图像预处理的几个模块:PIL、scipy.misc、OpenCV、TensorFlow

原创

蓬莱道人 于 2018-09-01 17:14:40 发布
分类专栏： Python

1、PIL库

Python Imaging Library (PIL)是PythonWare公司提供的免费的图像处理工具包,是python下的图像处理模块,支持多种格式,并提供强大的图形与图像处理功能。虽然在这个软件包上要实现类似MATLAB中的复杂的图像处理算法并不太适合,但是Python的快速开发能力以及面向对象等等诸多特点使得它非常适合用来进行原型开发。对于简单的图像处理或者大批量的简单图像处理任务,python+PIL是很好的选择。

```
from PIL import Image,ImageEnhance      #调用库

im = Image.open("E:/testdata/01.jpg")  #打开图片
print (im.format, im.size, im.mode)    #打印图像信息
'''
  format:图像属性，如png，jpeg
  size：图像分辨率
  mode：
        1     1位像素，黑和白，存成8位的像素
 L 8位像素，黑白
 P 8位像素，使用调色板映射到任何其他模式
 RGB 3× 8位像素，真彩
 RGBA 4×8位像素，真彩+透明通道
 CMYK 4×8位像素，颜色隔离
 YCbCr 3×8位像素，彩色视频格式
 I 32位整型像素
 F 32位浮点型像素
```

```
'''
new_im = im.convert('L')                          # 将图像转为其它模式
new_im.save("E:/testdata/02.png")                 # 保存图像，可以改变图像格式

box = (0, 0, 500, 500)                            # 确定拷贝区域大小(左上角坐标，右下角坐标)
region = im.crop(box)                             # 将im表示的图片对象拷贝到region中，大小为box

region = im.resize((400, 400),Image.ANTIALIAS) # 改变图像尺寸
'''
  函数原型：im.resize(size, filter):
    size：所要求的尺寸，是一个二元组：（width, height）
    filter：为NEAREST、BILINEAR、BICUBIC或者ANTIALIAS之一

'''


im_30 = im.rotate(30, Image.NEAREST,1)          # 图像逆时针旋转30度
'''
  函数原型：im.rotate(angle,filter=NEAREST, expand=0) ⇒ image
    angle：逆时针旋转的角度值
    filter：NEAREST、BILINEAR或者BICUBIC之一
    expand，如果为true，表示输出图像足够大，可以装载旋转后的图像。
            如果为false或者缺省，则输出图像与输入图像尺寸一样大。
'''
r,g,b=im.split()                                  # 分割成三个通道，此时r,g,b分别为三个图像对象。
new_im=Image.merge("RGB",(r,g,b))                 # 将r,g,b,三通道合并

example = np.random.randint(0,255,size=(300,300,3))  # numpy数组
new_img = Image.fromarray(example, 'RGB')       # 将numpy的narray转为Image类，第二个参数为mode

new_im.show()                                     # 显示图像
# ---------将索引图变成彩色图-------------------
cmap = [np.random.randint(0, 255) for x in range(768)]  # 生成色彩隐射图，大小需要为256*3=768
img = img.convert('P')                            # 转换为P模式
img.putpalette(cmap)                              # 给索引图着色
img.show()                                        # 显示图像

#----------图像增强-----------------------
# 增强亮度
enhanceImg = ImageEnhance.Brightness(img)
# 图片尖锐化
enhanceImg = ImageEnhance.Sharpness(img)
# 对比度增强
enhanceImg = ImageEnhance.Contrast(img)
# 色彩增强
enhanceImg = ImageEnhance.Color(img)
enhanceImg.enhance(2.0).show()    # 2.0表示增强两倍，1.0表示不增强。
```

注意：Image只接收uint8类型的数据，如果传入float32类型的数据就会出错。

2、scipy.misc

python在科学计算领域有三个非常受欢迎库，numpy、SciPy、matplotlib。numpy是一个高性能的多维数组的计算库，SciPy是构建在numpy的基础之上的，它提供了许多的操作numpy的数组的函数。SciPy是一款方便、易于使用、专为科学和工程设计的python工具包，它包括了统计、优化、整合以及线性代数模块、傅里叶变换、信号和图像图例，常微分方差的求解等，下面就简单的介绍一下SciPy在图像处理方面的应用，如果专业做图像处理当然还是建议使用opencv。

```
from scipy.misc import imread,imsave,imresize
import scipy.io

img = imread("E:/testdata/01.jpg")    # 读取图像
print (img)                           # numpy 数组

img_type = img.dtype                  # 获取图片的数据类型
print(img_type)                       # uint8

img_shape = img.shape                 # 获取图片的大小
print(img_shape)                      # (310, 493, 3)

newimg=imresize(img,(100,100))        # 图像裁剪
'''
  函数原型：imresize(arr, size, interp='bilinear', mode=None)
    arr：ndarray类型的图像
    size：裁剪后的尺寸，是一个元组
    interp ：  插值方法，为‘nearest’, ‘lanczos’, ‘bilinear’, ‘bicubic’ or ‘cubic’
    mode：PIL中的图像模式 (‘P’, ‘L’等等)，在resize之前先转换模式
'''

imsave("timg_color.png",newimg)       # 保存图片

# ------------使用scipy加载.mat文件-------------
data = scipy.io.loadmat('Data.mat')
print (data)                          # 一般data为一个字典类型的数据
```

3、OpenCV

OpenCV是一个开放源代码的计算机视觉应用平台，由英特尔公司下属研发中心俄罗斯团队发起该项目，开源BSD证书，OpenCV的目标是实现实时计算机视觉，，是一个跨平台的计算机视觉库。从开发之日起就得到了迅猛发展，获得了众多公司和业界大牛的鼎力支持与贡献，因为是BSD开源许可，因此可以免费应用在科研和商业应用领域。

OpenCV中已经包含如下应用领域功能：二维和三维特征工具箱、运动估算、人脸识别系统、姿势识别、人机交互、移动机器人、运动理解、对象鉴别、分割与识别、立体视觉、运动跟踪、增强现实（AR技术）。基于上述功能实现需要，OpenCV中还包括以下基于统计学机器学习库：Boosting算法、Decision Tree(决策树)学习、Gradient Boosting算法、EM算法(期望最大化)、KNN算法、朴素贝叶斯分类、人工神经网络、随机森林、支掌向量机。

编程语言：OpenCV中多数模块是基于C++实现，其中有少部分是基于C语言实现，当前OpenCV提供的SDK已经支持C++、Java、Python等语言应用开发。当前OpenCV本身新开发的算法和模块接口都是基于C++产生。OpenCV-Python使用Numpy，这是一个高度优化的数据库操作库，具有MATLAB风格的语法。所有OpenCV数组结构都转换为Numpy数组。这也使得与使用Numpy的其他库（如SciPy和Matplotlib）集成更容易。接下来介绍OpenCV的核心操作：

```
import cv2

img = cv2.imread('E:/testdata/01.jpg')  # 读取图像
print( img.shape )
print (img)

b,g,r = cv2.split(img)                    # 拆分图像通道
img = cv2.merge((b,g,r))                  # 合并图像通道

res=cv2.resize(img,(50,50),interpolation=cv2.INTER_CUBIC)   # 图像缩放
'''
 INTER_NEAREST 最近邻插值
 INTER_LINEAR 双线性插值（默认设置）
 INTER_AREA 使用像素区域关系进行重采样。 它可能是图像抽取的首选方法，因为它会产生无云纹理的结果。 但是当图像缩放时，它类
 INTER_CUBIC 4x4像素邻域的双三次插值
 INTER_LANCZOS4 8x8像素邻域的Lanczos插值
'''

img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB) # 转换读取图像的通道顺序，cv2默认为BGR顺序,这里转为RGB，其它软件均为RGB

cv2.imwrite("hehe.jpg",res)               # 保存图像

cv2.imshow("res",res)                     # 显示图像
cv2.waitKey(0)
cv2.destroyAllWindows()
```

注意：cv2默认为 BGR顺序，而其他软件（PIL、scopy.misc）一般使用RGB，所以当cv2和其它混用时需要用cv2.cvtColor()转换通道

4、tf.image模块

（1）编解码（编解码中处理的数据类型均为tf.uint8）：

tf.image.decode_jpeg(contents, channels=None, ratio=None, fancy_upscaling=None, try_recover_truncated=None, acceptable_fraction=None, name=None)
tf.image.encode_jpeg(image, format=None, quality=None, progressive=None, optimize_size=None, chroma_downsampling=None, density_unit=None, x_density=None, y_density=None, xmp_metadata=None, name=None)
tf.image.decode_png(contents, channels=None, name=None)
tf.image.encode_png(image, compression=None, name=None)

（2）缩放（第一个函数和后面四个等价，这几个函数接受任意的数据类型，但是输出的类型为tf.float32）：

tf.image.resize_images(images, new_height, new_width, method=0)
tf.image.resize_area(images, size, name=None)
tf.image.resize_bicubic(images, size, name=None)
tf.image.resize_bilinear(images, size, name=None)
tf.image.resize_nearest_neighbor(images, size, name=None)

（3）裁剪：

tf.image.resize_image_with_crop_or_pad(image, target_height, target_width)
tf.image.pad_to_bounding_box(image, offset_height, offset_width, target_height, target_width)
tf.image.crop_to_bounding_box(image, offset_height, offset_width, target_height, target_width)
tf.image.random_crop(image, size, seed=None, name=None)
tf.image.extract_glimpse(input, size, offsets, centered=None, normalized=None, uniform_noise=None, name=None)

（4）翻转：

tf.image.flip_up_down(image)
tf.image.random_flip_up_down(image, seed=None)
tf.image.flip_left_right(image)
tf.image.random_flip_left_right(image, seed=None)

```
#coding=utf-8
import tensorflow as tf
import numpy as np
import os
from PIL import Image

# 从文件读取数据，得到的是二进制文件
image_data = tf.gfile.FastGFile("D:/test/2.jpg", 'rb').read()
with tf.Session() as sess:
  # 将二进制数据解码为一个Tensor,此时的数据类型为tf.uint8
  img_data = tf.image.decode_jpeg(image_data)
  print (img_data.eval().shape)

  # 对图像进行resize，0：双线性差值。1：最近邻居法。2：双三次插值法。3：面积插值法。
  resized = tf.image.resize_images(img_data, [500, 500], method=0)

  # TensorFlow的函数处理图片后存储的数据是float32格式的，需要转换成uint8才能正确打印图片。
  resized = np.asarray(resized.eval(), dtype='uint8')

  # 显示图片
  new_img = Image.fromarray(resized, 'RGB')
  new_img.show()

  # 对图像进行编码,并且保存图像
  encoded_image = tf.image.encode_jpeg(resized)
  print (type(encoded_image))
  with tf.gfile.GFile("./3.jpg","wb") as f:
    f.write(encoded_image.eval())
```

（5）转换图像的数据类型：tf.image.convert_image_dtype(image，dtype)

如果传入的数据类型为uint8，该函数可以将0-255的uint8类型的像素值归一化为0-1。如果传入的数据类型为tf.float32，则该函数对图像不做任何处理。如果传入的数据类型为tf.int32，则处理后会变成极小的小数。

参考：https://blog.csdn.net/hopyGreat/article/details/81586563