置为 OPENCV，不用去配置环境变量

第二个设置，假如设置为 CPU 的话，速度较慢，通用性较好。设置为 OPENCL 的话，只能运行在 inter 的 GPU 上。假如电脑上有 NVIDIA 的话，会一直卡住，目前还没找到设置 OPENCV 运行哪块 GPU 的方法，没有在 NVIDIA 上的电脑上运行过。所以，为了确保 GPU 加速，不要在有 NVIDIA 电脑上运行

```
# Create a 4D blob from a frame.
blob = cv.dnn.blobFromImage(frame, 1/255, (inpWidth, inpHeight), [0,0,0], 1, crop=False)

# Sets the input to the network
net.setInput(blob)

# Runs the forward pass to get output of the output layers
outs = net.forward(getOutputsNames(net))
    # Get the names of all the layers in the network
```

(下页继续)

(续上页)

```
    layersNames = net.getLayerNames()
    # Get the names of the output layers, i.e. the layers with unconnected outputs
    return [layersNames[i[0] - 1] for i in net.getUnconnectedOutLayers()]

# Remove the bounding boxes with low confidence
postprocess(frame, outs)
    for out in outs:
        for detection in out:
            scores = detection[5:]
            classId = np.argmax(scores)
            confidence = scores[classId]
            if confidence > confThreshold:
                center_x ,center_y=xx
                width ,height , left,top = xx
                classIds.append(classId)
                confidences.append(float(confidence))
                boxes.append([left, top, width, height])

    # Perform non maximum suppression to eliminate redundant overlapping boxes with
    # lower confidences.
    indices = cv.dnn.NMSBoxes(boxes, confidences, confThreshold, nmsThreshold)
    for i in indices:
        drawPred(classIds[i], confidences[i], left, top, left + width, top + height)

# Put efficiency information. The function getPerfProfile returns the overall time for␣
↪inference(t) and the timings for each of the layers(in layersTimes)
t, _ = net.getPerfProfile()
label = 'Inference time: %.2f ms' % (t * 1000.0 / cv.getTickFrequency())
cv.putText(frame, label, (0, 15), cv.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255))
```

7.9 8 深度学习目标检测网络 YOLOv3 的训练 (略)

7.10 9 使用 OpenCV 寻找平面图形的质心

核心代码

```
gray_image = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
ret,thresh = cv2.threshold(gray_image,127,255,0)
```

(下页继续)

(续上页)

```
contours, hierarchy = cv2.findContours(thresh,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
for c in contours:
        # calculate moments for each contour
        M = cv2.moments(c)
        cX = int(M["m10"] / M["m00"])
        cY = int(M["m01"] / M["m00"])

        cv2.circle(img, (cX, cY), 5, (255, 255, 255), -1)
```

CHAPTER 8

opencv 小白 08PracticalExercise 学习笔记 02

8.1 10 使用 Hu 矩进行形状匹配

Hu 矩（或者更确切地说是 Hu 矩不变量）是使用对图像变换不变的中心矩计算的一组 7 个变量。事实证明，前 6 个矩不变量对于平移，缩放，旋转和映射都是不变的。而第 7 个矩会因为图像映射而改变。

OpenCV 中，我们 HuMoments() 用来计算输入图像中的 Hu 矩。

```
_,im = cv2.threshold(im, 128, 255, cv2.THRESH_BINARY)
moment = cv2.moments(im)
huMoments = cv2.HuMoments(moment)
```

基于 matchShapes 函数计算两个图形之间的距离


```
m2 = cv2.matchShapes(im1,im2,cv2.CONTOURS_MATCH_I2,0)
```

您可以通过第三个参数（CONTOURS_MATCH_I1，CONTOURS_MATCH_I2 或 CONTOURS_MATCH_I3）使用三种 b 不同的距离。如果上述距离很小，则两个图像（im1 和 im2）相似。您可以使用任何距离测量。它们通常产生类似的结果。

8.2 11 基于 OpenCV 的二维码扫描器

核心代码

```
qrDecoder = cv2.QRCodeDetector()
data,bbox,rectifiedImage = qrDecoder.detectAndDecode(inputImage)
```

8.3 12 使用深度学习和 OpenCV 进行手部关键点检测

核心代码

```
inpBlob = cv2.dnn.blobFromImage(frame, 1.0 / 255, (inWidth, inHeight), (0, 0, 0),␣
↪swapRB=False, crop=False)

net.setInput(inpBlob)

output = net.forward()
print("time taken by network : {:.3f}".format(time.time() - t))

points = []

for i in range(nPoints):
    probMap = output[0, i, :, :]
    probMap = cv2.resize(probMap, (frameWidth, frameHeight))

    minVal, prob, minLoc, point = cv2.minMaxLoc(probMap)

    if prob > threshold :
        cv2.circle(frameCopy, (int(point[0]), int(point[1])), 8, (0, 255, 255),␣
↪thickness=-1, lineType=cv2.FILLED)
        points.append((int(point[0]), int(point[1])))
    else :
        points.append(None)
```

8.4 13OpenCV 中使用 Mask R-CNN 进行对象检测和实例分割

核心代码

```
# For each frame, extract the bounding box and mask for each detected object
def postprocess(boxes, masks):
    numClasses = masks.shape[1]
    numDetections = boxes.shape[2]
```

(下页继续)

(续上页)

```
    frameH = frame.shape[0]
    frameW = frame.shape[1]

    for i in range(numDetections):
        box = boxes[0, 0, i]
        mask = masks[i]
        score = box[2]
        if score > confThreshold:
            classId = int(box[1])

            # Extract the bounding box
            left top right bottom = xxx

            classMask = mask[classId]
            drawBox(frame, classId, score, left, top, right, bottom, classMask)
```

8.5 14 使用 OpenCV 实现单目标跟踪

核心代码

```
bbox = cv2.selectROI(frame, False)
ok = tracker.init(frame, bbox)

while True:
    ok, frame = video.read()
    if not ok:
        break

    ok, bbox = tracker.update(frame)

    if ok:
        # Tracking success
        p1 = (int(bbox[0]), int(bbox[1]))
        p2 = (int(bbox[0] + bbox[2]), int(bbox[1] + bbox[3]))
        cv2.rectangle(frame, p1, p2, (255,0,0), 2, 1)
```

8.6 15 基于深度学习的目标跟踪算法 GOTURN

核心代码: 同上，只是模型变了，思路未变

8.7 16 使用 OpenCV 实现多目标跟踪 Video

核心代码: 和单目标类似，Tracker 变成多个

```
# Create MultiTracker object
multiTracker = cv2.MultiTracker_create()
for bbox in bboxes:
multiTracker.add(createTrackerByName(trackerType), frame, bbox)

while cap.isOpened():
success, frame = cap.read()
if not success:
  break

# get updated location of objects in subsequent frames
success, boxes = multiTracker.update(frame)

# draw tracked objects
for i, newbox in enumerate(boxes):
  p1 = (int(newbox[0]), int(newbox[1]))
  p2 = (int(newbox[0] + newbox[2]), int(newbox[1] + newbox[3]))
  cv2.rectangle(frame, p1, p2, colors[i], 2, 1)
```

8.8 17 基于卷积神经网络的 OpenCV 图像着色 (略)

8.9 18Opencv 中的单应性矩阵 Homography(略)

Homography 的应用-全景拼接

```
# Calculate Homography
h, status = cv2.findHomography(pts_src, pts_dst)

# Warp source image to destination based on homography
im_out = cv2.warpPerspective(im_src, h, (im_dst.shape[1],im_dst.shape[0]))
```

8.10 19 使用 OpenCV 实现基于特征的图像对齐

OpenCV 的图像对齐

2.1 基于特征的图像对齐的步骤

现在我们可以总结图像对齐所涉及的步骤。

Step1 读图

我们首先在 C ++ 中和 Python 中读取参考图像（或模板图像）和我们想要与此模板对齐的图像。

Step2 寻找特征点

我们检测两个图像中的 ORB 特征。虽然我们只需要 4 个特征来计算单应性，但通常在两个图像中检测到数百个特征。我们使用 Python 和 C ++ 代码中的参数 MAX_FEATURES 来控制功能的数量。

Step3 特征点匹配

我们在两个图像中找到匹配的特征，按匹配的评分对它们进行排序，并保留一小部分原始匹配。我们使用汉明距离（hamming distance）作为两个特征描述符之间相似性的度量。请注意，我们有许多不正确的匹配。

Step4 计算 Homography

当我们在两个图像中有 4 个或更多对应点时，可以计算单应性。上一节中介绍的自动功能匹配并不总能产生 100％准确的匹配。20-30％的比赛不正确并不罕见。幸运的是，findHomography 方法利用称为随机抽样一致性算法（RANSAC）的强大估计技术，即使在存在大量不良匹配的情况下也能产生正确的结果。RANSAC 具体介绍见：

https://www.cnblogs.com/xingshansi/p/6763668.html

https://blog.csdn.net/zinnc/article/details/52319716

Step5 图像映射

一旦计算出准确的单应性，我可以应用于一个图像中的所有像素，以将其映射到另一个图像。这是使用 OpenCV 中的 warpPerspective 函数完成的。

CHAPTER 9

opencv 小白 09PracticalExercise 学习笔记 03

9.1 20 使用 OpenCV 实现基于增强相关系数最大化的图像对齐 (略)

9.2 21 使用 OpenCV 的 Eigenface

如何计算如何计算 EigenFaces

要计算 EigenFaces，我们需要使用以下步骤：

1）获取面部图像数据集：我们需要一组包含不同类型面部的面部图像。在这篇文章中，我们使用了来自 CelebA 的约 200 张图片。CelebA 数据集见：http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html

2）对齐和调整图像大小：接下来我们需要对齐和调整图像大小，以便在所有图像中眼睛的中心都是对齐的。这可以通过首先找到面部特征点来完成。在这篇文章中，我们使用了 CelebA 中提供的对齐图像。此时，数据集中的所有图像应该具有相同的大小。

3）创建数据矩阵：创建一个包含所有图像作为行向量的数据矩阵。如果数据集中的所有图像大小为 100 x 100 且有 1000 个图像，我们将拥有大小为 30k x 1000 的数据矩阵。

4）计算平均向量 [可选]：在对数据执行 PCA 之前，我们需要减去平均向量。在我们的例子中，平均向量将是通过平均数据矩阵的所有行计算的 30k×1 行向量。使用 OpenCV 的 PCA 类不需要计算这个平均向量的原因是因为如果没有提供向量，OpenCV 可以方便地计算我们的平均值。在其他线性代数包中可能不是这种情况。

5）计算主成分：通过找到协方差矩阵的特征向量来计算该数据矩阵的主成分。幸运的是，OpenCV 中的 PCA 类为我们处理了这个计算。我们只需要提供数据矩阵，然后输出一个包含 Eigenvectors 的矩阵。

6）重塑特征向量以获得 EigenFaces：如果我们的数据集包含大小为 100 x 100 x 3 的图像，那么如此获得的特征向量将具有 30k 的长度。我们可以将这些特征向量重塑为 100 x 100 x 3 图像以获得 EigenFaces。

核心代码

```
data = createDataMatrix(images)# 每个图像一行
mean, eigenVectors = cv2.PCACompute(data, mean=None, maxComponents=NUM_EIGEN_FACES)
averageFace = mean.reshape(sz)
eigenFaces = [];

for eigenVector in eigenVectors:
        eigenFace = eigenVector.reshape(sz)
        eigenFaces.append(eigenFace)

# Display result at 2x size
output = cv2.resize(averageFace, (0,0), fx=2, fy=2)
cv2.imshow("Result", output)
```

9.3 22 使用 EigenFaces 进行人脸重建 (略)

9.4 23 使用 OpenCV 获取高动态范围成像 HDR(略)

9.5 24 使用 OpenCV 进行曝光融合 (略)

9.6 25 使用 OpenCV 进行泊松克隆 (略)

9.7 26 基于 OpenCV 实现选择性搜索算法

目标检测与目标识别

目标识别算法 Target Recognition **识别图像中存在哪些对象**。它将整个图像作为输入，并输出该图像中存在的对象的类标签和类概率。例如，类标签可以是“狗”，相关的类概率可以是 97%。另一方面，目标检测算法 **Target Detection 不仅告诉您图像中存在哪些对象，还输出边界框（x，y，width，height）以表示图像内对象的位置**。

所有目标检测算法的核心是物体识别算法。假设我们训练了一个目标识别模型，该模型识别图像中的狗。该模型将判断图像中是否有狗。它不会告诉对象的位置。

为了定位物体，我们必须要选择图片的次区域（子块）然后对这些图片子块应用目标识别算法。目标的位置由目标识别算法返回的类概率较高的图像块的位置给出。

最直接的生成较小的次区域的方法为滑动窗口方法。然而，滑动窗口方法有许多限制。这些限制叫做候选区域算法克服了。**选择性搜索就是候选区域算法中最流行的一种**。

OpenCV 有自带的 SelectiveSearchSegmentation 类

9.8 27 在 OpenCV 下使用 forEach 进行并行像素访问 (略，仅 C)

OpenCV 中有隐藏的功能，有时候并不是很有名。其中一个隐藏的功能是 Mat 类的 forEach 方法，它利用机器上的所有核心在每个像素上处理任何功能。

9.9 28 基于 OpenCV 的 GUI 库 cvui(略，仅 C)

它是一个基于 OpenCV 绘图基元构建的跨平台 GUI 库，仅需使用头文件就可以搭建。除了 OpenCV 本身（您可能已经在使用）之外，它没有依赖关系。Cvui 在 C++ 下通过.h 文件实现全部功能，在 Python 下直接提供.py 文件。本文仅仅讲述 cvui 在 C++ 下的构建，python 通常用的少。

cvui 遵循一行代码就可以在屏幕上产生一个 UI 组件的规则。cvui 具有友好的 C 类 API，没有类/对象和多个组件，例如，跟踪栏，按钮，文字等等。

9.10 29 使用 OpenCV 实现红眼自动去除

CHAPTER 10

opencv 小白 10PracticalExercise 学习笔记 04

10.1 30 使用 OpenCV 实现图像孔洞填充

10.2 31 使用 OpenCV 将一个三角形仿射变换到另一个三角形

```
# Given a pair of triangles, find the affine transform.
warpMat = cv2.getAffineTransform( np.float32(tri1Cropped), np.float32(tri2Cropped) )

# Apply the Affine Transform just found to the src image
img2Cropped = cv2.warpAffine( img1Cropped, warpMat, (r2[2], r2[3]), None, flags=cv2.
↪INTER_LINEAR, borderMode=cv2.BORDER_REFLECT_101 )

# Get mask by filling triangle
mask = np.zeros((r2[3], r2[2], 3), dtype = np.float32)
cv2.fillConvexPoly(mask, np.int32(tri2Cropped), (1.0, 1.0, 1.0), 16, 0);

img2Cropped = img2Cropped * mask
```

10.3 32 使用 OpenCV 进行非真实感渲染

核心代码

```
# Edge preserving filter with two different flags.
imout = cv2.edgePreservingFilter(im, flags=cv2.RECURS_FILTER);
cv2.imwrite("edge-preserving-recursive-filter.jpg", imout);

imout = cv2.edgePreservingFilter(im, flags=cv2.NORMCONV_FILTER);
cv2.imwrite("edge-preserving-normalized-convolution-filter.jpg", imout);

imout = cv2.detailEnhance(im);
cv2.imwrite("detail-enhance.jpg", imout);

imout_gray, imout = cv2.pencilSketch(im, sigma_s=60, sigma_r=0.07, shade_factor=0.05);
cv2.imwrite("pencil-sketch.jpg", imout_gray);
cv2.imwrite("pencil-sketch-color.jpg", imout);

cv2.stylization(im,imout);
cv2.imwrite("stylization.jpg", imout);
```

10.4 33 使用 OpenCV 进行 Hough 变换 (略)

10.5 34 使用 OpenCV 进行图像修复 (略)

10.6 35 使用 Tesseract 和 OpenCV 实现文本识别

如果使用 tesseract，在实际工程 tesseract 错误率很高，识别率极差。一般需要对图像进行各种图像处理后再用 tesseract 识别，最后根据错误类型进行二次识别。tesseract 的错误还是具有一定规律的。另外 tesseract 识别中文效果并不好，你要制作专门的中文训练集通过 jTessBoxEditor.jar 去训练它，但是整个制作流程较为复杂。具体见：

https://github.com/tesseract-ocr/tesseract/wiki/TrainingTesseract-4.00

tesseract 要想有好的识别效果，就必须有大量的训练样本。但是 tesseract 对英文支持还是不错的。

```
imPath = 'image/computer-vision.jpg'
config = ('-l eng --oem 1 --psm 3')
im = cv2.imread(imPath, cv2.IMREAD_COLOR)
text = pytesseract.image_to_string(im, config=config)
print(text)
```

10.7 36 使用 OpenCV 在视频中实现简单背景估计

核心代码

```
medianFrame = np.median(frames, axis=0).astype(dtype=np.uint8)
grayMedianFrame = cv2.cvtColor(medianFrame, cv2.COLOR_BGR2GRAY)

while(ret):
    ret, frame = cap.read()
    frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    dframe = cv2.absdiff(frame, grayMedianFrame)
    th, dframe = cv2.threshold(dframe, 30, 255, cv2.THRESH_BINARY)
```

10.8 37 图像质量评价 BRISQUE

10.9 38 基于 OpenCV 的相机标定

10.10 39 在 OpenCV 中使用 ArUco 标记的增强现实

aruco 标记是放置在被成像的对象或场景上的基准标记。它是一个具有黑色背景和边界的二元正方形，其内部生成的白色图案唯一地标识了它。黑边界有助于他们更容易被发现。它们可以产生多种大小。根据对象大小和场景选择大小，以便成功检测。如果很小的标记没有被检测到，仅仅增加它们的大小就可以使它们的检测更容易。

想法是您打印这些标记并将其放置在现实世界中。您可以拍摄现实世界并独特地检测这些标记。如果您是初学者，您可能会在想这有什么用？让我们看几个用例。

在我们在帖子中分享的示例中，我们将打印的内容和标记放在相框的角上。当我们唯一地标识标记时，我们可以用任意视频或图像替换相框。当我们移动相机时，新图片具有正确的透视失真。

在机器人应用程序中，您可以将这些标记沿着配备有摄像头的仓库机器人的路径放置。当安装在机器人上的摄像头检测到一个这些标记时，它可以知道它在仓库中的精确位置，因为每个标记都有一个唯一的 ID，我们知道标记在仓库中的位置。

10.11 40 计算机视觉工具对比

2 适用于计算机视觉的 MATLAB

```
2.1 为什么要使用 MATLAB 进行计算机视觉：优点
2.2 为什么不应该将 MATLAB 用于计算机视觉：缺点
```

3 适用于计算机视觉的 OpenCV（C++）

3.1 为什么要使用 OpenCV（C++）进行计算机视觉：优点
3.2 为什么不应该将 OpenCV（C++）用于计算机视觉：缺点

4 适用于计算机视觉的 OpenCV（Python）

4.1 为什么要使用 OpenCV（Python）进行计算机视觉：优点
4.2 为什么不应该将 OpenCV（Python）用于计算机视觉：缺点

10.12 41 嵌入式计算机视觉设备选择

总而言之，Raspberry Pi，Jetson TK1 和 Jetson TX1 明显领先于当今，拥有庞大的社区和公司。ODROID-C2 是一匹黑马，可以替代 Raspberry Pi。尽管如此，这个市场还处于新生阶段，有太多的大公司仍在努力在这个市场上有所作为。

实际上就个人经历而言，以深度学习为代表的人工智能技术最近遇到大挫折，深度学习存在许多瓶颈问题，计算机视觉技术也没有大的进展。现在工业应用上以 caffe 框架居多，实际也是云端/PC 端/android 端较多，嵌入式开发看看华为海思，英伟达的设备。树莓派搞搞研究挺好的，工业应用成本过高，其他的设备不建议使用。

10.13 42 数码单反相机的技术细节 (略)

CHAPTER 11

Indices and tables

- genindex
- modindex
- search