信息内容安全实验课第六周

Python数据分析

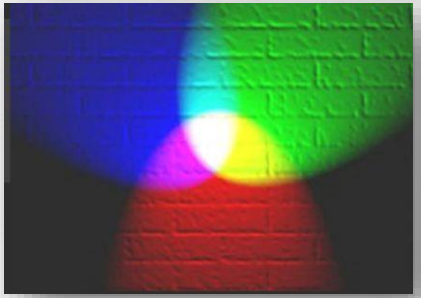
chapter 6 使用python进行图像处理

图像的表示

通常情况下，计算机中图像是一个三维数组，维度分别是高度、宽度和像素RGB值

RGB色彩模式，即每个像素点的颜色由红(R)、绿(G)、蓝(B)组成

RGB三个颜色通道的变化和叠加得到各种颜色，其中



- R 红色，取值范围，0-255
- G 绿色，取值范围，0-255
- B 蓝色，取值范围，0-255

RGB形成的颜色包括了人类视力所能感知的所有颜色。

Python中对数组操作最常用的库为 ：

NumPy(更多关注数据的结构) 和 pandas (更多关注数据的内容)

NumPy

NumPy是一个开源的Python科学计算基础库，包含：

- 一个强大的N维数组对象 ndarray
- 广播功能函数
- 整合C/C++/Fortran代码的工具
- 线性代数、傅里叶变换、随机数生成等功能

NumPy是SciPy、Pandas等数据处理或科学计算库的基础

http://www.numpy.org/

Anaconda中以整合了NumPy、SciPy等数据处理的常用库

```
import numpy as np
```

as 后的内容指引入模块的别名

NumPy

N维数组对象：ndarray

Python已有列表类型，为什么需要一个数组对象(类型)？

- 数组对象可以去掉**元素间运算**所需的循环，使一维向量更像单个数据
- 设置专门的数组对象，经过优化，可以提升这类应用的运算速度

例：计算 A^2+B^3，其中，A和B是一维数组

```
def pySum():
    a = [0, 1, 2, 3, 4]
    b = [9, 8, 7, 6, 5]
    c = []

    for i in range(len(a)):
        c.append(a[i]**2 + b[i]**3)

    return c

print(pySum())
```

```
import numpy as np

def npSum():
    a = np.array([0, 1, 2, 3, 4])
    b = np.array([9, 8, 7, 6, 5])

    c = a**2 + b**3

    return c

print(npSum())
```

NumPy

ndarray 数组**一般**要求所有元素类型相同（同质），数组下标从0开始

由两部分构成：1、实际的数据

2、描述这些数据的元数据（数据维度、数据类型等）

ndarray对象的属性：

属性	说明
.ndim	秩，即轴的数量或维度的数量
.shape	ndarray对象的尺度，对于矩阵，n行m列
.size	ndarray对象元素的个数，相当于.shape中n*m的
.dtype	ndarray对象的元素类型
.itemsize	ndarray对象中每个元素的大小，以字节为单位

ndarray数组的创建方法：

1、从Python中的列表、元组等类型创建ndarray数组

x = np.array(list/tuple, dtype=np.float32)

当np.array()不指定dtype时，NumPy将根据数据情况关联一个dtype类型

```
In [32]: x = np.array([0, 1, 2, 3])

In [33]: print(x)
[0 1 2 3]

In [34]: x = np.array((4, 5, 6, 7))

In [35]: print(x)
[4 5 6 7]

In [36]: x = np.array([ [1, 2], [9, 8], (0.1, 0.2)])

In [37]: print(x)
[[ 1.   2. ]
 [ 9.   8. ]
 [ 0.1  0.2]]
```

从列表类型创建

从元组类型创建

从列表和元组混合类型创建

NumPy

ndarray数组的创建方法：

2、使用NumPy中函数创建ndarray数组，如：arange, ones, zeros等

函数	说明
np.arange(n)	类似range()函数，返回ndarray类型，元素从0到n-1
np.ones(shape)	根据shape生成一个全1数组，shape是元组类型
np.zeros(shape)	根据shape生成一个全0数组，shape是元组类型
np.full(shape,val)	根据shape生成一个数组，每个元素值都是val
np.eye(n)	创建一个正方的n*n单位矩阵，对角线为1，其余为0
np.ones_like(a)	根据数组a的形状生成一个全1数组
np.zeros_like(a)	根据数组a的形状生成一个全0数组
np.full_like(a,val)	根据数组a的形状生成一个数组，每个元素值都是val

ndarray数组的创建方法：

3、使用NumPy中其他函数创建ndarray数组

函数	说明
np.linspace()	根据起止数据等间距地填充数据，形成数组
np.concatenate()	将两个或多个数组合并成一个新的数组

```
In [51]: a = np.linspace(1, 10, 4)

In [52]: a
Out[52]: array([  1.,   4.,   7.,  10.])

In [53]: b = np.linspace(1, 10, 4, endpoint=False)

In [54]: b
Out[54]: array([ 1.  ,  3.25,  5.5 ,  7.75])


In [56]: c = np.concatenate((a, b))

In [57]: c
Out[57]: array([  1.  ,   4.  ,   7.  ,  10.  ,   1.  ,   3.25,   5.5 ,   7.75])
```

ndarray数组的变换

对于创建后的ndarray数组，可以对其进行维度变换和元素类型变换

元素类型变换：a = np.ones((2,3,4), dtype=np.int32)	
函数	**说明**
.reshape(shape)	不改变数组元素，返回一个shape形状的数组，原数组不变
.resize(shape)	与.reshape()功能一致，但修改原数组
.swapaxes(ax1,ax2)	将数组n个维度中两个维度进行调换
.flatten()	对数组进行降维，返回折叠后的一维数组，原数组不变

NumPy

数组的索引和切片

索引：获取数组中特定位置元素的过程

切片：获取数组元素子集的过程

```
In [131]: a = np.array([9, 8, 7, 6, 5])

In [132]: a[2]
Out[132]: 7

In [133]: a[ 1 : 4 : 2 ]
Out[133]: array([8, 6])
```

起始编号：终止编号(不含)：步长，3元素冒号分割

编号0开始从左递增，或-1开始从右递减

数组的索引和切片

多维数组的切片：

```
In [146]: a = np.arange(24).reshape((2,3,4))

In [147]: a
Out[147]:
array([[[ 0,  1,  2,  3],
        [ 4,  5,  6,  7],
        [ 8,  9, 10, 11]],

       [[12, 13, 14, 15],
        [16, 17, 18, 19],
        [20, 21, 22, 23]]])
```

```
In [158]: a[:, 1, -3]
Out[158]: array([ 5, 17])
```

选取一个维度用：

```
In [159]: a[:, 1:3, :]
Out[159]:
array([[[ 4,  5,  6,  7],
        [ 8,  9, 10, 11]],

       [[16, 17, 18, 19],
        [20, 21, 22, 23]]])
```

每个维度切片方法与一维数组相同

```
In [160]: a[:, :, ::2]
Out[160]:
array([[[ 0,  2],
        [ 4,  6],
        [ 8, 10]],

       [[12, 14],
        [16, 18],
        [20, 22]]])
```

每个维度可以使用步长跳跃切片

ndarray数组的运算

函数	说明
np.abs(x) np.fabs(x)	计算数组各元素的绝对值
np.sqrt(x)	计算数组各元素的平方根
np.square(x)	计算数组各元素的平方
np.log(x) np.log10(x) np.log2(x)	计算数组各元素的自然对数、 10底对数和2底对数
np.ceil(x) np.floor(x)	计算数组各元素的ceiling值（向上取整） 或 floor值（向下取整）
np.rint(x)	计算数组各元素的四舍五入值
np.modf(x)	将数组各元素的小数和整数部分以两个独立数组形式返回
np.cos(x) np.cosh(x) np.sin(x) np.sinh(x) np.tan(x) np.tanh(x)	计算数组各元素的普通型和双曲型三角函数
np.exp(x)	计算数组各元素的指数值
np.sign(x)	计算数组各元素的符号值，1(+), 0, -1(-)

ndarray数组的运算

数组与标量之间的运算作用于数组的每一个元素

函数	说明
+ - * / **	两个数组各元素进行对应运算
np.maximum(x,y) np.fmax() np.minimum(x,y) np.fmin()	元素级的最大值/最小值计算
np.mod(x,y)	元素级的模运算
np.copysign(x,y)	将数组y中各元素值的符号赋值给数组x对应元素

NumPy的随机函数

函数	说明
rand(d0,d1,..,dn)	根据d0-dn创建随机数数组，浮点数，[0,1)，均匀分布
randn(d0,d1,..,dn)	根据d0-dn创建随机数数组，标准正态分布
randint(low[,high,shape])	根据shape创建随机整数或整数数组，范围是[low, high)
seed(s)	随机数种子，s是给定的种子值
shuffle(a)	根据数组a的第1轴进行随排列，改变数组x
permutation(a)	根据数组a的第1轴产生一个新的乱序数组，不改变数组x
从一维数组a中以概率p抽取元素，形成size形状新数组	replace表示是否可以重用元素，默认为False
uniform(low,high,size)	产生具有均匀分布的数组,low起始值,high结束值,size形状
normal(loc,scale,size)	产生具有正态分布的数组,loc均值,scale标准差,size形状
poisson(lam,size)	产生具有泊松分布的数组,lam随机事件发生率,size形状

上述随机函数在numpy.random中

NumPy的统计函数

函数	说明
sum(a, axis=None)	根据给定轴axis计算数组a相关元素之和，axis整数或元组
mean(a, axis=None)	根据给定轴axis计算数组a相关元素的期望，axis整数或元组
average(a,axis=None,weights=None)	根据给定轴axis计算数组a相关元素的加权平均值
std(a, axis=None)	根据给定轴axis计算数组a相关元素的标准差
min(a) max(a)	计算数组a中元素的最小值、最大值
argmin(a) argmax(a)	计算数组a中元素最小值、最大值的降一维后下标
unravel_index(index, shape)	根据shape将一维下标index转换成多维下标
ptp(a)	计算数组a中元素最大值与最小值的差
median(a)	计算数组a中元素的中位数（中值）
np.gradient(f)	计算数组f中元素的梯度，当f为多维时，返回每个维度梯度

图像的表示

Python中图像处理最常用的库是PIL库（Python Image Library）

PILLOW官方文档：https://pillow.readthedocs.io/en/latest/reference/Image.html

中文博客：http://blog.csdn.net/column/details/pythonpil.html?&page=1

PIL库是一个具有强大图像处理能力的第三方库

在anaconda下的安装方法： conda install pillow

图像的表示

例1、将彩色RGB图片变为黑白（灰度）图片

PIL中，灰度表示模式为L模式，它的每个像素用8个bit表示，0表示黑，255表示白，其他数字表示不同的灰度。

```
from PIL import Image
im = Image.open('1.jpg')
im_1 = im.convert("L")
im_1.show()
im_1.save('output1.jpg' )
```

灰度转换公式：L = R * 299/1000 + G * 587/1000+ B * 114/1000

图像的表示

例2，将彩色RGB图片变为“底片”模式

PIL中，当需要更个性化的图片时，可配合使用numpy,对图像中的数据进行操作

将原图以灰度打开后，取反码（b=255-a）

```
from PIL import Image
import numpy as np
a = np.array(Image.open("1.jpg").convert('L'))
b = 255 - a
im = Image.fromarray(b.astype('uint8'))
im.save("output3.jpg")
```

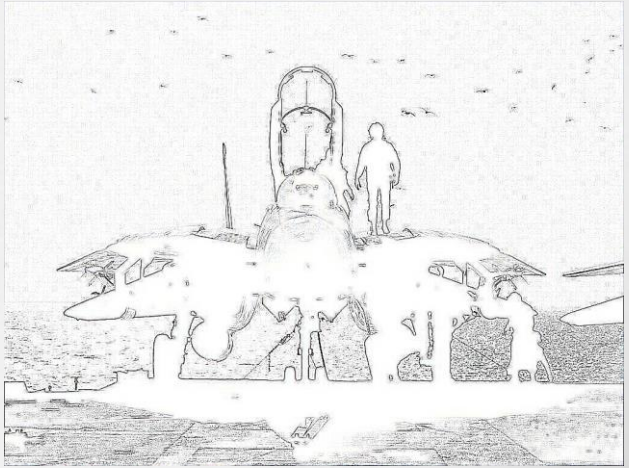
图像的表示

PIL的ImageFilter模块

ImageFilter模块提供了滤波器相关定义；这些滤波器主要用于Image类的filter()方法。

例如，提取图片的轮廓信息：使用ImageFilter.CONTOUR

```
from PIL import Image
from PIL import ImageFilter
a = Image.open('5.jpg')
b =
a.filter(ImageFilter.CONTOUR)
b.save("output5.jpg")
```



OpenCV

OpenCV是计算机视觉领域应用最广泛的开源工具包，基于C/C++，支持Linux/Windows/MacOS/Android/iOS，并提供了Python，Matlab和Java等语言的接口，因为其丰富的接口，优秀的性能和商业友好的使用许可，不管是学术界还是业界中都非常受欢迎。OpenCV旨在提供一个用于计算机视觉的科研和商业应用的高性能通用库。

安装（两种方法二选一）

- windows命令行
  pip install *.whl
  下载相应版本http://www.lfd.uci.edu/~gohlke/pythonlibs/#opencv

- Anaconda命令行
  conda install --channel https://conda.anaconda.org/menpo opencv3

主要模块

根据功能和需求的不同，OpenCV中的函数接口大体可以分为如下部分：
core：核心模块，主要包含了OpenCV中最基本的结构（矩阵，点线和形状等），以及相关的基础运算/操作。
imgproc：图像处理模块，包含和图像相关的基础功能（滤波，梯度，改变大小等），以及一些衍生的高级功能（图像分割，直方图，形态分析和边缘/直线提取等）。
highgui：提供了用户界面和文件读取的基本函数，比如图像显示窗口的生成和控制，图像/视频文件的IO等。

针对视频和一些特别的视觉应用，OpenCV也提供了强劲的支持：
video：用于视频分析的常用功能，比如光流法（Optical Flow）和目标跟踪等。
calib3d：三维重建，立体视觉和相机标定等的相关功能。
features2d：二维特征相关的功能，主要是一些不受专利保护的，商业友好的特征点检测和匹配等功能，比如ORB特征。
object：目标检测模块，包含级联分类和Latent SVM
ml：机器学习算法模块，包含一些视觉中最常用的传统机器学习算法。
flann：最近邻算法库，Fast Library for Approximate Nearest Neighbors，用于在多维空间进行聚类和检索，经常和关键点匹配搭配使用。
gpu：包含了一些gpu加速的接口，底层的加速是CUDA实现。
photo：计算摄像学（Computational Photography）相关的接口，当然这只是个名字，其实只有图像修复和降噪而已。
stitching：图像拼接模块，有了它可以自己生成全景照片。
nonfree：受到专利保护的一些算法，其实就是SIFT和SURF。
contrib：一些实验性质的算法，考虑在未来版本中加入的。
legacy：字面是遗产，意思就是废弃的一些接口，保留是考虑到向下兼容。
ocl：利用OpenCL并行加速的一些接口。
superres：超分辨率模块，其实就是BTV-L1（Biliteral Total Variation – L1 regularization）算法
viz：基础的3D渲染模块，其实底层就是著名的3D工具包VTK（Visualization Toolkit）。

图片读、写和显示操作

```
import numpy as np
import cv2 as cv

#读图片（有多种模式）
# Load an color image in grayscale
img = cv.imread('lena.jpg',0)

#显示图片
cv.imshow('image',img)  #第一个参数定义窗口名
cv.waitKey(0)       #无限制的等待用户的按键
cv.destroyAllWindows()

#写
cv.imwrite('graylena.png',img)
```

cv.IMREAD_COLOR	1 彩色
cv.IMREAD_GRAYSCALE	0 灰度图像
cv.IMREAD_UNCHANGED	-1 原始图像

原图

灰度图

图片属性

```
import cv2 as cv

img = cv.imread('lena.jpg’)

#图片的相关属性
print(img.shape)   #高度×宽度×通道数（灰度图只返回前两项）
print(img.size)    #像素总数
print(img.dtype)   #图像数据类型
```

结果：

(512, 512, 3)
786432
uint8

注意：最常见的RGB通道中，第一个元素就是红色（Red）的值，第二个元素是绿色（Green）的值，第三个元素是蓝色（Blue），最终得到的图像如6-1a所示。RGB是最常见的情况，然而在OpenCV中，默认的图像的表示确实反过来的，也就是BGR。在OpenCV刚开始研发的年代，BGR是相机设备厂商的主流表示方法，虽然后来RGB成了主流和默认，但是这个底层的顺序却保留下来了，事实上Windows下的最常见格式之一bmp，底层字节的存储顺序还是BGR。

图片缩放

resize

```
cv2.resize(src, dsize, dst, fx=0, fy=0, interpolation=INTER_LINEAR )
```

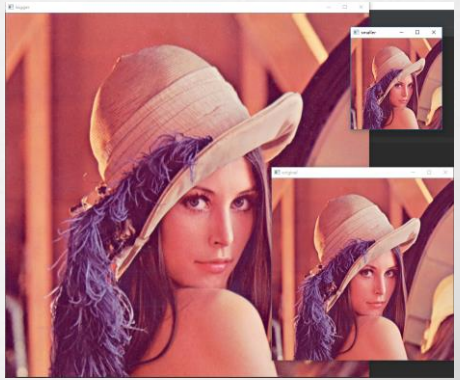
参数的含义：

1. InputArray src -原图像
2. OutputArray dst -输出图像
3. Size dsize -目标图像的大小
4. double fx=0 -在x轴上的缩放比例
5. double fy=0 -在y轴上的缩放比例
6. int interpolation -插值方式，INTER_NN -最近邻插值 、INTER_LINEAR -双线性插值（缺省使用）、INTER_AREA -使用象素关系重采样（当图像缩小时候，该方法可以避免波纹出现。当图像放大时，类似于 INTER_NN 方法）、INTER_CUBIC -立方插值。

说明：dsize与fx和fy必须不能同时为零

图片缩放示例

```
import numpy as np
import cv2 as cv

img = cv.imread('lena.jpg')
smaller = cv.resize(img,None,fx=0.5, fy=0.5, interpolation = cv.INTER_CUBIC)
#OR
height, width = img.shape[:2]
bigger = cv.resize(img,(2*width, 2*height), interpolation = cv.INTER_CUBIC)

cv.imshow('original',img)
cv.imshow('smaller',smaller)
cv.imshow('bigger',bigger)

cv.waitKey(0)
cv.destroyAllWindows()
```



仿射变换

原理

一个任意的仿射变换都能表示为 *乘以一个矩阵* (线性变换) 接着再 *加上一个向量* (平移).

- 旋转 (线性变换)
- 平移 (向量加)
- 缩放操作 (线性变换)

我们通常使用 2 x 3 矩阵来表示仿射变换.其中左边的2×2子矩阵是线性变换矩阵，右边的2×1的两项是平移项：

$$M = [A \quad B] = \begin{bmatrix} a_{00} & a_{01} & b_0 \\ a_{10} & a_{11} & b_1 \end{bmatrix} \qquad A = \begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix}, B = \begin{bmatrix} b_0 \\ b_1 \end{bmatrix}$$

对于图像上的任一位置(x,y)，仿射变换执行的是如下的操作：

$$T_{affine} = A \begin{bmatrix} x \\ y \end{bmatrix} + B = M \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

需要注意的是，对于图像而言，宽度方向是x，高度方向是y，坐标的顺序和图像像素对应下标一致。所以原点的位置不是左下角而是左上角，y的方向也不是向上，而是向下。

仿射变换

平移、旋转变换

- **平移**

将每一点移到到（x+t ， y+t)，变换矩阵为

$$M = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix}$$

- **旋转变换**

目标图形围绕原点顺时针旋转θ弧度，线性变换矩阵为

$$\begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}$$

目标图形以(x,y)为轴心顺时针旋转θ弧度，变换矩阵为

$$M = \begin{bmatrix} cos\Theta & -sin\Theta & x - x \cdot cos\Theta + y \cdot sin\Theta \\ sin\Theta & cos\Theta & y - x \cdot sin\Theta - y \cdot cos\Theta \end{bmatrix}$$

相当于两次平移与一次原点旋转变换的复合，即先将轴心（x,y)移到到原点，然后做旋转变换，最后将图片的左上角置为图片的原点

$$M = \begin{bmatrix} 1 & 0 & -x \\ 0 & - & -y \end{bmatrix} \begin{bmatrix} cos\Theta & -sin\Theta & 0 \\ sin\Theta & cos\Theta & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & x \\ 0 & 1 & y \end{bmatrix}$$

思考：缩放操作的变换矩阵是怎样的？

仿射函数1

warpAffine、 getRotationMatrix2D

cv2.warpAffine(src, M, dsize, dst, flags, borderMode, borderValue)

InputArray src：输入的图像
InputArray M：透视变换的矩阵
Size dsize：输出图像的大小
OutputArray dst：输出的图像
flags=INTER_LINEAR：输出图像的插值方法，
borderMode=BORDER_CONSTANT：图像边界的处理方式
borderValue=Scalar()：边界的颜色设置，一般默认是0

cv2.getRotationMatrix2D(center, angle, scale)
#用来获得变换矩阵M
Point2f center:旋转中心
double angle：旋转弧度，一定要将角度转换成弧度
double scale:缩放尺度

$$M = \begin{bmatrix} \alpha & \beta & (1-\alpha)\cdot center.x - \beta \cdot center.y \\ -\beta & \alpha & \beta \cdot center.x + (1-\alpha)\cdot center.y \end{bmatrix}$$

$$\alpha = scale \cdot \cos\theta$$
$$\beta = scale \cdot \sin\theta$$

仿射函数2

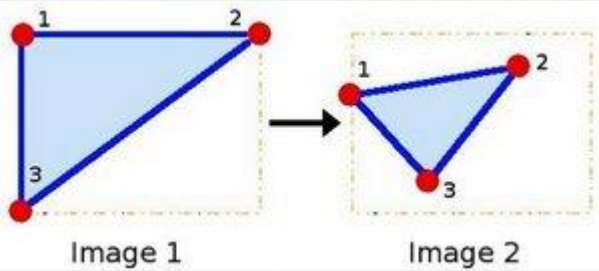
getAffineTransform

cv.getAffineTransform(src, dst)

src：输入图像的三角形顶点坐标。

dst：输出图像的相应的三角形顶点坐标。

因为矩阵 M 联系着两幅图片，我们以其表示两图中各三点直接的联系为例．见下图：

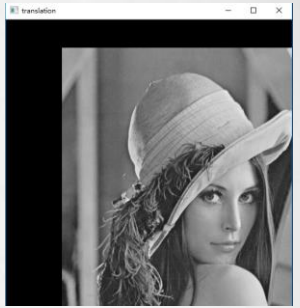


点1，2 和 3（在image1中形成一个三角形）与image2中三个点一一映射，仍然形成三角形，但形状已经大大改变．如果我们能通过这样两组三点求出仿射变换（你能选择自己喜欢的点），接下来我们就能把仿射变换应用到图像中所有的点．

仿射变换示例

```
import numpy as np
import cv2 as cv

# Load an color image in grayscale
img = cv.imread('lena.jpg',0)
rows,cols = img.shape

#Translation
M = np.float32([[1,0,100],[0,1,50]])
dst1 = cv.warpAffine(img,M,(cols,rows))

#Rotation
# cols-1 and rows-1 are the coordinate limits.
M = cv.getRotationMatrix2D(((cols-1)/2.0,(rows-1)/2.0),90,1)
dst2 = cv.warpAffine(img,M,(cols,rows))

#AffineTransform
pts1 = np.float32([[50,50],[200,50],[50,200]])
pts2 = np.float32([[10,100],[200,50],[100,250]])
M = cv.getAffineTransform(pts1,pts2)
dst3 = cv.warpAffine(img, M, (cols, rows))

cv.imshow('original',img)
cv.imshow('rotation',dst2)
cv.imshow('translation',dst1)
cv.imshow('Affine Transformation',dst3)
```

原图
（仿射变换的三个顶点已在图中标出）



平移

旋转

仿射变换

HSV

色调、饱和度、明度

HSV空间是由美国的图形学专家A. R. Smith提出的一种颜色空间，HSV分别是色调（Hue），饱和度（Saturation）和明度（Value）。
在HSV空间中进行调节就避免了直接在RGB空间中调节是还需要考虑三个通道的相关性。
OpenCV中H的取值是[0, 180)，其他两个通道的取值都是[0, 256)，通过HSV空间对图像进行调色更加方便：

```
cv2.cvtColor(src, code, dst, dstCn)
```

InputArray src: 输入图像
OutputArray dst: 输出图像
int code: 转换的代码或标识，即将什么制式的图片转换成什么制式的图片
int dstCn = 0: 目标图像通道数，如果取值为0，则由src和code决定

转换类型表

转换类型	Opencv2.x	Opencv3.x
RGB<-->BGR	CV_BGR2BGRA 、 CV_RGB2BGRA 、 CV_BGRA2RGBA 、 CV_BGR2BGRA 、 CV_BGRA2BGR	COLOR_BGR2BGRA,COLOR_RGB2BGRA 、 COLOR_BGRA2RGBA,COLOR_BGR2BGRA 、 COLOR_BGRA2BGR
RGB<-->GRAY	CV_RGB2GRAY 、 CV_GRAY2RGB 、 CV_RGBA2GRAY、CV_GRAY2GRBA	COLOR_RGB2GRAY,COLOR_GRAY2RGB 、 COLOR_RGBA2GRAY,COLOR_GRAY2GRBA
RGB<-->HSV	CV_BGR2HSV 、 CV_RGB2HSV 、 CV_HSV2BGR、CV_HSV2RGB	COLOR_BGR2HSV、COLOR_RGB2HSV、 COLOR_HSV2BGR、COLOR_HSV2RGB
RGB<-->YCrCb JPEG(或 YCC)	CV_RGB2YCrCb 、 CV_RGB2YCrCb 、 CV_YCrCb2BGR、CV_YCrCb2RGB (可以用 YUV 代替 YCrCb)	COLOR_RGB2YCrCb,COLOR_RGB2YCrCb, COLOR_YCrCb2BGR、COLOR_YCrCb2RGB (可以用 YUV 代替 YCrCb)
RGB <-->CIE XYZ	CV_BGR2XYZ,CV_RGB2XYZ, CV_XYZ2BGR, CV_XYZ2RGB	COLOR_BGR2XYZ,COLOR_RGB2XYZ, COLOR_XYZ2BGR, COLOR_XYZ2RGB
RGB<-->HLS	CV_BGR2HLS,CV_RGB2HLS, CV_HLS2BGR, CV_HLS2RGB	COLOR_BGR2HLS,COLOR_RGB2HLS, COLOR_HLS2BGR, COLOR_HLS2RGB
RGB<-->CIE L*a*b	CV_BGR2Lab,CV_RGB2Lab, CV_Lab2BGR, CV_Lab2RGB	COLOR_BGR2Lab,COLOR_RGB2Lab, COLOR_Lab2BGR, COLOR_Lab2RGB
RGB<-->CIE L*a*b	CV_BGR2Luv,CV_RGB2Luv, CV_Luv2BGR, CV_Luv2RGB	COLOR_BGR2Luv,COLOR_RGB2Luv, COLOR_Luv2BGR, COLOR_Luv2RGB
Bay-->RGB	CV_BayerBG2BGR,CV_BayerGB2BGR, CV_BayerRG2BGR,CV_BayerGR2BGR, CV_BayerBG2RGB,CV_BayerGB2RGB, CV_BayerRG2RGB,CV_BayerGR2RGB	COLOR_BayerBG2BGR,COLOR_BayerGB2BGR, COLOR_BayerRG2BGR,COLOR_BayerGR2BGR, COLOR_BayerBG2RGB,COLOR_BayerGB2RGB, COLOR_BayerRG2RGB,COLOR_BayerGR2RGB

HSV色彩变换示例

```
import cv2

img = cv2.imread('leaves.jpg')
img_hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

# H空间中，绿色比黄色的值高一点，所以给每个像素+15，黄色的树叶就会变绿
turn_green_hsv = img_hsv.copy()
turn_green_hsv[:, :, 0] = (turn_green_hsv[:, :, 0]+15) % 180
turn_green_img = cv2.cvtColor(turn_green_hsv, cv2.COLOR_HSV2BGR)
cv2.imwrite('turn_green.jpg', turn_green_img)

# 减小饱和度会让图像损失鲜艳，变得更灰
colorless_hsv = img_hsv.copy()
colorless_hsv[:, :, 1] = 0.5 * colorless_hsv[:, :, 1]
colorless_img = cv2.cvtColor(colorless_hsv, cv2.COLOR_HSV2BGR)
cv2.imwrite('colorless.jpg', colorless_img)

# 减小明度为原来一半
darker_hsv = img_hsv.copy()
darker_hsv[:, :, 2] = 0.5 * darker_hsv[:, :, 2]
darker_img = cv2.cvtColor(darker_hsv, cv2.COLOR_HSV2BGR)
cv2.imwrite('darker.jpg', darker_img)
```

原图

调整色调

调整饱和度

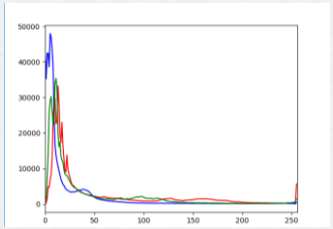
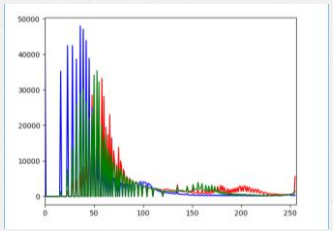
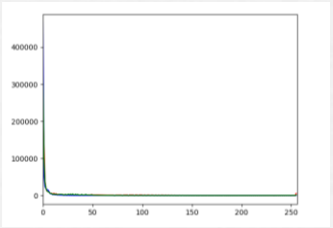
调整明度

Gamma变换

Gamma变换是矫正相机直接成像和人眼感受图像差别的一种常用手段，简单来说就是通过非线性变换(因为人眼对自然的感知是非线性的)让图像从对曝光强度的线性响应变得更接近人眼感受到的响应。Gamma压缩公式：

$$I(x,y) = I(x,y)^{gamma}$$

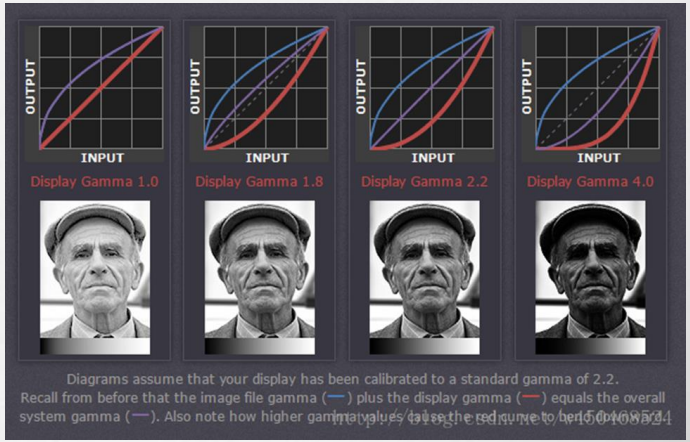
如果直方图中的成分过于靠近0或者255，可能就出现了暗部细节不足或者亮部细节丢失的情况。一个常用方法是考虑用Gamma变换来提升/降低暗部细节。


原图　　Gamma = 0.5　　Gamma = 2.2

Gamma变换示例

```
import cv2
import numpy as np

img = cv2.imread('leaf.jpg’)

# 定义Gamma矫正的函数
def gamma_trans(img, gamma):
    # 具体做法是先归一化到1，然后gamma作为指数值求出新的像素值再还原
    gamma_table = [np.power(x / 255.0, gamma) * 255.0 for x in range(256)]
    gamma_table = np.round(np.array(gamma_table)).astype(np.uint8)

    # 实现这个映射用的是OpenCV的查表函数
    return cv2.LUT(img, gamma_table)

# 执行Gamma矫正，小于1的值让暗部细节大量提升，同时亮部细节少量提升
img_corrected = gamma_trans(img, 0.5)
cv2.imwrite('gamma_corrected.jpg', img_corrected)
```



不同大小的Gamma对图片的影响

Opencv实例

OCR

OCR （Optical Character Recognition，光学字符识别）是指电子设备（例如扫描仪或数码相机）检查纸上打印的字符，通过检测暗、亮的模式确定其形状，然后用字符识别方法将形状翻译成计算机文字的过程；即，对文本资料进行扫描，然后对图像文件进行分析处理，获取文字及版面信息的过程。

一般流程如下：

图像文件输入 → 图像特征提取 → 分类器训练 → 预测 → 识别结果输出→ 计算正确率 → …

图像特征提取

HOG特征描述

方向梯度直方图（Histogram of Oriented Gradient, HOG）特征是一种在计算机视觉和图像处理中用来进行物体检测的特征描述子。它通过计算和统计图像局部区域的梯度方向直方图来构成特征。HOG特征结合SVM分类器已经被广泛应用于图像识别中，尤其在行人检测中获得了极大的成功。

HOG特征提取方法就是将一个image（你要检测的目标或者扫描窗口）：

1）灰度化（将图像看做一个x,y,z（灰度）的三维图像）；

2）采用Gamma校正法对输入图像进行颜色空间的标准化（归一化）；目的是调节图像的对比度，降低图像局部的阴影和光照变化所造成的影响，同时可以抑制噪音的干扰；

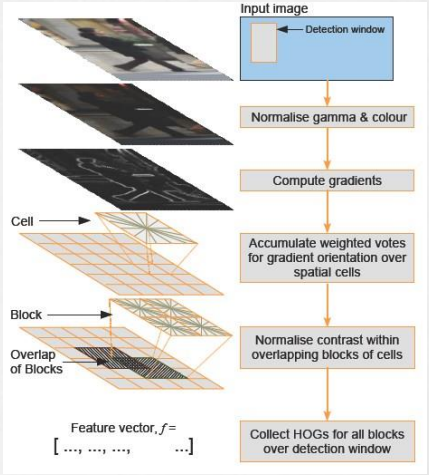
3）计算图像每个像素的梯度（包括大小和方向）；主要是为了捕获轮廓信息，同时进一步弱化光照的干扰。

4）将图像划分成小cells（例如10*10像素/cell）；

5）统计每个cell的梯度直方图（不同梯度的个数），即可形成每个cell的descriptor；

6）将每几个cell组成一个block（例如2*2个cell/block），一个block内所有cell的特征descriptor串联起来便得到该block的HOG特征descriptor。

7）将图像image内的所有block的HOG特征descriptor串联起来就可以得到该image（你要检测的目标）的HOG特征descriptor了。这个就是最终的可供分类使用的特征向量了。

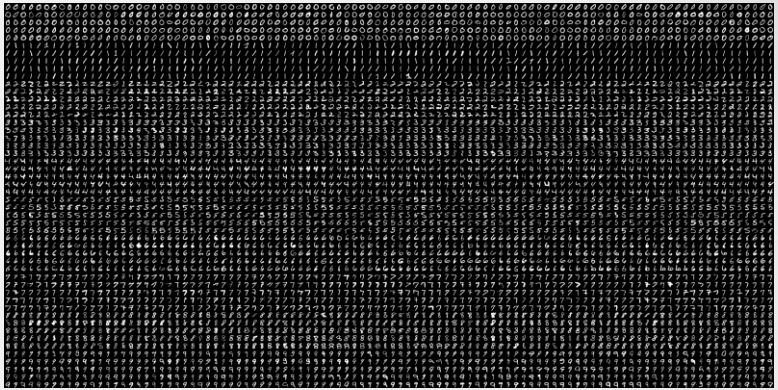


HOG的经典论文：Dalal N, Triggs B. Histograms of oriented gradients for human detection[C]//Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. IEEE, 2005, 1: 886-893. (2016:Google Citation: 14046)

digits.py

main

```
#main
digits, labels = load_digits(DIGITS_FN) #图像切分 导入

    print('preprocessing...')
    # shuffle digits 打乱数字
    rand = np.random.RandomState(321)   #随机种子321
    shuffle = rand.permutation(len(digits))
    digits, labels = digits[shuffle], labels[shuffle]

    digits2 = list(map(deskew, digits)) #纠正图片倾斜
    samples = preprocess_hog(digits2)   #计算hog特征

    train_n = int(0.9*len(samples)) #划分训练测试集
    cv.imshow('test set', mosaic(25, digits[train_n:]))
    digits_train, digits_test = np.split(digits2, [train_n])
    samples_train, samples_test = np.split(samples, [train_n])
    labels_train, labels_test = np.split(labels, [train_n])


    print('training KNearest...')   #knn分类器
    model = KNearest(k=4)
    model.train(samples_train, labels_train)
    vis = evaluate_model(model, digits_test, samples_test, labels_test)
    cv.imshow('KNearest test', vis)

    print('training SVM...')    #SVM分类器
    model = SVM(C=2.67, gamma=5.383)
    model.train(samples_train, labels_train)
    vis = evaluate_model(model, digits_test, samples_test, labels_test)
    cv.imshow('SVM test', vis)
    print('saving SVM as "digits_svm.dat"...')
    model.save('digits_svm.dat')
```



- 数据集总数5k，每个数字500
- 每张数字图片block大小20*20
- HOG划分的cell大小10*10
- 直方图bin个数16
- HOG向量的维数bin*(block/cell)=64
- 训练集：测试集 = 9：1

digits.py

预测结果

测试集

Knn预测结果
error: 3.40 %

SVM预测结果
error: 1.80 %

比较流行的ORC工具：Tesseract
https://github.com/tesseract-ocr/tesseract/wiki

应用实例参考

1、使用PIL 图像处理库，划分图像的皮肤区域，识别图片是否为色情图片：

https://www.shiyanlou.com/courses/589

2、验证码识别：

http://www.cnblogs.com/beer/p/5672678.html

2、二维码识别（结合zbar库）：

http://www.tuicool.com/articles/quEziy6

计算机视觉

TED：如何教计算机理解图片？

https://www.ted.com/talks/fei_fei_li_how_we_re_teaching_computers_to_understand_pictures?language=zh-cn

李飞飞　女

美国斯坦福大学计算机科学系

2015年12月1日，入选2015年“全球百大思想者”

2016年，加入Google 担任云计算机器学习负责人

实验报告（六）

截止时间
2018年5月13日
18:00

请各位同学在截止时间之前将实验报告发送给课程学习委员

实验六 内容：

1. 用OpenCV的仿射变换实现图片缩放
2. 理解HOG、ORC过程，修改digits.py或独立编程，实现数字图像的识别，自己找测试数据（可以是一个或多个数字）。示例：

实验要求：

1. 独立完成，请勿抄袭，自行选择模块内容请勿与其他同学完全雷同。
2. 关键步骤请截图，并保存在实验报告文档中，**截图总数应大于10张**。
3. 实验报告文档命名格式：学号-姓名-实验X
4. 如在实验基础上有创新与拓展、可获得额外分数奖励。