opencv基础知识和API

1.1创建和显示窗口namedWindow()

imshow()显示窗口

destroyAllwindws()摧毁窗口

resizeWindow()改变窗口大小

waitKey()等待用户输入

**1.2视频录制**

VideoWriter :参数一为输出文件,参数二为多媒体文件格式(VideoWriter\_fourcc,参数三为帧率,参数四为分辨率.write编码并写入缓存release缓存内容写入磁盘,并释放资源 编解码器常见：\*.mpg,\*.avi,\*mp4

**1.3视频采集**

视频是由图片组成的,视频的每一帧就是一幅图片,一般是30帧,表示一秒显示30张图片.cv2.VideoCapture可以捕获摄像头,用数字来表示不同的设备,比如0, 1可以直接指定路径即可.

1.4 从文件中读取视频并播放在OpenCV中我们要获取一个视频，需要创建一个VideoCapture对象，指定你要读取的视频文件：

创建读取视频的对象

cap = cv.VideoCapture(filepath)

参数：

1. filepath: 视频文件路径

判断图像是否读取成功

isornot = cap.isOpened()

若读取成功则返回true，否则返回False

获取视频的一帧图像

ret, frame = cap.read()

参数：

* 1. ret: 若获取成功返回True，获取失败，返回False
  2. Frame: 获取到的某一帧的图像

调用cv.imshow()显示图像，在显示图像时使用cv.waitkey()设置适当的持续时间，如果太低视频会播放的非常快，如果太高就会播放的非常慢，通常情况下我们设置25ms就可以了。最后，调用cap.realease()将视频释放掉

# 2 保存视频

在OpenCV中我们保存视频使用的是VedioWriter对象，在其中指定输出文件的名称，如下所示：

1. 创建视频写入的对象

out = cv2.VideoWriter(filename,fourcc, fps, frameSize)

参数：

* filename：视频保存的位置
* fourcc：指定视频编解码器的4字节代码
* fps：帧率
* frameSize：帧大小
* 设置视频的编解码器，如下所示，

retval = cv2.VideoWriter\_fourcc( c1, c2, c3, c4 )

参数：c1,c2,c3,c4: 是视频编解码器的4字节代码，在[fourcc.org](http://www.fourcc.org/codecs.php" \t "C:/Users/dsg/Desktop/%E7%AC%AC%E5%85%AB%E7%BB%84/%E6%96%B0%E5%BB%BA%E6%96%87%E4%BB%B6%E5%A4%B9/_blank)中找到可用代码列表，与平台紧密相关，常用的有：在Windows中：DIVX（.avi）在OS中：MJPG（.mp4），DIVX（.avi），X264（.mkv）。利用cap.read()获取视频中的每一帧图像，并使用out.write()将某一帧图像写入视频中。

使用cap.release()和out.release()释放资源。

**总结**

读取视频：

读取视频：cap = cv.VideoCapture()

判断读取成功：cap.isOpened()

读取每一帧图像：ret,frame = cap.read(）

获取属性：cap.get(proid)

设置属性：cap.set(proid,value)

资源释放：cap.release()

保存视频： out = cv.VideoWrite()

视频写入：out.write()

资源释放：out.release()

1. 几何变化

## 1 图像缩放

缩放是对图像的大小进行调整，即使图像放大或缩小。

API

cv2.resize(src,dsize,fx=0,fy=0,interpolation=cv2.INTER\_LINEAR)

参数：

src : 输入图像

dsize: 绝对尺寸，直接指定调整后图像的大小

fx,fy: 相对尺寸，将dsize设置为None，然后将fx和fy设置为比例因子即可

interpolation：插值方法，

## 2 图像平移

图像平移将图像按照指定方向和距离，移动到相应的位置。

1. API

cv.warpAffine(img,M,dsize)

参数：

img: 输入图像

M： 2\*∗3移动矩阵

注意：将M设置为np.float32类型的Numpy数组。

dsize: 输出图像的大小

**注意：输出图像的大小，它应该是(宽度，高度)的形式。请记住,width=列数，height=行数。**

## 3 图像旋转

API

cv2.getRotationMatrix2D(center, angle, scale)

参数：

center：旋转中心

angle：旋转角度

scale：缩放比例

M：旋转矩阵

调用cv.warpAffine完成图像的旋转

**总结**

图像缩放：对图像进行放大或缩小

cv.resize()

图像平移：

指定平移矩阵后，调用cv.warpAffine()平移图像

图像旋转：

调用cv.getRotationMatrix2D获取旋转矩阵，然后调用cv.warpAffine()进行旋转

仿射变换：

调用cv.getAffineTransform将创建变换矩阵，最后该矩阵将传递给cv.warpAffine()进行变换

透射变换：通过函数cv.getPerspectiveTransform()找到变换矩阵，将cv.warpPerspective()进行投射变换

金字塔

图像金字塔是图像多尺度表达的一种，使用的API：

cv.pyrUp(): 向上采样

cv.pyrDown(): 向下采样

1. **形态学操作**
2. **腐蚀**

**腐蚀的作用是消除物体边界点，使目标缩小，可以消除小于结构元素的噪声点。**

**API**：

cv.erode(img,kernel,iterations)

参数：

* img: 要处理的图像
* kernel: 核结构
* iterations: 腐蚀的次数，默认是1

1. **膨胀**

**膨胀的作用是将与物体接触的所有背景点合并到物体中，使目标增大，可添补目标中的孔洞。**

**API**：

cv.dilate(img,kernel,iterations)

参数：

img: 要处理的图像

kernel: 核结构

iterations: 腐蚀的次数，默认是1

## 2.2 开闭运算

开运算和闭运算是将腐蚀和膨胀按照一定的次序进行处理。 但这两者并不是可逆的，即先开后闭并不能得到原来的图像

**1.开运算**

开运算是先腐蚀后膨胀，其**作用**是：分离物体，消除小区域。**特点**：消除噪点，去除小的干扰块，而不影响原来的图像。

**2.闭运算**

闭运算与开运算相反，是先膨胀后腐蚀，**作用**是消除/“闭合”物体里面的孔洞，**特点**：可以填充闭合区域。

**3.API**

cv.morphologyEx(img, op, kernel)

参数：

* 1. img: 要处理的图像
  2. op: 处理方式：若进行开运算，则设为cv.MORPH\_OPEN，若进行闭运算，则设为cv.MORPH\_CLOSE
  3. Kernel： 核结构

## 2.3 礼帽和黑

**1.礼帽运算：**因为开运算带来的结果是放大了裂缝或者局部低亮度的区域，因此，从原图中减去开运算后的图，得到的效果图突出了比原图轮廓周围的区域更明亮的区域，且这一操作和选择的核的大小相关。礼帽运算用来分离比邻近点亮一些的斑块。当一幅图像具有大幅的背景的时候，而微小物品比较有规律的情况下，可以使用顶帽运算进行背景提取。

2.**黑帽运算：**黑帽运算后的效果图突出了比原图轮廓周围的区域更暗的区域，且这一操作和选择的核的大小相关。黑帽运算用来分离比邻近点暗一些的斑块。

**API**

cv.morphologyEx(img, op, kernel)

参数：

img: 要处理的图像

op: 处理方式：

Kernel： 核结构

**总结**

连通性 邻接关系：4邻接，8邻接和D邻接

连通性：4连通，8连通和m连通

形态学操作

腐蚀和膨胀：

腐蚀：求局部最大值

膨胀：求局部最小值

开闭运算：

开：先腐蚀后膨胀

闭：先膨胀后腐蚀

礼帽和黑帽：

礼帽：原图像与开运算之差

黑帽：闭运算与原图像之差

1. 图像平滑

# 1 图像平滑简介

图像平滑从信号处理的角度看就是去除其中的高频信息，保留低频信息。因此我们可以对图像实施低通滤波。低通滤波可以去除图像中的噪声，对图像进行平滑。根据滤波器的不同可分为均值滤波，高斯滤波，中值滤波， 双边滤波。

## 均值滤波

均值滤波的优点是算法简单，计算速度较快，缺点是在去噪的同时去除了很多细节部分，将图像变得模糊。

API:

cv.blur(src, ksize, anchor, borderType)

参数:

* src：输入图像
* ksize：卷积核的大小
* anchor：默认值 (-1,-1) ，表示核中心
* borderType：边界类型

## 高斯滤波

高斯平滑在从图像中去除高斯噪声方面非常有效。

API：

cv2.GaussianBlur(src,ksize,sigmaX,sigmay,borderType)

参数：

* src: 输入图像
* ksize:高斯卷积核的大小，**注意** ： 卷积核的宽度和高度都应为奇数，且可以不同
* sigmaX: 水平方向的标准差
* sigmaY: 垂直方向的标准差，默认值为0，表示与sigmaX相同
* borderType:填充边界类型

## 2.3 中值滤波

中值滤波对椒盐噪声（salt-and-pepper noise）来说尤其有用，因为它不依赖于邻域内那些与典型值差别很大的值。

API：

cv.medianBlur(src, ksize )

参数：

* src：输入图像
* ksize：卷积核的大小

**总结**

图像噪声

* 1. 椒盐噪声：图像中随机出现的白点或者黑点
  2. 高斯噪声：噪声的概率密度分布是正态分布

图像平滑

均值滤波：算法简单，计算速度快，在去噪的同时去除了很多细节部分，将图像变得模糊cv.blur()

高斯滤波: 去除高斯噪声cv.GaussianBlur()

中值滤波: 去除椒盐噪声cv.medianBlur()

1. 直方图

* dims：需要统计的特征数目。在上例中，dims = 1 ，因为仅仅统计了灰度值。
* bins：每个特征空间子区段的数目，可译为 “直条” 或 “组距”，在上例中， bins = 16。
* range：要统计特征的取值范围。在上例中，range = [0, 255]。

直方图的**意义**：

* 直方图是图像中像素强度分布的图形表达方式。
* 它统计了每一个强度值所具有的像素个数。
* 不同的图像的直方图可能是相同的

## 1.2 直方图的计算和绘制

我们使用OpenCV中的方法统计直方图，并使用matplotlib将其绘制出来。

API：

cv2.calcHist(images,channels,mask,histSize,ranges[,hist[,accumulate]])

参数：

images: 原图像。当传入函数时应该用中括号 [] 括起来，例如：[img]。

channels: 如果输入图像是灰度图，它的值就是 [0]；如果是彩色图像的话，传入的参数可以是 [0]，[1]，[2] 它们分别对应着通道 B，G，R。

mask: 掩模图像。要统计整幅图像的直方图就把它设为 None。但是如果你想统计图像某一部分的直方图的话，你就需要制作一个掩模图像，并使用它。

histSize:BIN 的数目。也应该用中括号括起来，例如：[256]。

* ranges: 像素值范围，通常为 [0，256]

## 1.3 掩膜的应用

掩膜是用选定的图像、图形或物体，对要处理的图像进行遮挡，来控制图像 处理的区域。

在数字图像处理中，我们通常使用二维矩阵数组进行掩膜。掩膜是由0和1组成一个二进制图像，利用该掩膜图像要处理的图像进行掩膜，其中1值的区域被处理，0 值区域被屏蔽，不会处理。

掩膜的主要用途是：

* 提取感兴趣区域：用预先制作的感兴趣区掩模与待处理图像进行”与“操作，得到感兴趣区图像，感兴趣区内图像值保持不变，而区外图像值都为0。
* 屏蔽作用：用掩模对图像上某些区域作屏蔽，使其不参加处理或不参加处理参数的计算，或仅对屏蔽区作处理或统计。
* 结构特征提取：用相似性变量或图像匹配方法检测和提取图像中与掩模相似的结构特征。
* 特殊形状图像制作

掩膜在遥感影像处理中使用较多，当提取道路或者河流，或者房屋时，通过一个掩膜矩阵来对图像进行像素过滤，然后将我们需要的地物或者标志突出显示出来。

我们使用cv.calcHist（）来查找完整图像的直方图。 如果要查找图像某些区域的直方图，该怎么办？ 只需在要查找直方图的区域上创建一个白色的掩膜图像，否则创建黑色， 然后将其作为掩码mask传递即可。

# 2 直方图均衡化

###### “直方图均衡化”是把原始图像的灰度直方图从比较集中的某个灰度区间变成在更广泛灰度范围内的分布。直方图均衡化就是对图像进行非线性拉伸，重新分配图像像素值，使一定灰度范围内的像素数量大致相同。这种方法提高图像整体的对比度，特别是有用数据的像素值分布比较接近时，在X光图像中使用广泛，可以提高骨架结构的显示，另外在曝光过度或不足的图像中可以更好的突出细节

使用opencv进行直方图统计时，使用的是：

API：

dst = cv.equalizeHist(img)

参数：

* img: 灰度图像

返回：

* dst : 均衡化后的结果

## 2.2 自适应的直方图均衡化

API：

cv.createCLAHE(clipLimit, tileGridSize)

参数：

* clipLimit: 对比度限制，默认是40
* tileGridSize: 分块的大小，默认为8\*88∗8

**总结**

灰度直方图：

直方图是图像中像素强度分布的图形表达方式。

统计了每一个强度值所具有的像素个数。

不同的图像的直方图可能是相同的

cv.calcHist（images，channels，mask，histSize，ranges [，hist [，accumulate]]）

掩膜

创建蒙版，透过mask进行传递，可获取感兴趣区域的直方图

直方图均衡化：增强图像对比度的一种方法

cv.equalizeHist(): 输入是灰度图像，输出是直方图均衡图像

自适应的直方图均衡

将整幅图像分成很多小块，然后再对每一个小块分别进行直方图均衡化，最后进行拼接

clahe = cv.createCLAHE(clipLimit, tileGridSize)

1. 边缘检测

# Sobel检测算子

Sobel边缘检测算法比较简单，实际应用中效率比canny边缘检测效率要高，但是边缘不如Canny检测的准确，但是很多实际应用的场合，sobel边缘却是首选，Sobel算子是高斯平滑与微分操作的结合体，所以其抗噪声能力很强，用途较多。尤其是效率要求较高，而对细纹理不太关心的时候

利用OpenCV进行sobel边缘检测的API是：

Sobel\_x\_or\_y = cv2.Sobel(src, ddepth, dx, dy, dst, ksize, scale, delta, borderType)

参数：

src：传入的图像

ddepth: 图像的深度

dx和dy: 指求导的阶数，0表示这个方向上没有求导，取值为0，1

ksize: 是Sobel算子的大小，即卷积核的大小，必须为奇数1、3、5、7，默认为3

注意：如果ksize=-1，就演变成为3x3的Scharr算子。

Scale：缩放导数的比例常数，默认情况为没有伸缩系数。

borderType：图像边界的模式，默认值为cv2.BORDER\_DEFAULT。

# Canny边缘检测

## 2.2 应用

在OpenCV中要实现Canny检测使用的API:

canny = cv2.Canny(image, threshold1, threshold2)

参数：

* image:灰度图，
* threshold1: minval，较小的阈值将间断的边缘连接起来
* threshold2: maxval，较大的阈值检测图像中明显的边缘

**总结**

边缘检测的原理

* 1. 基于搜索：利用一阶导数的最大值获取边界
  2. 基于零穿越：利用二阶导数为0获取边界

Sobel算子

基于搜索的方法获取边界

cv.sobel()

cv.convertScaleAbs()

cv.addweights()

Laplacian算子

基于零穿越获取边界

cv.Laplacian()

Canny算法

流程：

* 1. 噪声去除：高斯滤波
  2. 计算图像梯度：sobel算子，计算梯度大小和方向
  3. 非极大值抑制：利用梯度方向像素来判断当前像素是否为边界点
  4. 滞后阈值：设置两个阈值，确定最终的边界