数字图像处理实验

2022-04-15 21:16:36

# opencv中文文档：[https://opencv.apachecn.org](https://opencv.apachecn.org/#/)

[opencv中文文档：[https://opencv.apachecn.org](https://opencv.apachecn.org/#/)](#opencv中文文档httpsopencvapachecnorg)  
 [实验一 Python中数字图像处理的基本操作](#实验一-python中数字图像处理的基本操作)  
 [一、实验目的](#一实验目的-1)  
 [二、实验原理](#二实验原理)  
 [1.数字图像的表示和类别](#X8fef5b64b5a4f5248ab1824949d065d98342980)  
 [2.OpenCV图像文件格式图像文件格式](#X5199a5635702004794c3df1003a27fff5dfddd4)  
 [三、实验内容](#三实验内容)  
 [四、实验报告要求](#四实验报告要求)  
 [五、预习要求](#五预习要求)  
 [六、实验](#六实验)  
 [1.示例程序](#X549b7e981d10849d4333407af0f5cc392025208)  
 [2 实验程序](#X5c43bdcf5f25db43666d81d005ea07732a22cf1)  
 [[访问和修改像素值](https://opencv.apachecn.org/#/docs/4.0.0/3.1-tutorial_py_basic_ops?id=访问和修改像素值)](#访问和修改像素值)  
 [[变换](https://opencv.apachecn.org/#/docs/4.0.0/4.2-tutorial_py_geometric_transformations?id=变换)](#变换)  
 [[缩放](https://opencv.apachecn.org/#/docs/4.0.0/4.2-tutorial_py_geometric_transformations?id=缩放)](#缩放)  
 [[平移变换](https://opencv.apachecn.org/#/docs/4.0.0/4.2-tutorial_py_geometric_transformations?id=平移变换)](#平移变换)  
 [七、 部分实验结果](#七-部分实验结果)  
 [1 原图像](#X3a20eaf384fe3cc959bfc12be2e97ed5b5e6777)  
 [2 灰度图像](#Xf48397b2341ff07645653135912336593ef0926)  
 [3 二值图像](#Xad7fd864c846664d6df047010fa7840ca649df4)  
 [4 放大两倍后的图像：](#Xee0518fe835521a5d2f4441e8a1a49526d095e6)  
 [实验二 数字图像增强实验](#实验二-数字图像增强实验)  
 [一、实验目的](#一实验目的-2)  
 [二、实验主要仪器设备](#二实验主要仪器设备-1)  
 [三、实验原理](#三实验原理-1)  
 [1 直方图](#Xc34d9ac01e4560ad824d71f35a75f9cccbca66f)  
 [2 图像锐化](#X649ede733e2f02de6b51b661b0860832b6a4a50)  
 [3 图像平滑](#X74e8325eaff0c21bfc4c7dea40a8e723cad407a)  
 [四、实验内容](#四实验内容-1)  
 [五、实验报告要求](#五实验报告要求-1)  
 [六、预习要求](#六预习要求-1)  
 [七、思考题](#七思考题)  
 [八、实验指导](#八实验指导)  
 [1绘制灰度直方图](#X7cc6e05342ddabe5575e353b8626c2a235112e6)  
 [2.直方图均衡](#Xe628e00e70cb4ab584c359632029c116260adaf)  
 [3.空域滤波(空域平滑或锐化)](#X8aeca3a3ff50b41b2f2e38d89c04c5af13c9e98)  
 [(1)平滑滤波](#Xdc0269cdcb90881bf3cb596cf565e6fce1bda2a)  
 [均值滤波](#均值滤波)  
 [高斯模糊滤波](#高斯模糊滤波)  
 [中值滤波](#中值滤波)  
 [(2)锐化滤波](#Xd130c01b7d4a7427214f7bc9c5f4c5e735ebb04)  
 [Roberts 算子](#roberts-算子)  
 [Prewitt 算子](#prewitt-算子)  
 [Sobel 算子](#sobel-算子)  
 [4、频域滤波(频域平滑或锐化)](#X8e9f773feaf77dab49cdeea9311c19b7c2d4fa4)  
 [(1)常见低通（平滑）滤波器](#X8d3a0f1f204bd9e278f105b2e4743c828d72f50)  
 [(2)常见高通（锐化）滤波器](#X40019e848f62412f0143bb89212b2a34eb90e5e)  
 [九、实验结果](#九实验结果)  
 [(1)直方图处理：](#X033dd25e12c7ba3f85704dca25196e47ff3d261)  
 [(2)平滑与锐化](#Xbd8ccfc67de7f897f2b23755d899d155b649493)  
 [实验三 形态学图像处理实验](#实验三-形态学图像处理实验)  
 [一、实验目的](#一实验目的-3)  
 [二、实验主要仪器设备](#二实验主要仪器设备-2)  
 [三、实验原理](#三实验原理-2)  
 [四、实验内容](#四实验内容-2)  
 [五、实验报告要求](#五实验报告要求-2)  
 [六、预习要求](#六预习要求-2)  
 [七、实验](#七实验-1)  
 [1 腐蚀](#X693085c9da1966327664f05b594007a5d77a6b6)  
 [2 膨胀](#X45d794ef0041da69174a8d780f2d89aa5654175)  
 [3 开运算](#X502d78b444e2b4ee02c679187a4eb7d88aa551e)  
 [4 闭运算](#X8dde4e5d61801d742dce08ce3c782796979b414)  
 [实验四 数字图像的边缘检测实验](#实验四-数字图像的边缘检测实验)  
 [一、实验目的](#一实验目的-4)  
 [二、实验主要仪器设备](#二实验主要仪器设备-3)  
 [三、实验原理](#三实验原理-3)  
 [四、实验内容](#四实验内容-3)  
 [五、实验报告要求](#五实验报告要求-3)  
 [六、预习要求](#六预习要求-3)  
 [七、实验](#七实验-2)  
 [1 LoG 检测器](#Xe9c0ae4992f5122c7023caf08f54af7b5456302)  
 [2 Scharr 算子](#X723e71f2d92c0087a57ef3716b1750bb5d53e60)  
 [3 Canny 边缘检测器](#X5c60d981e78997c012c85498fd64ffb7905ab10)  
 [实验五 基于图像分割的车牌定位识别](#实验五-基于图像分割的车牌定位识别)  
 [一、实验目的](#一实验目的-5)  
 [二、实验主要仪器设备](#二实验主要仪器设备-4)  
 [三、实验原理](#三实验原理-4)  
 [四、实验内容](#四实验内容-4)  
 [五、实验报告](#五实验报告)  
 [六、预习要求](#六预习要求-4)  
 [七、实验](#七实验-3)

## 实验一 Python中数字图像处理的基本操作

### 一、实验目的

1 熟悉及掌握在 python中能够处理哪些格式图像；

2 熟练掌握在 python中如何 用 OpenCV读取图像；

3 掌握如何利用 python来获取图像的大小、颜色、高度、宽度等等相关信息；

4 掌握如何在 python中 用 OpenCV按照指定要求存储一幅图像的方法；

5 图像间如何转化。

### 二、实验原理

#### 1.数字图像的表示和类别

一幅图像可以被定义为一个二维函数f(x,y),其中 x和 y是空间 (平面 )坐标， f 在任何坐标 (x,y)处的振幅称为图像在该点的亮度。灰度是用来表示黑白图像亮度的一个术语，而彩色图像是由单个二维图像组合形成的。例如，在 RGB彩色系统中，一幅彩色图像是由三幅独立的分量图像 (红、绿、蓝 )组成的。因此，许多为黑白图像处理开发的技术适用于彩色图像处理，方法是分别处理三副独立的分量图像即可。

要将这样的一幅图像转化为数字形式，就要求数字化坐标和振幅。将坐标值数字化成为取样；将振幅数字化成为量化。采样和量化的过程如图 1所示。因此，当 f的 x、 y分量和振幅都是有限且离散的量时，称该图像为数字图像。

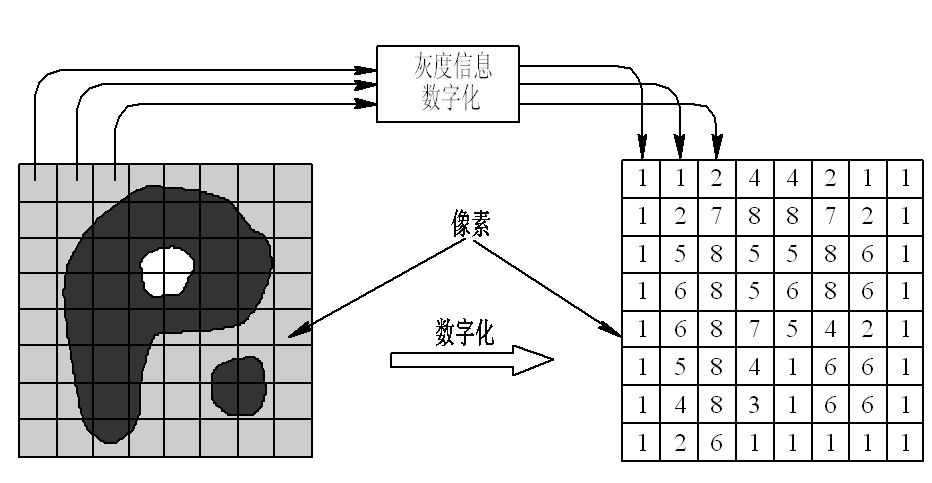


image-20220415212141944

图1 图像的采样和量化

在python中，一幅图像可能包含一个数据矩阵，也可能有一个颜色映射表矩阵。 opencv支持四种图像类型，其区别在于数据矩阵元素的不同含义。它们是：

 亮度图像（ Intensity images）

 二值图像（ Binary images）

 索引图像（ Indexed images）

 RGB图像（ RGB images）

(1)亮度图像。

也称灰度图像。一幅亮度图像是一个数据矩阵，其归一化的取值表示亮度。若亮度图像一幅亮度图像是一个数据矩阵，其归一化的取值表示亮度。若亮度图像的像素都是的像素都是uint8类或类或uint16类，则它们的整数值范围分别是类，则它们的整数值范围分别是[0，255]和和[0，65536]。若。若图像图像是是double类，则像素取值就是浮点数。规定双精度型归一化亮度图像的取值范围是类，则像素取值就是浮点数。规定双精度型归一化亮度图像的取值范围是[0，1]。。

(2)二值图像

二值图像是指在图像中，每个像素的灰度等级只有两种。即全黑或者全白，在二值图像是指在图像中，每个像素的灰度等级只有两种。即全黑或者全白，在python种，种，一幅二值图像是一个取值只有一幅二值图像是一个取值只有0和和255的的numpy数组数组。

(3)索引图像

索引颜色通常也称为映射颜色，在这种模式下，颜色都是预先定义的，并且可供选用的索引颜色通常也称为映射颜色，在这种模式下，颜色都是预先定义的，并且可供选用的一组颜色也很有限，索引颜色的图像最多只能显示一组颜色也很有限，索引颜色的图像最多只能显示256种颜色。种颜色。一幅索引颜色图像在图像文件里定义，当打开该文件时，构成该图像具体颜色的索引值一幅索引颜色图像在图像文件里定义，当打开该文件时，构成该图像具体颜色的索引值就被读入程序里，然后根据索引值找到最终的颜色。就被读入程序里，然后根据索引值找到最终的颜色。

(4) RGB图像

一幅RGB图像就是彩色像素的一个图像就是彩色像素的一个M×N×3数组，其中每一个彩色相似点都是在特定数组，其中每一个彩色相似点都是在特定空间位置的彩色图像相对应的红、绿、蓝三个分量。按照惯例，形成一幅空间位置的彩色图像相对应的红、绿、蓝三个分量。按照惯例，形成一幅RGB彩色图像的彩色图像的三个图像常称为红、绿或蓝分量图像。三个图像常称为红、绿或蓝分量图像。

值得注意的是，在OpenCV中，加载图像通道的顺序是中，加载图像通道的顺序是BGR（实验五中您将用到此特（实验五中您将用到此特性）性）。。但是但是Matplotlib（（python的一个绘图库）以的一个绘图库）以RGB模式显示。因此，如果使用模式显示。因此，如果使用OpenCV读取彩色图像，则读取彩色图像，则Matplotlib中将无法正确显示彩色图像。中将无法正确显示彩色图像。

#### 2.OpenCV图像文件格式图像文件格式

OpenCV支持支持处理处理以下几种图像文件格式：以下几种图像文件格式：

1 PCX（Windows Paintbrus），可处理1，4，8，16，24位等图像数据。文件位等图像数据。文件内容包括：文件头（内容包括：文件头（128字节），图像数据、扩展颜色映射表数据。字节），图像数据、扩展颜色映射表数据。

2 BMP（Windows Bitmap格式。有1，4，8，24位非压缩图像，位非压缩图像，8位位RLE（（Run-length Encoded）图像。文件内容包括：文件头（一个）图像。文件内容包括：文件头（一个BITMAP FILEHEADER数据结构），数据结构），位图信息数据块（位图信息头位图信息数据块（位图信息头BITMAP INFOHEADER和一个颜色表）和图像数据。和一个颜色表）和图像数据。

3 HDF（（Hierarchical Data Format）格式。有8位，位，24位光栅数据集。位光栅数据集。

4 JPEG(Joint Photographic Experts Group)格式，是一种成为联合图像专家组的图像压格式，是一种成为联合图像专家组的图像压缩格式。缩格式。

5 TIFF（Tagged Image File Format）格式，处理1，4，8，24位非压缩图像，位非压缩图像，1，，4，，8，，24位位packbit压缩图像，一位压缩图像，一位CCITT压缩图像等。文件内容压缩图像等。文件内容包括：文件头，参数指针表包括：文件头，参数指针表与参数域，参数数据表和图像数据四部分。与参数域，参数数据表和图像数据四部分。

6 XWD(X Windows Dump)格式。1，8位位Zpixmaps,XYbitmaps,1位位XYpixmaps。。

7 PNG（Portable Network Graphics）格式。

### 三、实验内容

1利用OpenCV读取一幅彩色图像，并读取图像的基本信息；

2 利用OpenCV显示图像；

3 对彩色图像进行灰度化化处理；

4 对灰度图像进行二值化处理；

5 对图像进行几何变换（缩放，平移，翻转）；

6 储存处理后的图像；

### 四、实验报告要求

1 给出使用给出使用opencv-python进行图像读取、显示、翻转、裁剪，存储的完整代码。

2 写出实验的心得与体会。

### 五、预习要求

1 了解python 基本语法以及基本语法以及图像处理API--OpenCV。

2 了解opencv图像基础操作函数。

### 六、实验

#### 1.示例程序

import cv2 as cv #引入 OpenCV库  
img = cv.imread('1.jpg') #使用 imread函数读取图像 ，并以 numpy数组形式储存  
print(img.shape) #查看图像的大小。返回的元组（ touple）中的三个数依次表示高度、宽度和通道数  
print(img.dtype) #查看图片的类型  
cv.imshow('img',img) #使用 imshow函数显示图像，第一个参数是窗口名称（可不写），第二个参数是要显示的图像的名称，一定要写  
cv.waitKey(0) #可以让窗口一直显示图像直到按下任意按键  
img\_GRAY = cv.cvtColor(img,cv.COLOR\_BGR2GRAY) #使用 cv.cvtColor函数转换色彩空间 参数 ‘cv.COLOR\_BGR2GRAY’表示从RGB空间转换到灰度空间  
cv.imshow('gray',img\_GRAY)  
cv.waitKey(0)  
ret,thresh = cv.threshold(img\_GRAY,127,255,cv.THRESH\_BINARY) #使用 cv.threshold函数进行图像阈值处理 参数‘cv.THRESH\_BINARY’代表了阈值的类型 127为阈值  
cv.imshow('threshold',thresh)  
cv.waitKey(0)  
res = cv.resize(img,None,fx=2,fy=2,interpolation=cv.INTER\_CUBIC) #使用 cv.resize函数进行图像缩放  
cv.imshow('resize',res)  
cv.waitKey(0)  
cv.imwrite('result.jpg',res) #保存图像

#### 2 实验程序

1利用OpenCV读取一幅彩色图像，并读取图像的基本信息；

Python:  
retval = cv.imread( filename[, flags] )  
从文件加载图像。函数IMRead从指定文件加载图像并返回它。 如果无法读取图像（由于缺少文件，不正确的权限，不受支持或无效格式），则该函数返回一个空矩阵（MAT :: data == null）。

##### [访问和修改像素值](https://opencv.apachecn.org/#/docs/4.0.0/3.1-tutorial_py_basic_ops?id=访问和修改像素值)

先来理解一下，图像与一般的矩阵或张量有何不同(不考虑图像的格式，元数据等信息)。首先，一张图像有自己的属性，宽，高，通道数。其中宽和高是我们肉眼可见的属性，而通道数则是图像能呈现色彩的属性。我们都知道，光学三原色是红色，绿色和蓝色，这三种颜色的混合可以形成任意的颜色。常见的图像的像素通道也是对应的R，G，B三个通道，在OpenCV中，每个通道的取值为0～255，。(注：还有RGBA，YCrCb，HSV等其他图像通道表示)。即，一般彩色图像读进内存之后是一个h \* w \* c的矩阵，其中h为图像高(相当于矩阵的行)，w为图像宽(相当于矩阵列)，c为通道数。

下面我们先加载一副彩色图像，更准确的说，是一副黄色图像，如图所示。



黄色为绿色和红色的混合，所以，该图像的所有像素值都应为R=255，G=255，B=0

>>> import numpy as np  
>>> import cv2  
>>> img = cv2.imread("img/yellow.jpg")  
>>> h,w,c = img.shape  
#图像为128\*128\*3的大小  
>>> print(h,w,c)  
128 128 3

从上面的代码中可以看到，您可以通过行和列坐标访问像素值。注意,对于 常见的RGB 图像，OpenCV的imread函数返回的是一个蓝色(Blue)、绿色(Green)、红色(Red)值的数组，维度大小为3。而对于灰度图像，仅返回相应的强度。

>>> img[100,100]  
#OpenCV的读取顺序为B，G，R，由于图像所有像素为黄色，因此，G=255，R=255  
array([ 0, 255, 255], dtype=uint8)  
  
# 仅访问蓝色通道的像素  
>>> blue = img[100,100,0]  
>>> print(blue)  
0复制ErrorOK!

你也可以使用同样的方法来修改像素值

>>> img[100,100] = [255,255,255]  
>>> print(img[100,100])  
[255 255 255]

2 利用OpenCV显示图像；

cv.imshow('img',img) #使用 imshow函数显示图像，第一个参数是窗口名称（可不写），第二个参数是要显示的图像的名称，一定要写  
cv.waitKey(0) #可以让窗口一直显示图像直到按下任意按键

3 对彩色图像进行灰度化化处理；

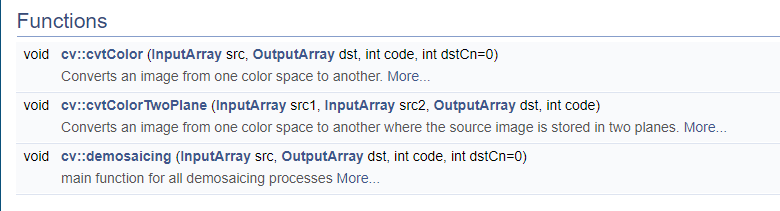


image-20220425232528758

img\_GRAY = cv.cvtColor(img,cv.COLOR\_BGR2GRAY) #使用 cv.cvtColor函数转换色彩空间 参数 ‘cv.COLOR\_BGR2GRAY’表示从RGB空间转换到灰度空间  
cv.imshow('gray',img\_GRAY)  
cv.waitKey(0)

4 对灰度图像进行二值化处理；

ret,thresh = cv.threshold(img\_GRAY,127,255,cv.THRESH\_BINARY) #使用 cv.threshold函数进行图像阈值处理 参数‘cv.THRESH\_BINARY’代表了阈值的类型 127为阈值  
cv.imshow('threshold',thresh)  
cv.waitKey(0)

5 对图像进行几何变换（缩放，平移，翻转）；

[图像的几何变换 (apachecn.org)](https://opencv.apachecn.org/#/docs/4.0.0/4.2-tutorial_py_geometric_transformations)

###### [变换](https://opencv.apachecn.org/#/docs/4.0.0/4.2-tutorial_py_geometric_transformations?id=变换)

OpenCV 提供了两个转换函数，[**cv.warpAffine**](https://docs.opencv.org/4.0.0/da/d54/group__imgproc__transform.html#ga0203d9ee5fcd28d40dbc4a1ea4451983) 和 [**cv.warpPerspective**](https://docs.opencv.org/4.0.0/da/d54/group__imgproc__transform.html#gaf73673a7e8e18ec6963e3774e6a94b87)**\**，可以进行各种转换。 \****[**cv.warpAffine**](https://docs.opencv.org/4.0.0/da/d54/group__imgproc__transform.html#ga0203d9ee5fcd28d40dbc4a1ea4451983) 采用 2x3 变换矩阵，而 [**cv.warpPerspective**](https://docs.opencv.org/4.0.0/da/d54/group__imgproc__transform.html#gaf73673a7e8e18ec6963e3774e6a94b87) 采用 3x3 变换矩阵作为输入。

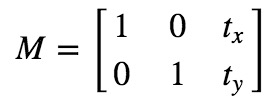
###### [缩放](https://opencv.apachecn.org/#/docs/4.0.0/4.2-tutorial_py_geometric_transformations?id=缩放)

缩放是调整图片的大小。 OpenCV 使用 [**cv.resize()**](https://docs.opencv.org/4.0.0/da/d54/group__imgproc__transform.html#ga47a974309e9102f5f08231edc7e7529d) 函数进行调整。可以手动指定图像的大小，也可以指定比例因子。可以使用不同的插值方法。对于下采样(图像上缩小)，最合适的插值方法是 [**cv.INTER\_AREA**](https://docs.opencv.org/4.0.0/da/d54/group__imgproc__transform.html#gga5bb5a1fea74ea38e1a5445ca803ff121acf959dca2480cc694ca016b81b442ceb) 对于上采样(放大),最好的方法是 [**cv.INTER\_CUBIC**](https://docs.opencv.org/4.0.0/da/d54/group__imgproc__transform.html#gga5bb5a1fea74ea38e1a5445ca803ff121a55e404e7fa9684af79fe9827f36a5dc1) （速度较慢）和 [**cv.INTER\_LINEAR**](https://docs.opencv.org/4.0.0/da/d54/group__imgproc__transform.html#gga5bb5a1fea74ea38e1a5445ca803ff121ac97d8e4880d8b5d509e96825c7522deb) (速度较快)。默认情况下，所使用的插值方法都是 [**cv.INTER\_AREA**](https://docs.opencv.org/4.0.0/da/d54/group__imgproc__transform.html#gga5bb5a1fea74ea38e1a5445ca803ff121acf959dca2480cc694ca016b81b442ceb) 。你可以使用如下方法调整输入图片大小：

import numpy as np  
import cv2 as cv  
img = cv.imread('messi5.jpg')  
res = cv.resize(img,None,fx=2, fy=2, interpolation = cv.INTER\_CUBIC)  
#OR  
height, width = img.shape[:2]  
res = cv.resize(img,(2\*width, 2\*height), interpolation = cv.INTER\_CUBIC)

###### [平移变换](https://opencv.apachecn.org/#/docs/4.0.0/4.2-tutorial_py_geometric_transformations?id=平移变换)

平移变换是物体位置的移动。如果知道 **（x，y）** 方向的偏移量，假设为 **(t\_x,t\_y)\**，则可以创建如下转换矩阵 \**M**：



您可以将变换矩阵存为 np.float32 类型的 numpy 数组，并将其作为 [**cv.warpAffine**](https://docs.opencv.org/4.0.0/da/d54/group__imgproc__transform.html#ga0203d9ee5fcd28d40dbc4a1ea4451983) 的第二个参数。请参见以下转换（100,50）的示例：

import numpy as np  
import cv2 as cv  
img = cv.imread('messi5.jpg',0)  
rows,cols = img.shape  
M = np.float32([[1,0,100],[0,1,50]])  
dst = cv.warpAffine(img,M,(cols,rows))  
cv.imshow('img',dst)  
cv.waitKey(0)  
cv.destroyAllWindows()

6 储存处理后的图像；

cv.imwrite('result.jpg',res) #保存图像

### 七、 部分实验结果

#### 1 原图像



#### 2 灰度图像



#### 3 二值图像



#### 4 放大两倍后的图像：



## 实验二 数字图像增强实验

#### 一、实验目的

1.熟悉并学会 opencv python 中图像增强的相关函数

2.了解图像增强的方法、去噪的方法和效果 。

### 二、实验主要仪器设备

(1)计算机

(2)python 软件

(3)典型的灰度、彩色图像文件 。

### 三、实验原理

图像增强是指按特定的需要突出一幅图像中的某些信息，同时，消弱或去除某些不需要的信息的处理方法。其主要目的是处理后的图像对某些特定的应用比原来的图像更加有效。图像增强技术主要有直方图修改处理、图像平滑化处理、图像尖锐化处理和彩色处理技术等。本实验以直方图均衡化 增强图像对比度的方法为主要内容，其他方法同学们可以在课后自行练习 。

#### 1 直方图

直方图是多种空间域处理技术的基础。直方图操作能有效地用于图像增强。除了提供有用的图像统计资料外，直方图固有的信息在其他图像处理应用中也是非常有用的，如图像压缩与分割。直方图在软件中易于计算，也适用于商用硬件设备，因此，它们成为了实时图像处理的一个流行工具。

直方图是图像的最基本的统计特征，它反映的是图像的灰度值的分布情况。直方图均衡化的目的是使图像在整个灰度值动态变化范围内的分布均匀化，改善图像的亮度分布状态，增强图像的视觉效果 。灰度直方图是图像预处理中涉及最广泛的基本概念之一。

图像的直方图事实上就是图像的亮度分布的概率密度函数，是一幅图像的所有象素集合的最基本的统计规律。直方图反映了图像的明暗分布规律，可以通过图像变换进行直方图调整，获得较好的视觉效果。

直方图均衡化是通过灰度变换将一幅图像转换为另一幅具有均衡直方图，即在每个灰度级上都具有相同的象素点数的过程。

#### 2 图像锐化

图像锐化(image 是补偿图像的轮廓，增强图像的边缘及灰度跳变的部分，使图像变得清晰，分为空域处理和频域处理两类。

#### 3 图像平滑

图像平滑是对图像作低通滤波 ，可在空间域或频率域实现。

### 四、实验内容

1 绘制灰度图像直方图；

2 对直方图均衡化

3利用模版进行空域滤波

4 分别利用常见低通（平滑）滤波器与高通（锐化）滤波器进行频域滤波（滤波器公式见实验指导）

### 五、实验报告要求

(1)说明利用 opencv python 图像处理 工具包 实现灰度修正、图像平滑、锐化的方法

(2)列出上述图像处理的程序

(3)记录灰度修正、图像平滑、图像锐化的图像，回答思考题

(4)心得和体会 。

### 六、预习要求

(1)了解 opencv python 图像处理 包 关于图像 增强的有关功能；

(2)列出上述图像处理的流程。

### 七、思考题

(1)如何针对图像过暗、过亮、对比度不足设计灰度变换函数？

(2)比较同一种去噪方法对不同噪声处理的效果。

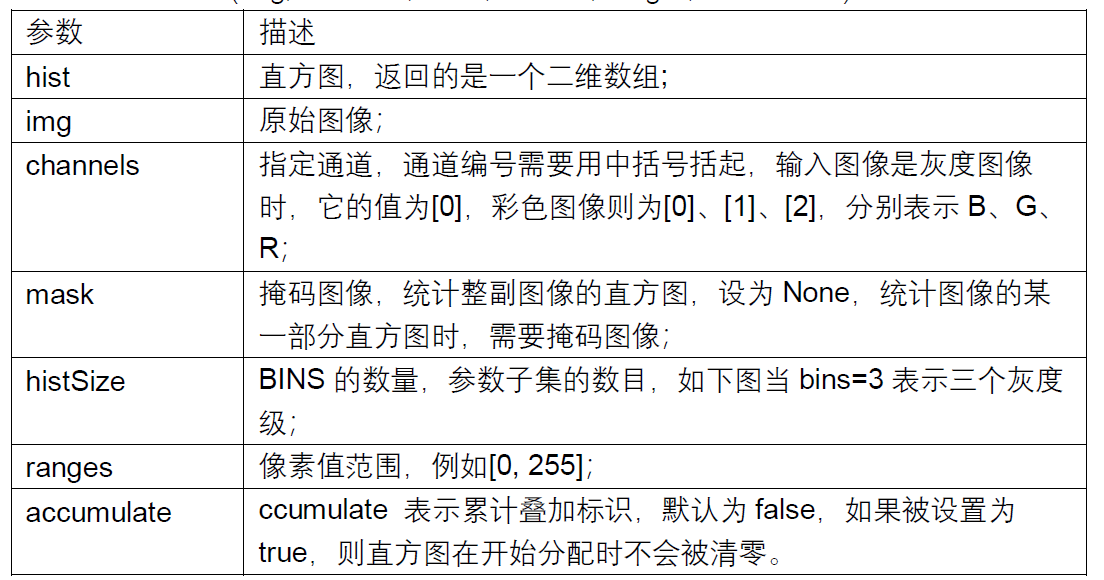
(3)讨论梯度法锐度图像的 4 种不同方法的应用范围。

### 八、实验指导

#### 1绘制灰度直方图

OpenCV利用 calcHist() 函数来绘制直方图 ca lcHist() 函数原型为：

hist = cv2.calcHist(img , channels, mask, histSize, ranges, accumulate)



由于calcHist() 函数返回的是二维数组，可利用 matplotlib 库 绘制图像。部分代码示例如下：

import cv2  
import numpy as np  
import matplotlib.pyplot as plt  
#读取图片  
img = cv2.imread('images/buffalos.png',0)  
hist = cv2.calcHist([img], [0], None, [256], [0, 255])  
plt.plot(hist)  
plt.show()

#### 2.直方图均衡

OpenCV 中的直方图均衡化函数为 cv2.equalizeHist()。该函数的输入为灰度图像，输出结果为直方图均衡化后的图像。

equ = cv2.equalizeHist(img)

#### 3.空域滤波(空域平滑或锐化)

##### (1)平滑滤波

平滑滤波是低频增强的空间域滤波技术。它的目的有两类：一类是模糊；另一类是消除噪音。空间域的平滑滤波一般采用简单平均法进行，就是求邻近像元点的平均亮度值。邻域的大小与平滑的效果直接相关，邻域越大平滑的效果越好，但邻域过大，平滑会使边缘信息损失的越大，从而使输出的图像变得模糊，因此需合理选择邻域的大小。

###### 均值滤波

OpenCV 中均值模板可以用cv2.blur 和cv2.boxFilter 函数实现。其基本用法如下：

blur = cv2.blur(img, (3, 5)) # 模板大小为3\*5, 模板的大小是可以设定的  
box = cv2.boxFilter(img, -1, (3, 5))

###### 高斯模糊滤波

Opencv 中使用cv2.GaussianBlur()函数实现高斯模糊。其基本用法如下：

blur = cv2.GaussianBlur(img, (5, 5), 0) # （5,5）表示的是卷积模板的大小，0 表示的是沿x 与y 方向上的标准差

###### 中值滤波

Opencv 用cv2.medianBlur()函数实现中值滤波。其基本用法如下：

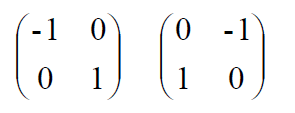
blur = cv2.medianBlur(img, 5)

##### (2)锐化滤波

###### Roberts 算子

Roberts 算法简介

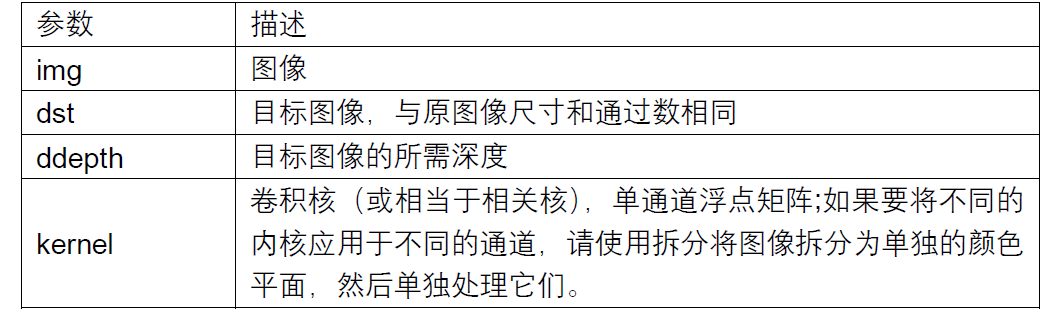
Roberts 算法又称为交叉微分算法，它是基于交叉差分的梯度算法，通过局部差分计算检测边缘线条。常用来处理具有陡峭的低噪声图像，当图像边缘接近于正45 度或负45 度时，该算法处理效果更理想。Roberts 算子的模板分为水平方向和垂直方向，如下所示：



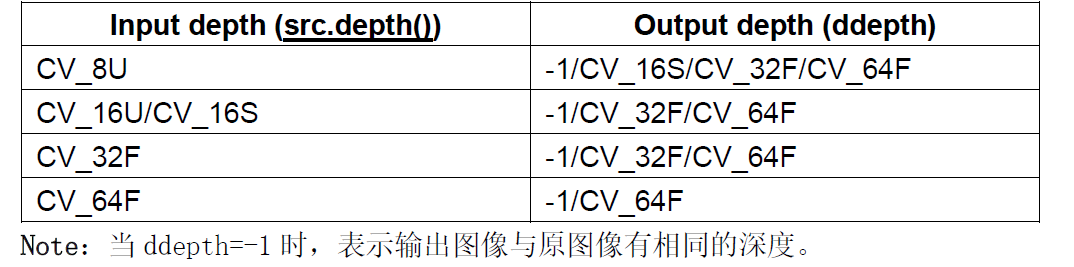
OpenCV 中Roberts 算法库函数使用

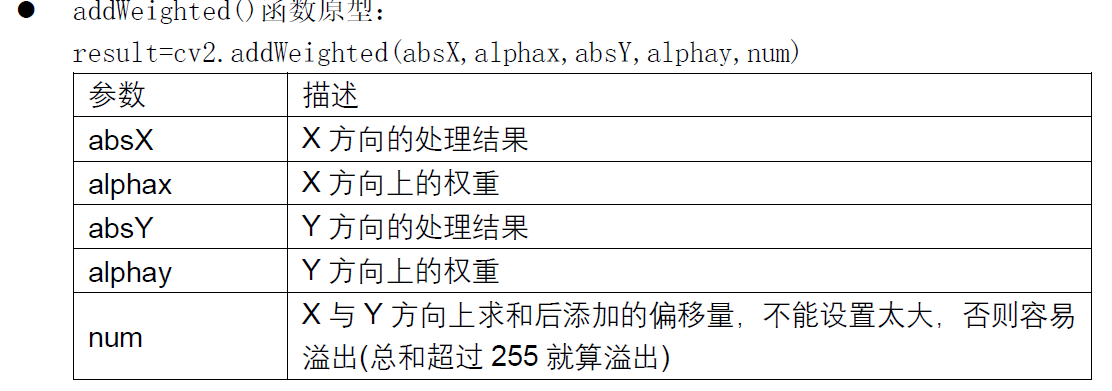
在OpenCV 官方的库中，也提供了 Roberts 算法库函数，调用 OpenCV 的 filter2D() 函数实现边缘提取。该函数主要是利用内核实现对图像的卷积运算，然后通过 addWeighted()函数来进行 x 方向与 y 方向上的结合 。

 filter2D() 函数原型： result=cv2.filter2D( img , cv2.CV\_16S,



其中 ddepth表示目标图像的所需深度，它包含有关图像中存储的数据类型的信息，可以是 unsigned char CV\_8U signed char CV\_8S unsigned short CV\_16U等等 ...





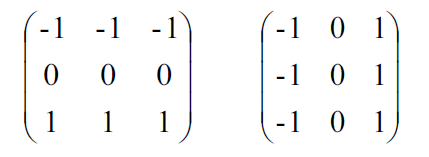
代码示例如下：

import cv2  
import numpy as np  
'''  
一般来说，对图像轮廓提取都会经过如下步骤，灰度  
滤波去噪 阈值化处理 形态学处  
理，前面如果达标，这步骤可以省略 轮廓提取  
'''  
#  
读取图像  
img=cv2.imread("images/building.jpg")  
#  
图像高斯滤波去噪  
blur=cv2.GaussianBlur(img,(3,3),1,1)  
核尺寸通过对图像的调节自行定义  
#  
图像阈值化处理  
ret,thresh1=cv2.threshold(blur,127,255,cv2.THRESH\_BINARY) #  
二进制阈值化  
#  
调用 Roberts 算法的 OpenCV 库函数进行图像轮廓提取  
  
kernelx = np.array([[-1,0],[0,1]], dtype=int)  
kernely = np.array([[0,-1],[1,0]], dtype=int)  
x = cv2.filter2D(thresh1, cv2.CV\_16S, kernelx)  
y = cv2.filter2D(thresh1, cv2.CV\_16S, kernely)  
#转uint8  
absX = cv2.convertScaleAbs(x)  
absY = cv2.convertScaleAbs(y)  
Roberts = cv2.addWeighted(absX,0.5,absY,0.5,0)  
cv2.imshow("Roberts",Roberts)  
cv2.waitKey()

###### Prewitt 算子

Prewitt 算法简介

Prewitt 算子是一种一阶微分算子的边缘检测，利用像素点上下、左右邻点的灰度差，在边缘处达到极值检测边缘，去掉部分伪边缘，对噪声具有平滑作用 。其原理是在图像空间利用两个方向模板与图像进行邻域卷积来完成的，这两个方向模板一个检测水平边缘，一个检测垂直边缘。Prewitt 算法适合用来识别噪声较多、灰度渐变的图像；水平和竖直方向上的卷积模板如下所示：



OpenCV 中Prewitt 算法库函数使用

OpenCV 官方同样对Prewitt 算法有对应的库函数，与Roberts 算法算法的库函数一样，只是传递的卷积核有变化；通过Numpy 定义模板，再调用OpenCV 的filter2D()函数实现对图像的卷积运算，最终通过convertScaleAbs()和addWeighted()函数实现边缘提取，函数原型参考Roberts 算法介绍的函数原型。

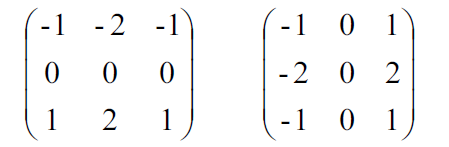
部分代码示例如下：

#调用Prewitt 算法的OpenCV 库函数进行图像轮廓提取  
kernelx = np.array([[1,1,1],[0,0,0],[-1,-1,-1]],dtype=int)  
kernely = np.array([[-1,0,1],[-1,0,1],[-1,0,1]],dtype=int)  
x = cv2.filter2D(thresh1, -1, kernelx)  
y = cv2.filter2D(thresh1, -1, kernely)

###### Sobel 算子

Sobel 算法简介

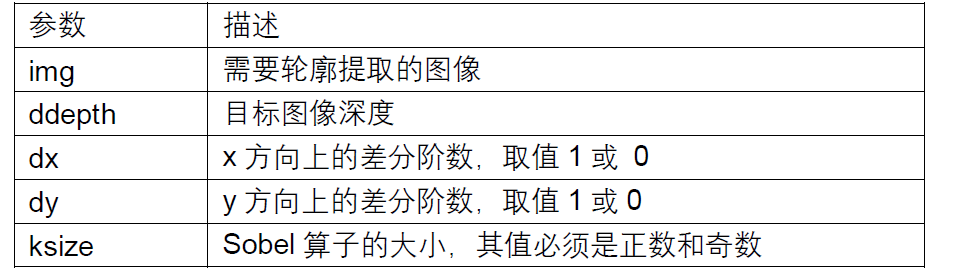
Sobel 算法(索贝尔算子)是一种用于边缘检测的离散微分算子，它结合了高斯平滑和微分求导。该算子用于计算图像明暗程度近似值，根据图像边缘旁边明暗程度把该区域内超过某个数的特定点记为边缘。Sobel 算子在Prewitt 算子的基础上增加了权重的概念，认为相邻点的距离远近对当前像素点的影响是不同的，距离越近的像素点对应当前像素的影响越大，从而实现图像锐化并突出边缘轮廓。当对精度要求不是很高时，Sobel 算子是一种较为常用的边缘检测方法，其卷积模板如下所示：



OpenCV 中Roberts 算法库函数使用

OpenCV 中提供了专门的Soble 算法的库函数,函数原型：

result=cv2.Sobel(img, ddepth, dx, dy, ksize)

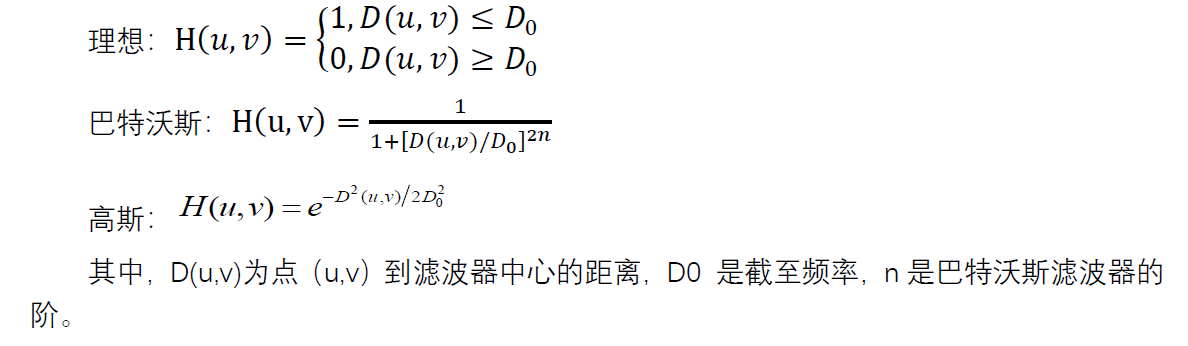


部分代码示例如下：

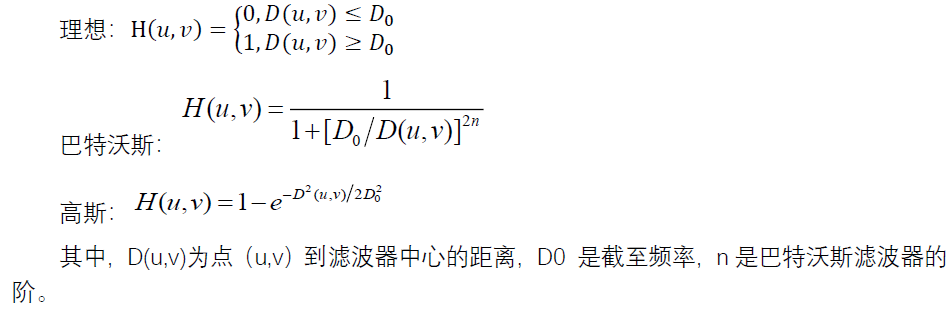
#调用Sobel 算法的OpenCV 库函数进行图像轮廓提取  
x = cv2.Sobel(thresh1, cv2.CV\_16S, 1, 0) #对x 求一阶导  
y = cv2.Sobel(thresh1, cv2.CV\_16S, 0, 1) #对y 求一阶导  
absX = cv2.convertScaleAbs(x) #对x 取绝对值，并将图像转换为8 位图  
absY = cv2.convertScaleAbs(y) #对y 取绝对值，并将图像转换为8 位图  
Sobel = cv2.addWeighted(absX, 0.5, absY, 0.5, 0)

#### 4、频域滤波(频域平滑或锐化)

##### (1)常见低通（平滑）滤波器



##### (2)常见高通（锐化）滤波器

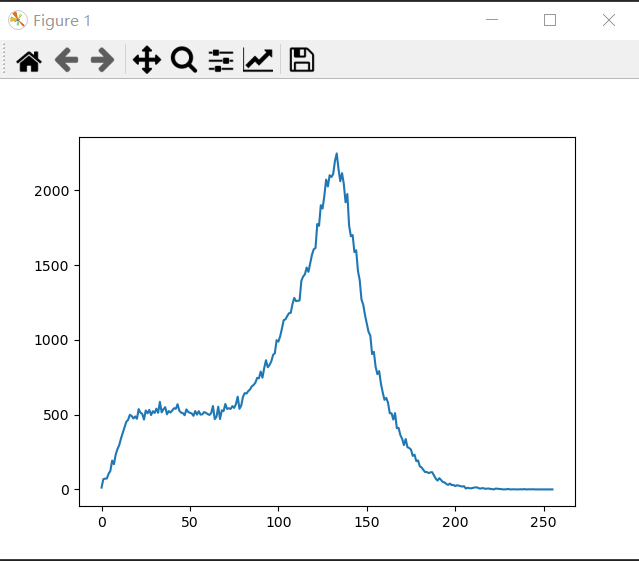


#### 九、实验结果

##### (1)直方图处理：



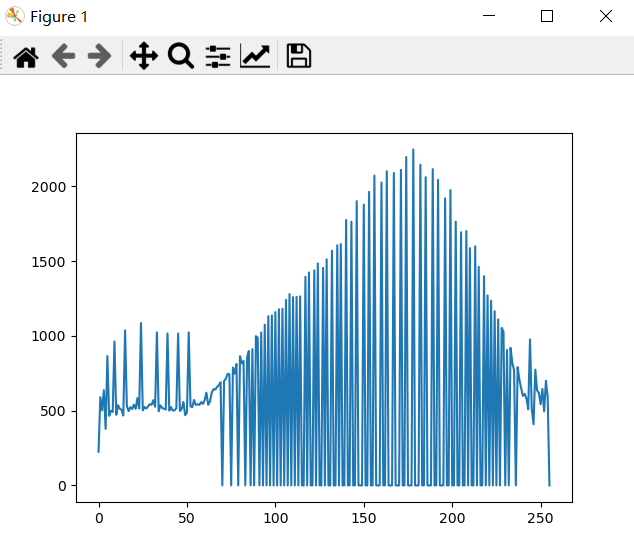
原图像



灰度直方图



直方均衡后的图像



直方图均 衡 后的 灰度直方图

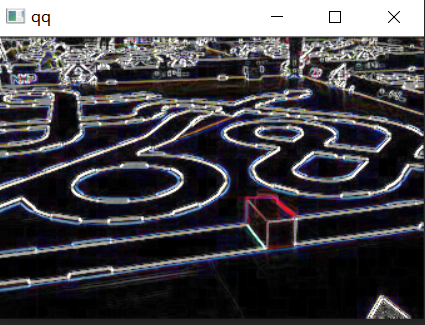
代码如下：

import cv2  
  
import numpy as np  
import matplotlib.pyplot as plt  
img=cv2.imread("1.jpg")  
cv2.imshow("aaa",img)  
cv2.waitKey(0)  
img\_GRAY = cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY) #使用 cv.cvtColor函数转换色彩空间 参数 ‘cv.COLOR\_BGR2GRAY’表示从RGB空间转换到灰度空间  
cv2.imshow('gray',img\_GRAY)  
hist = cv2.calcHist([img\_GRAY], [0], None, [256], [0, 255])  
plt.plot(hist)  
plt.show()  
equ = cv2.equalizeHist(img\_GRAY)  
cv2.imshow("aaa1",equ)  
cv2.waitKey(0)  
hist = cv2.calcHist([equ], [0], None, [256], [0, 255])  
plt.plot(hist)  
plt.show()

##### (2)平滑与锐化



原图像



Sobel 算子

代码如下：

import cv2 as cv  
  
img=cv.imread("2.jpg")  
x = cv.Sobel(img, cv.CV\_16S, 1, 0) #对x 求一阶导  
y = cv.Sobel(img, cv.CV\_16S, 0, 1) #对y 求一阶导  
absX = cv.convertScaleAbs(x) #对x 取绝对值，并将图像转换为8 位图  
absY = cv.convertScaleAbs(y) #对y 取绝对值，并将图像转换为8 位图  
Sobel = cv.addWeighted(absX, 0.5, absY, 0.5, 0)  
cv.imshow("qq",Sobel)  
cv.waitKey(0)

## 实验三 形态学图像处理实验

### 一、实验目的

1 使用形态学滤波对图像进行腐蚀、膨胀运算

2 使用形态学滤波对图像进行开闭运算

3 利用 opencv python 对图像进行形态学运算

### 二、实验主要仪器设备

(1)计算机

(2)python 软件

(3)典型的灰度、彩色图像文件 。

### 三、实验原理

数学形态学是以形态结构元素为基础对图像进行分析的数学工具。它的基本思想是用具有一定形态的结构元素去度量和提取图像中的对应形状以达到对图像分析和识别的目的。数学形态学的应用可以简化图像数据，保持它们基本的形状特征，并除去不相干的结构。数学形态学的基本运算有 4 个：膨胀、腐蚀、开启和闭合。它们在二值图像中和灰度图像中各有特点。基于这些基本运算还可以推导和组合成各种数学形态学实用算法。

基本的形态运算是腐蚀和膨胀。

在形态学中，结构元素是最重要最基本的概念。结构元素在形态变换中的作用相当于信号处理中的 滤波窗口 。用代表结构元素，对工作空间中的每一点 ，腐蚀和膨胀的定义为：

腐蚀;

膨胀。

用 对 进行膨胀的结果就是把结构元素 B 平移后使 B 与 E 的交集非空的点构成的集合。先腐蚀后膨胀的过程称为开运算。它具有消除细小物体 ，在纤细处分离物体和平滑较大物体边界的作用。先膨胀后腐蚀的过程称为闭运算。它具有填充物体内细小空洞，连接邻近物体和平滑边界的作用。

可见，二值形态膨胀与腐蚀可转化为集合的逻辑运算，算法简单，适于并行处理，且易于硬件实现，适于对二值图像进行图像分割、细化、抽取骨架、边缘提取、形状分析。但是，在不同的应用场合，结构元素的选择及其相应的处理算法是不一样的，对不同的目标图像需设计不同的结构元素和不同的处理算法。结构元素的大小、形状选择合适与否，将直接影响图像的形态运算结果。因此，很多学者结合自己的应用实际，提出了一 系列的改进算法。如梁勇提出的用多方位形态学结构元素进行边缘检测算法既具有较好的边缘定位能力，又具有很好的噪声平滑能力。许超提出的以最短线段结构元素构造准圆结构元素或序列结构元素生成准圆结构元素相结合的设计方法，用于骨架的提取，可大大减少形态运算的计算量，并可同时满足尺度、平移及旋转相容性，适于对形状进行分析和描述。

### 四、实验内容

(1)对原始图像进行灰度化、二值化处理

(2)对所得二值图像进行腐蚀运算；

(3)对二值图像进行膨胀运算；

(4)对二值图像进行开运算；

(5)对二值图像进行闭运算。

### 五、实验报告要求

(1)说明对图像进行形态学运算的理论方法

(2)列出上述图像处理程序

(3)记录结果图像

(4)心得和体会 。

### 六、预习要求

(1)了解图像形态学运算的方法

(2)列出上述图像处理方法的流程 。

### 七、实验

#### 1 腐蚀

腐蚀主要就是调用cv2.erode(img,kernel,iterations) 其中 第一个参数： img指需要腐蚀的图 第二个参数： kernel 指腐蚀操作的内核，默认是一个简单的 3X3矩阵 第三个参数： iterations 指的是腐蚀次数，省略是默认为 1 。

使用3 X 3 的卷积核，对二值图进行腐蚀操作的示例如下：

import cv2  
import numpy as np  
img = cv2.imread("./image-20220425233418464.png",0)  
kernel = np.ones((3,3),np.uint8)  
erosion = cv2.erode(img,kernel)  
cv2.imshow('eroded image',erosion)  
cv2.waitKey(0)  
cv2.destroyAllWindows()



原图像



腐蚀图像

#### 2 膨胀

代码如下

import cv2  
import numpy as np  
img = cv2.imread("./image-20220425233418464.png",0)  
kernel = np.ones((3,3),np.uint8)  
erosion = cv2.erode(img,kernel)  
cv2.imshow('eroded image',erosion)  
cv2.waitKey(0)  
cv2.destroyAllWindows()  
dilation = cv2.dilate(img,kernel,iterations = 1)  
cv2.imshow('dilation',dilation)  
cv2.waitKey(0)



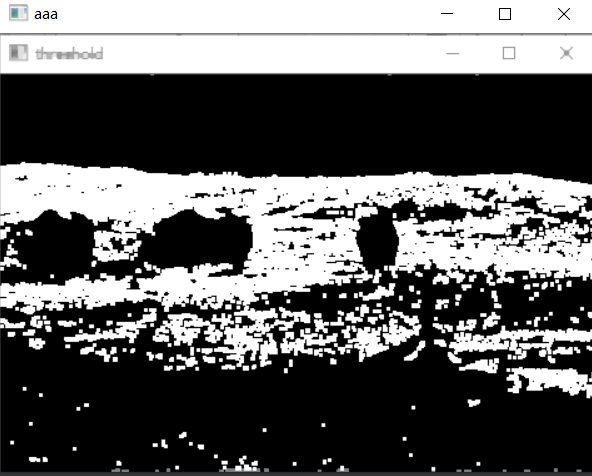
膨胀图像

#### 3 开运算

openvc 中 morphologyEx() 函数是一种形态学变化函数 。开运算是调用cv2.morphologyEx(img, cv2.MORPH\_OPEN, kernel) 其中第一个参数： img 指需要开运算 的图 ；第二个参数 指的是开运算； 第三个参数 kernel 指开运算的内核 。

代码如下

import cv2  
import numpy as np  
img = cv2.imread("./image-20220425233418464.png",0)  
kernel = np.ones((3,3),np.uint8)  
erosion = cv2.erode(img,kernel)  
cv2.imshow('eroded image',erosion)  
cv2.waitKey(0)  
cv2.destroyAllWindows()  
dilation = cv2.dilate(img,kernel,iterations = 1)  
cv2.imshow('dilation',dilation)  
cv2.waitKey(0)  
aaa=cv2.morphologyEx(img, cv2.MORPH\_OPEN, kernel)  
cv2.imshow('aaa',aaa)  
cv2.waitKey(0)



开运算

#### 4 闭运算

闭运算：先 膨胀再腐蚀

代码如下：

import cv2  
import numpy as np  
img = cv2.imread("./image-20220425233418464.png",0)  
kernel = np.ones((3,3),np.uint8)  
erosion = cv2.erode(img,kernel)  
cv2.imshow('eroded image',erosion)  
cv2.waitKey(0)  
cv2.destroyAllWindows()  
dilation = cv2.dilate(img,kernel,iterations = 1)  
cv2.imshow('dilation',dilation)  
cv2.waitKey(0)  
aaa=cv2.morphologyEx(img, cv2.MORPH\_OPEN, kernel)  
cv2.imshow('aaa',aaa)  
cv2.waitKey(0)  
closing = cv2.morphologyEx(img, cv2.MORPH\_CLOSE, kernel)  
  
cv2.imshow('closing',closing)  
cv2.waitKey(0)



闭运算

## 实验四 数字图像的边缘检测实验

### 一、实验目的

1、掌握数字图像的空间域滤波原理

2、掌握数字图像的边缘检测原理及常用的边缘检测算子

3、掌握阈值分割及图像的二值化 。

### 二、实验主要仪器设备

(1)计算机

(2)python 软件

(3)典型的灰度、彩色图像文件 。

### 三、实验原理

（1）图像空间滤波原理：空间滤波是一种采用滤波处理的影像增强方法。其理论基础是空间卷积和空间相关。目的是改善影像质量，包括去除高频噪声与干扰，及影像边缘增强 、线性增强以及去模糊等。分为低通滤波（平滑化）、 高通滤波 （锐化 ）和带通滤波 。

（2）边缘检测原理：边缘检测是图像处理和计算机视觉中的基本问题，边缘检测的目的是标识数字图像中亮度变化明显的点。图像属性中的显著变化通常反映了属性的重要事件和变化。这些包括

（ i ）深度上的不连续、

（ ii ）表面方向不连续、

（iii ）物质属性变化和 iv场景照明变化。

边缘检测是图像处理和计算机视觉中，尤其是特征提取中的一个研究领域。

### 四、实验内容

（1） 读取图像

（2）使用 opencv python 对六种边缘检测器 进行图片边缘检测并对三 种算子得出的结果进行比较 。

### 五、实验报告要求

(1)熟悉边缘检测的原理

(2)列出实验程序及结果，并比较各边缘提取算子的特点

(3)写出心得、体会 。

### 六、预习要求

（1）了解边缘检测原理

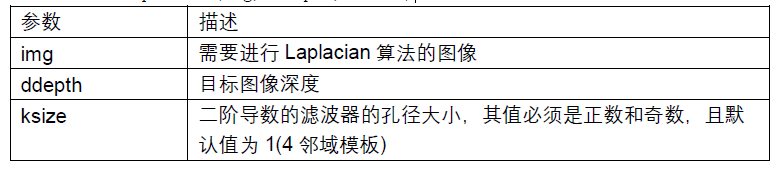
（2）熟悉边缘检测程序流程

### 七、实验

#### 1 LoG 检测器

实现LOG 算法要对图像先进行高斯滤波降噪处理，然后通过Laplacian 算法进行轮廓提取。

OpenCV 官方将Laplacian 算法封装到cv2.Laplacian()函数中，函数原型：result=cv2.Laplacian(img, ddepth, ksize)

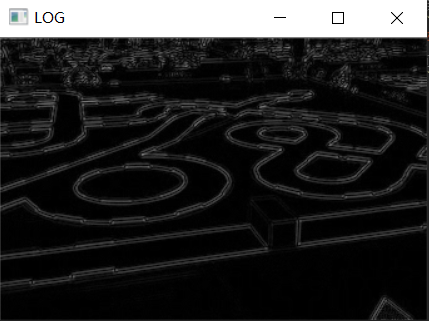


代码如下：

import cv2  
import numpy as np  
img = cv2.imread("./2.jpg",0)  
blur=cv2.GaussianBlur(img,(3,3),1,1)#核尺寸通过对图像的调节自行定义  
#调用Laplacian 算法的OpenCV 库函数进行图像轮廓提取  
result = cv2.Laplacian(blur,cv2.CV\_16S,ksize=1)  
LOG = cv2.convertScaleAbs(result)#得到LOG 算法处理结果  
cv2.imshow('LOG',LOG)  
cv2.waitKey(0)



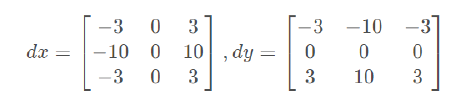
原图像



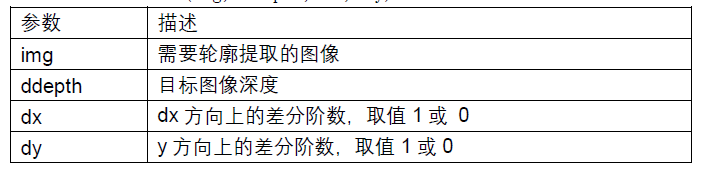
处理后的图像

#### 2 Scharr 算子

Scharr 算子又称为Scharr 滤波器，也是计算x 或y 方向上的图像差分，在OpenCV 中主要是配合Sobel 算子的运算而存在的。scharr 算子的卷积模板如下所示：



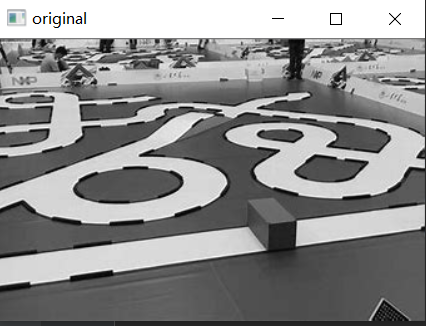
OpenCV 官方给出了明确的Scharr 算法的库函数，函数原型：result=cv2.Scharr(img, ddepth, dx, dy)



Scharr 算子的代码基本实现类似于Sobel 算子。

代码如下

import cv2  
import numpy as np  
img = cv2.imread("./2.jpg",0)  
scharrx = cv2.Scharr(img,cv2.CV\_64F,1,0)  
scharrx = cv2.convertScaleAbs(scharrx) # 转回uint8  
cv2.imshow("original",img)  
cv2.imshow("x",scharrx)  
cv2.waitKey()  
cv2.destroyAllWindows()



原图像



处理后的图像

#### 3 Canny 边缘检测器

Canny 边缘检测器是函数edge 中最强大的边缘检测器。方法总结如下：

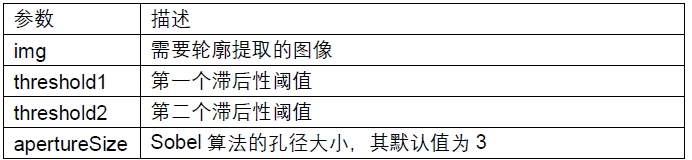
1.使用具有指定标准差δ 的一个高斯滤波器来平滑图像，以减少噪声。

2.在每个点处计算局部梯度和边缘方向。边缘点定义为梯度方向强度局部最大的点。

3.步骤2 中确定的边缘点产生梯度中的脊线。然后，算法沿这些脊线的顶部进行追踪，并将实际上不在脊线顶部的像素设置为零，从而在输出中给出一条细线，该过程称为非最大值抑制。然后使用称为滞后阈值处理的方法来对这些脊线像素进行阈值处理，这一处理方法使用两个阈值和其中。其值大于的脊线像素称为“强”边缘像素，值在 和之间的脊线像素称为“弱”边缘像素。

在OpenCV 中，Canny()库函数原型为：

Canny = cv2.Canny(img, threshold1, threshold2, apertureSize)



代码如下

import cv2 # opencv读取的格式是BGR  
import numpy as np  
import matplotlib.pyplot as plt # Matplotlib是RGB  
  
# 读图  
img = cv2.imread("2.jpg", cv2.IMREAD\_GRAYSCALE)  
  
# 采用Canny并调参，前者minval，后者maxval  
v1 = cv2.Canny(img, 80, 150)  
v2 = cv2.Canny(img, 50, 100)  
  
# 对比  
res = np.hstack((img, v1, v2))  
cv2.imshow("res",res)  
cv2.waitKey()  
cv2.destroyAllWindows()



## 实验五 基于图像分割的车牌定位识别

### 一、实验目的

1.掌握车牌阈值分割；

2.掌握基于形态学计算的图像分割；

3.掌握图像的二值化；

4.掌握基于像素投影的字符分割；

5.掌握字符识别原理。

### 二、实验主要仪器设备

(1)计算机；

(2)python 3.x

(3)需进行车牌识别的图片。

### 三、实验原理

（1）图像灰度化

灰度数字图像是每个像素只有一个采样颜色的图像。这类图像通常显示为从最暗黑色到最亮的白色的灰度，尽管理论上这个采样可以任何颜色的不同深浅，甚至可以是不同亮度上的不同颜色。灰度图像与黑白图像不同，在计算机图像领域中黑白图像只有黑白两种颜色，灰度图像在黑色与白色之间还有许多级的颜色深度。

（2）图像二值化

图像二值化就是将图像上的像素点的灰度值设置为 0或 255，也就是将整个图像呈现出明显的黑白效果。

（3）图像形态学运算

(见实验三 )

（4）阈值分割原理

阈值分割算法是图形分割中应用场景最多的算法之一。简单 地说，对灰度图像进行阈值分割就是先确定一个处于图像灰度取值范围内的阈值，然后将图像中各个像素的灰度值与这个阈值比较，并根据比较的结果将对应的像素划分为两类：像素灰度大于阈值的一类和像素值小于阈值的另一类，灰度值等于阈值的像素可以归入这两类之一。分割后的两类像素一般分属图像的两个不同区域，所以对像素根据阈值分类达到了区域分割的目的。

（5）字符分割原理

二值化后的图像 ,在没有字符的区域， y方向上像素灰度和为 0，在有字符的区域为灰度和非 0。

### 四、实验内容

（1）定位车牌区域 (可基于灰度阈值或形态学运算等

（2）车牌图像预处理 (灰度化、二值化

（3）字符分割 (可基于灰度垂直投影等

（4）字符识别 (可使用模版匹配等方法 )

### 五、实验报告

（1）说明车牌区域识别、车牌图像预处理、字符分割和识别的原理

（2）列出实验程序

（3）记录实验过程中每一步的图像

（4）心得和体会。

### 六、预习要求

（1）总结和掌握车牌识别的基本流程和常见方法

（2）熟悉 opencv图像基本处理的相关函数。

### 七、实验

import cv2  
import numpy as np  
from numpy.linalg import norm  
import sys  
import os  
import json  
  
SZ = 20 # 训练图片长宽  
MAX\_WIDTH = 1000 # 原始图片最大宽度  
Min\_Area = 2000 # 车牌区域允许最大面积  
PROVINCE\_START = 1000  
  
  
# 读取图片文件  
def imreadex(filename):  
 return cv2.imdecode(np.fromfile(filename, dtype=np.uint8), cv2.IMREAD\_COLOR)  
  
  
def point\_limit(point):  
 if point[0] < 0:  
 point[0] = 0  
 if point[1] < 0:  
 point[1] = 0  
  
  
# 根据设定的阈值和图片直方图，找出波峰，用于分隔字符  
def find\_waves(threshold, histogram):  
 up\_point = -1 # 上升点  
 is\_peak = False  
 if histogram[0] > threshold:  
 up\_point = 0  
 is\_peak = True  
 wave\_peaks = []  
 for i, x in enumerate(histogram):  
 if is\_peak and x < threshold:  
 if i - up\_point > 2:  
 is\_peak = False  
 wave\_peaks.append((up\_point, i))  
 elif not is\_peak and x >= threshold:  
 is\_peak = True  
 up\_point = i  
 if is\_peak and up\_point != -1 and i - up\_point > 4:  
 wave\_peaks.append((up\_point, i))  
 return wave\_peaks  
  
  
# 根据找出的波峰，分隔图片，从而得到逐个字符图片  
def seperate\_card(img, waves):  
 part\_cards = []  
 for wave in waves:  
 part\_cards.append(img[:, wave[0]:wave[1]])  
 return part\_cards  
  
  
# 来自opencv的sample，用于svm训练  
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=cv2.WARP\_INVERSE\_MAP | cv2.INTER\_LINEAR)  
 return img  
  
  
# 来自opencv的sample，用于svm训练  
def preprocess\_hog(digits):  
 samples = []  
 for img in digits:  
 gx = cv2.Sobel(img, cv2.CV\_32F, 1, 0)  
 gy = cv2.Sobel(img, cv2.CV\_32F, 0, 1)  
 mag, ang = cv2.cartToPolar(gx, gy)  
 bin\_n = 16  
 bin = np.int32(bin\_n \* ang / (2 \* np.pi))  
 bin\_cells = bin[:10, :10], bin[10:, :10], bin[:10, 10:], bin[10:, 10:]  
 mag\_cells = mag[:10, :10], mag[10:, :10], mag[:10, 10:], mag[10:, 10:]  
 hists = [np.bincount(b.ravel(), m.ravel(), bin\_n) for b, m in zip(bin\_cells, mag\_cells)]  
 hist = np.hstack(hists)  
  
 # transform to Hellinger kernel  
 eps = 1e-7  
 hist /= hist.sum() + eps  
 hist = np.sqrt(hist)  
 hist /= norm(hist) + eps  
  
 samples.append(hist)  
 return np.float32(samples)  
  
  
# 不能保证包括所有省份  
provinces = [  
 "zh\_cuan", "川",  
 "zh\_e", "鄂",  
 "zh\_gan", "赣",  
 "zh\_gan1", "甘",  
 "zh\_gui", "贵",  
 "zh\_gui1", "桂",  
 "zh\_hei", "黑",  
 "zh\_hu", "沪",  
 "zh\_ji", "冀",  
 "zh\_jin", "津",  
 "zh\_jing", "京",  
 "zh\_jl", "吉",  
 "zh\_liao", "辽",  
 "zh\_lu", "鲁",  
 "zh\_meng", "蒙",  
 "zh\_min", "闽",  
 "zh\_ning", "宁",  
 "zh\_qing", "靑",  
 "zh\_qiong", "琼",  
 "zh\_shan", "陕",  
 "zh\_su", "苏",  
 "zh\_sx", "晋",  
 "zh\_wan", "皖",  
 "zh\_xiang", "湘",  
 "zh\_xin", "新",  
 "zh\_yu", "豫",  
 "zh\_yu1", "渝",  
 "zh\_yue", "粤",  
 "zh\_yun", "云",  
 "zh\_zang", "藏",  
 "zh\_zhe", "浙"  
]  
  
  
class StatModel(object):  
 def load(self, fn):  
 self.model = self.model.load(fn)  
  
 def save(self, fn):  
 self.model.save(fn)  
  
  
class SVM(StatModel):  
 def \_\_init\_\_(self, C=1, gamma=0.5):  
 self.model = cv2.ml.SVM\_create()  
 self.model.setGamma(gamma)  
 self.model.setC(C)  
 self.model.setKernel(cv2.ml.SVM\_RBF)  
 self.model.setType(cv2.ml.SVM\_C\_SVC)  
  
 # 训练svm  
 def train(self, samples, responses):  
 self.model.train(samples, cv2.ml.ROW\_SAMPLE, responses)  
  
 # 字符识别  
 def predict(self, samples):  
 r = self.model.predict(samples)  
 return r[1].ravel()  
  
  
class CardPredictor:  
 def \_\_init\_\_(self):  
 # 车牌识别的部分参数保存在js中，便于根据图片分辨率做调整  
 f = open('./data/config.js')  
 j = json.load(f)  
 for c in j["config"]:  
 if c["open"]:  
 self.cfg = c.copy()  
 break  
 else:  
 raise RuntimeError('没有设置有效配置参数')  
  
 def \_\_del\_\_(self):  
 self.save\_traindata()  
  
 def train\_svm(self):  
 # 识别英文字母和数字  
 self.model = SVM(C=1, gamma=0.5)  
 # 识别中文  
 self.modelchinese = SVM(C=1, gamma=0.5)  
 if os.path.exists("./data/svm.dat"):  
 self.model.load("./data/svm.dat")  
 else:  
 chars\_train = []  
 chars\_label = []  
  
 for root, dirs, files in os.walk("train\\chars2"):  
 if len(os.path.basename(root)) > 1:  
 continue  
 root\_int = ord(os.path.basename(root))  
 for filename in files:  
 filepath = os.path.join(root, filename)  
 digit\_img = cv2.imread(filepath)  
 digit\_img = cv2.cvtColor(digit\_img, cv2.COLOR\_BGR2GRAY)  
 chars\_train.append(digit\_img)  
 # chars\_label.append(1)  
 chars\_label.append(root\_int)  
  
 chars\_train = list(map(deskew, chars\_train))  
 chars\_train = preprocess\_hog(chars\_train)  
 # chars\_train = chars\_train.reshape(-1, 20, 20).astype(np.float32)  
 chars\_label = np.array(chars\_label)  
 print(chars\_train.shape)  
 self.model.train(chars\_train, chars\_label)  
 if os.path.exists("./data/svmchinese.dat"):  
 self.modelchinese.load("./data/svmchinese.dat")  
 else:  
 chars\_train = []  
 chars\_label = []  
 for root, dirs, files in os.walk("train\\charsChinese"):  
 if not os.path.basename(root).startswith("zh\_"):  
 continue  
 pinyin = os.path.basename(root)  
 index = provinces.index(pinyin) + PROVINCE\_START + 1 # 1是拼音对应的汉字  
 for filename in files:  
 filepath = os.path.join(root, filename)  
 digit\_img = cv2.imread(filepath)  
 digit\_img = cv2.cvtColor(digit\_img, cv2.COLOR\_BGR2GRAY)  
 chars\_train.append(digit\_img)  
 # chars\_label.append(1)  
 chars\_label.append(index)  
 chars\_train = list(map(deskew, chars\_train))  
 chars\_train = preprocess\_hog(chars\_train)  
 # chars\_train = chars\_train.reshape(-1, 20, 20).astype(np.float32)  
 chars\_label = np.array(chars\_label)  
 print(chars\_train.shape)  
 self.modelchinese.train(chars\_train, chars\_label)  
  
 def save\_traindata(self):  
 if not os.path.exists("./data/svm.dat"):  
 self.model.save("./data/svm.dat")  
 if not os.path.exists("./data/svmchinese.dat"):  
 self.modelchinese.save("./data/svmchinese.dat")  
  
 def accurate\_place(self, card\_img\_hsv, limit1, limit2, color):  
 row\_num, col\_num = card\_img\_hsv.shape[:2]  
 xl = col\_num  
 xr = 0  
 yh = 0  
 yl = row\_num  
 # col\_num\_limit = self.cfg["col\_num\_limit"]  
 row\_num\_limit = self.cfg["row\_num\_limit"]  
 col\_num\_limit = col\_num \* 0.8 if color != "green" else col\_num \* 0.5 # 绿色有渐变  
 for i in range(row\_num):  
 count = 0  
 for j in range(col\_num):  
 H = card\_img\_hsv.item(i, j, 0)  
 S = card\_img\_hsv.item(i, j, 1)  
 V = card\_img\_hsv.item(i, j, 2)  
 if limit1 < H <= limit2 and 34 < S and 46 < V:  
 count += 1  
 if count > col\_num\_limit:  
 if yl > i:  
 yl = i  
 if yh < i:  
 yh = i  
 for j in range(col\_num):  
 count = 0  
 for i in range(row\_num):  
 H = card\_img\_hsv.item(i, j, 0)  
 S = card\_img\_hsv.item(i, j, 1)  
 V = card\_img\_hsv.item(i, j, 2)  
 if limit1 < H <= limit2 and 34 < S and 46 < V:  
 count += 1  
 if count > row\_num - row\_num\_limit:  
 if xl > j:  
 xl = j  
 if xr < j:  
 xr = j  
 return xl, xr, yh, yl  
  
 def predict(self, car\_pic):  
 if type(car\_pic) == type(""):  
 img = imreadex(car\_pic)  
 else:  
 img = car\_pic  
 pic\_hight, pic\_width = img.shape[:2]  
  
 if pic\_width > MAX\_WIDTH:  
 resize\_rate = MAX\_WIDTH / pic\_width  
 img = cv2.resize(img, (MAX\_WIDTH, int(pic\_hight \* resize\_rate)), interpolation=cv2.INTER\_AREA)  
  
 blur = self.cfg["blur"]  
 # 高斯去噪  
 if blur > 0:  
 img = cv2.GaussianBlur(img, (blur, blur), 0) # 图片分辨率调整  
 oldimg = img  
 img = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  
 # equ = cv2.equalizeHist(img)  
 # img = np.hstack((img, equ))  
 # 去掉图像中不会是车牌的区域  
 kernel = np.ones((20, 20), np.uint8)  
 img\_opening = cv2.morphologyEx(img, cv2.MORPH\_OPEN, kernel)  
 img\_opening = cv2.addWeighted(img, 1, img\_opening, -1, 0);  
  
 # 找到图像边缘  
 ret, img\_thresh = cv2.threshold(img\_opening, 0, 255, cv2.THRESH\_BINARY + cv2.THRESH\_OTSU)  
 img\_edge = cv2.Canny(img\_thresh, 100, 200)  
 # 使用开运算和闭运算让图像边缘成为一个整体  
 kernel = np.ones((self.cfg["morphologyr"], self.cfg["morphologyc"]), np.uint8)  
 img\_edge1 = cv2.morphologyEx(img\_edge, cv2.MORPH\_CLOSE, kernel)  
 img\_edge2 = cv2.morphologyEx(img\_edge1, cv2.MORPH\_OPEN, kernel)  
  
 # 查找图像边缘整体形成的矩形区域，可能有很多，车牌就在其中一个矩形区域中  
 try:  
 contours, hierarchy = cv2.findContours(img\_edge2, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)  
 except ValueError:  
 image, contours, hierarchy = cv2.findContours(img\_edge2, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)  
 contours = [cnt for cnt in contours if cv2.contourArea(cnt) > Min\_Area]  
 print('len(contours)', len(contours))  
 # 一一排除不是车牌的矩形区域  
 car\_contours = []  
 for cnt in contours:  
 rect = cv2.minAreaRect(cnt)  
 area\_width, area\_height = rect[1]  
 if area\_width < area\_height:  
 area\_width, area\_height = area\_height, area\_width  
 wh\_ratio = area\_width / area\_height  
 # print(wh\_ratio)  
 # 要求矩形区域长宽比在2到5.5之间，2到5.5是车牌的长宽比，其余的矩形排除  
 if wh\_ratio > 2 and wh\_ratio < 5.5:  
 car\_contours.append(rect)  
 box = cv2.boxPoints(rect)  
 box = np.int0(box)  
 # oldimg = cv2.drawContours(oldimg, [box], 0, (0, 0, 255), 2)  
 # cv2.imshow("edge4", oldimg)  
 # print(rect)  
  
 print(len(car\_contours))  
  
 print("精确定位")  
 card\_imgs = []  
 # 矩形区域可能是倾斜的矩形，需要矫正，以便使用颜色定位  
 for rect in car\_contours:  
 if rect[2] > -1 and rect[2] < 1: # 创造角度，使得左、高、右、低拿到正确的值  
 angle = 1  
 else:  
 angle = rect[2]  
 rect = (rect[0], (rect[1][0] + 5, rect[1][1] + 5), angle) # 扩大范围，避免车牌边缘被排除  
  
 box = cv2.boxPoints(rect)  
 heigth\_point = right\_point = [0, 0]  
 left\_point = low\_point = [pic\_width, pic\_hight]  
 for point in box:  
 if left\_point[0] > point[0]:  
 left\_point = point  
 if low\_point[1] > point[1]:  
 low\_point = point  
 if heigth\_point[1] < point[1]:  
 heigth\_point = point  
 if right\_point[0] < point[0]:  
 right\_point = point  
  
 if left\_point[1] <= right\_point[1]: # 正角度  
 new\_right\_point = [right\_point[0], heigth\_point[1]]  
 pts2 = np.float32([left\_point, heigth\_point, new\_right\_point]) # 字符只是高度需要改变  
 pts1 = np.float32([left\_point, heigth\_point, right\_point])  
 M = cv2.getAffineTransform(pts1, pts2)  
 dst = cv2.warpAffine(oldimg, M, (pic\_width, pic\_hight))  
 point\_limit(new\_right\_point)  
 point\_limit(heigth\_point)  
 point\_limit(left\_point)  
 card\_img = dst[int(left\_point[1]):int(heigth\_point[1]), int(left\_point[0]):int(new\_right\_point[0])]  
 card\_imgs.append(card\_img)  
 # cv2.imshow("card", card\_img)  
 # cv2.waitKey(0)  
 elif left\_point[1] > right\_point[1]: # 负角度  
  
 new\_left\_point = [left\_point[0], heigth\_point[1]]  
 pts2 = np.float32([new\_left\_point, heigth\_point, right\_point]) # 字符只是高度需要改变  
 pts1 = np.float32([left\_point, heigth\_point, right\_point])  
 M = cv2.getAffineTransform(pts1, pts2)  
 dst = cv2.warpAffine(oldimg, M, (pic\_width, pic\_hight))  
 point\_limit(right\_point)  
 point\_limit(heigth\_point)  
 point\_limit(new\_left\_point)  
 card\_img = dst[int(right\_point[1]):int(heigth\_point[1]), int(new\_left\_point[0]):int(right\_point[0])]  
 card\_imgs.append(card\_img)  
 # cv2.imshow("card", card\_img)  
 # cv2.waitKey(0)  
 # 开始使用颜色定位，排除不是车牌的矩形，目前只识别蓝、绿、黄车牌  
 colors = []  
 for card\_index, card\_img in enumerate(card\_imgs):  
 green = yello = blue = black = white = 0  
 card\_img\_hsv = cv2.cvtColor(card\_img, cv2.COLOR\_BGR2HSV)  
 # 有转换失败的可能，原因来自于上面矫正矩形出错  
 if card\_img\_hsv is None:  
 continue  
 row\_num, col\_num = card\_img\_hsv.shape[:2]  
 card\_img\_count = row\_num \* col\_num  
  
 for i in range(row\_num):  
 for j in range(col\_num):  
 H = card\_img\_hsv.item(i, j, 0)  
 S = card\_img\_hsv.item(i, j, 1)  
 V = card\_img\_hsv.item(i, j, 2)  
 if 11 < H <= 34 and S > 34: # 图片分辨率调整  
 yello += 1  
 elif 35 < H <= 99 and S > 34: # 图片分辨率调整  
 green += 1  
 elif 99 < H <= 124 and S > 34: # 图片分辨率调整  
 blue += 1  
  
 if 0 < H < 180 and 0 < S < 255 and 0 < V < 46:  
 black += 1  
 elif 0 < H < 180 and 0 < S < 43 and 221 < V < 225:  
 white += 1  
 color = "no"  
  
 limit1 = limit2 = 0  
 if yello \* 2 >= card\_img\_count:  
 color = "yello"  
 limit1 = 11  
 limit2 = 34 # 有的图片有色偏偏绿  
 elif green \* 2 >= card\_img\_count:  
 color = "green"  
 limit1 = 35  
 limit2 = 99  
 elif blue \* 2 >= card\_img\_count:  
 color = "blue"  
 limit1 = 100  
 limit2 = 124 # 有的图片有色偏偏紫  
 elif black + white >= card\_img\_count \* 0.7: # TODO  
 color = "bw"  
 print(color)  
 colors.append(color)  
 print(blue, green, yello, black, white, card\_img\_count)  
 # cv2.imshow("color", card\_img)  
 # cv2.waitKey(0)  
 if limit1 == 0:  
 continue  
 # 以上为确定车牌颜色  
 # 以下为根据车牌颜色再定位，缩小边缘非车牌边界  
 xl, xr, yh, yl = self.accurate\_place(card\_img\_hsv, limit1, limit2, color)  
 if yl == yh and xl == xr:  
 continue  
 need\_accurate = False  
 if yl >= yh:  
 yl = 0  
 yh = row\_num  
 need\_accurate = True  
 if xl >= xr:  
 xl = 0  
 xr = col\_num  
 need\_accurate = True  
 card\_imgs[card\_index] = card\_img[yl:yh, xl:xr] if color != "green" or yl < (yh - yl) // 4 else card\_img[  
 yl - (  
 yh - yl) // 4:yh,  
 xl:xr]  
 if need\_accurate: # 可能x或y方向未缩小，需要再试一次  
 card\_img = card\_imgs[card\_index]  
 card\_img\_hsv = cv2.cvtColor(card\_img, cv2.COLOR\_BGR2HSV)  
 xl, xr, yh, yl = self.accurate\_place(card\_img\_hsv, limit1, limit2, color)  
 if yl == yh and xl == xr:  
 continue  
 if yl >= yh:  
 yl = 0  
 yh = row\_num  
 if xl >= xr:  
 xl = 0  
 xr = col\_num  
 card\_imgs[card\_index] = card\_img[yl:yh, xl:xr] if color != "green" or yl < (yh - yl) // 4 else card\_img[  
 yl - (  
 yh - yl) // 4:yh,  
 xl:xr]  
 # 以上为车牌定位  
 # 以下为识别车牌中的字符  
 predict\_result = []  
 roi = None  
 card\_color = None  
 for i, color in enumerate(colors):  
 if color in ("blue", "yello", "green"):  
 card\_img = card\_imgs[i]  
 gray\_img = cv2.cvtColor(card\_img, cv2.COLOR\_BGR2GRAY)  
 # 黄、绿车牌字符比背景暗、与蓝车牌刚好相反，所以黄、绿车牌需要反向  
 if color == "green" or color == "yello":  
 gray\_img = cv2.bitwise\_not(gray\_img)  
 ret, gray\_img = cv2.threshold(gray\_img, 0, 255, cv2.THRESH\_BINARY + cv2.THRESH\_OTSU)  
 # 查找水平直方图波峰  
 x\_histogram = np.sum(gray\_img, axis=1)  
 x\_min = np.min(x\_histogram)  
 x\_average = np.sum(x\_histogram) / x\_histogram.shape[0]  
 x\_threshold = (x\_min + x\_average) / 2  
 wave\_peaks = find\_waves(x\_threshold, x\_histogram)  
 if len(wave\_peaks) == 0:  
 print("peak less 0:")  
 continue  
 # 认为水平方向，最大的波峰为车牌区域  
 wave = max(wave\_peaks, key=lambda x: x[1] - x[0])  
 gray\_img = gray\_img[wave[0]:wave[1]]  
 # 查找垂直直方图波峰  
 row\_num, col\_num = gray\_img.shape[:2]  
 # 去掉车牌上下边缘1个像素，避免白边影响阈值判断  
 gray\_img = gray\_img[1:row\_num - 1]  
 y\_histogram = np.sum(gray\_img, axis=0)  
 y\_min = np.min(y\_histogram)  
 y\_average = np.sum(y\_histogram) / y\_histogram.shape[0]  
 y\_threshold = (y\_min + y\_average) / 5 # U和0要求阈值偏小，否则U和0会被分成两半  
  
 wave\_peaks = find\_waves(y\_threshold, y\_histogram)  
  
 # for wave in wave\_peaks:  
 # cv2.line(card\_img, pt1=(wave[0], 5), pt2=(wave[1], 5), color=(0, 0, 255), thickness=2)  
 # 车牌字符数应大于6  
 if len(wave\_peaks) <= 6:  
 print("peak less 1:", len(wave\_peaks))  
 continue  
  
 wave = max(wave\_peaks, key=lambda x: x[1] - x[0])  
 max\_wave\_dis = wave[1] - wave[0]  
 # 判断是否是左侧车牌边缘  
 if wave\_peaks[0][1] - wave\_peaks[0][0] < max\_wave\_dis / 3 and wave\_peaks[0][0] == 0:  
 wave\_peaks.pop(0)  
  
 # 组合分离汉字  
 cur\_dis = 0  
 for i, wave in enumerate(wave\_peaks):  
 if wave[1] - wave[0] + cur\_dis > max\_wave\_dis \* 0.6:  
 break  
 else:  
 cur\_dis += wave[1] - wave[0]  
 if i > 0:  
 wave = (wave\_peaks[0][0], wave\_peaks[i][1])  
 wave\_peaks = wave\_peaks[i + 1:]  
 wave\_peaks.insert(0, wave)  
  
 # 去除车牌上的分隔点  
 point = wave\_peaks[2]  
 if point[1] - point[0] < max\_wave\_dis / 3:  
 point\_img = gray\_img[:, point[0]:point[1]]  
 if np.mean(point\_img) < 255 / 5:  
 wave\_peaks.pop(2)  
  
 if len(wave\_peaks) <= 6:  
 print("peak less 2:", len(wave\_peaks))  
 continue  
 part\_cards = seperate\_card(gray\_img, wave\_peaks)  
 for i, part\_card in enumerate(part\_cards):  
 # 可能是固定车牌的铆钉  
 if np.mean(part\_card) < 255 / 5:  
 print("a point")  
 continue  
 part\_card\_old = part\_card  
 w = abs(part\_card.shape[1] - SZ) // 2  
  
 part\_card = cv2.copyMakeBorder(part\_card, 0, 0, w, w, cv2.BORDER\_CONSTANT, value=[0, 0, 0])  
 part\_card = cv2.resize(part\_card, (SZ, SZ), interpolation=cv2.INTER\_AREA)  
  
 # part\_card = deskew(part\_card)  
 part\_card = preprocess\_hog([part\_card])  
 if i == 0:  
 resp = self.modelchinese.predict(part\_card)  
 charactor = provinces[int(resp[0]) - PROVINCE\_START]  
 else:  
 resp = self.model.predict(part\_card)  
 charactor = chr(resp[0])  
 # 判断最后一个数是否是车牌边缘，假设车牌边缘被认为是1  
 if charactor == "1" and i == len(part\_cards) - 1:  
 if part\_card\_old.shape[0] / part\_card\_old.shape[1] >= 7: # 1太细，认为是边缘  
 continue  
 predict\_result.append(charactor)  
 roi = card\_img  
 card\_color = color  
 break  
  
 return predict\_result, roi, card\_color # 识别到的字符、定位的车牌图像、车牌颜色  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 c = CardPredictor()  
 c.train\_svm()  
 r, roi, color = c.predict('./1.jpg')  
 print(r)

识别结果如下

C:\ProgramData\Anaconda3\python.exe F:/Mirror/Learning/研一/数字图像处理/code/LicensePlaterecognize-master/predict.py  
len(contours) 5  
2  
精确定位  
blue  
3060 147 0 0 4 3213  
no  
645 292 75 83 2 12455  
['鲁', 'R', 'R', 'Y', '9', '0', '2']  
Process finished with exit code 0

