Python Imaging Library

中文版

译者：放学后

邮箱：941642558@qq.com

Table of Contents

介绍

Python Imaging Library (PIL)

- PIL是Python中强大的图像处理类库，通过这个翻译进行学习这个类库，下面这幅图片就是通过PIL类库进行处理出来的效果图，图像为黑白二值模式。
- 文档若有翻译理解错误的地方，联系邮箱：941642558@qq.com

概念

概念

PIL用于处理栅格化的图像，也就是像素矩形。

波段

图像可以由一个或多个波段数据组成。 PIL能使你可以在单个图像中存储多个波段，前提是它们都具有相同的尺寸和深度。使用getbands方法，可获取图像中的波段的数量和名称。

模式

图像的模式定义图像中像素的类型和深度。 当前版本支持以下标准模式：

1 (1-bit pixels, black and white, stored with one pixel per byte) 黑白二值模式

L (8-bit pixels, black and white) 黑白灰度模式

P (8-bit pixels, mapped to any other mode using a colour palette) 调色板（索引）模式

RGB (3x8-bit pixels, true colour) RGB真彩色

RGBA (4x8-bit pixels, true colour with transparency mask) RGBA真彩色

CMYK (4x8-bit pixels, colour separation) CMYK模式，一种套色模式

YCbCr (3x8-bit pixels, colour video format)

I (32-bit signed integer pixels) 有符号整数像素

F (32-bit floating point pixels) 浮点数像素

Image模块

The Image Module

该Image模块也一个Image类来代表一个PIL image,这个模块提供了许多工厂函数：比如从文件中载入图像、创建新的图像。

示例

打开，旋转，显示一个图像

```
from PIL import Image
im=Image.open('bride.jpg') #通过图像路径载入图片
im.rotate(45).show() #旋转45度,并用外部的默认的图像查看器显示
```

创建缩略图

```
from PIL import Image
import glob,os
size=128,128
for infile in glob.glob("*.jpg"):
    file,ext=os.path.splitext(inflie) #分离后缀
    im=Image.open('infile')
    im.thumbnail(size,Image.ANTIALIAS)
    im.save(file+'.thumbnail','JPEG')
```

函数

new
Image.new(mode,size,color=0) ⇒ image

- 由给定的模式和大小，新建一张图像
- size为一个二元尺寸元组，单位为像素
- color可以为单个值(单频段图像)也可多个值组成的元组（多频段图像）
- color为None,图像将不会进行初始化

```
from PIL import Image
im=Image.new('RGB',(512,512),'white')
```

open

Image.open(fp,mode='r') ⇒ image

- 打开，识别给定的图像
- 这是一个迟缓（lazy）操作，读取文件头，但实际数据并没有读取
- load方法就会强力载入
- mode若给定必须为'r'
- fp可以为一个文件路径或者是一个文件对象(必须是二进制模式打开)

```
from PIL import Image
im=Image.open('lenna,jpg')
```

```
from PIL import image
from StringIO import StringIO
im=Image.open(StringIO(data))
```

blend

Image.blend(img1,img2,alpha) ⇒ image

- 使用alpha常量,通过将两副图像相互插入创建新的图像，类似于图像合并，两者透明度相加为1
- out = image1 *(1.0 - alpha)* + *image2* alpha
- 效果图如下： 

composite

Image.composite(image1, image2, mask) ⇒ image

- 使用透明蒙板混合图像创建合成图像
- mask为一个image对象，使用该image对象的对应的像素值作为alpha值。
- mask图像的模式可以为'1','L','RGBA'
- 与blend十分相似

eval
Image.eval(image, function)⇒ image

- 对给定的图像的每一个像素值应用一个函数，该函数只能有一个整型参数，来接收像素值
- 多频段的图像像素也不例外

frombuffer
Image.frombuffer(mode, size, data) ⇒ image

- 创建一个图像内存引用字节缓冲区中的像点数据
- 这个函数只用来解码像素数据

merge

Image.merge(mode, bands) ⇒ image

- 从给定的多个单波段的图像创建一个新多波段的图像
- band参数为image列表或者元组，与模式描述的波段一一对应

方法

Image类的实例都有以下方法，除非另有声明，所有方法都会返回一个新的实例，保存生成图像

convert

convert(mode=None, matrix=None, dither=None, palette=0, colors=256)⇒ image

- 将图像转为另一种模式，返回转换后图像的副本
- 当将彩色图像转换为黑白灰度图像(模式为'L'),该库使用ITU-R 601-2 luma转换： L = R *299\/1000* + *G* 587\/1000 + B * 114\/1000
- 当转换黑白二值图像(模式为'1'),源图像首先被转换成黑白灰度图像，产生的值大于127设置为白色，图像是抖动的,使用其他阈值，使用point方法
- 模式'P'为调色板图像模式，也称为索引模式，将图像最需要的颜色放到一个颜色表中，当将图像转换为调色板模式的时候，会给每个像素分配一个固定的颜色值，这些颜色保存在简洁的颜色表中，可以通过限制调色板、索引颜色将会减少文件大小
- 当前版本支持'L','RGB','CMKYK'之间的所有可能的相互转换
- 矩阵转换只支持'L'和'RGB'模式* dither：抖动方法,当模式从'RGB'转换到'P',从'RGB'或者'L'到'1'时使用
- colors:当调色板是Image.ADAPTIVE时，控制调色板颜色的阶梯数
- matrix:进行矩阵转换，为一个4位或者16位的元组

copy

im.copy()⇒ image

- 创建一个图像的副本

crop

im.crop(box)⇒ image

- 返回来自当前图像矩形区域的副本，这个盒子一个4元元组，定义其坐标
- 这个一个延缓操作，源图像的变化有可能会影响到被裁剪的图像。为了获取一个独立出来的副本，在裁剪的图像上调用load方法

draft

im.draft(mode,size)

getbands

im.getbands()⇒ image

- 返回一个包含每个波段的名字的元组，例如：对于”RGB“图像而言，返回("R","G","B")

getbbox

im.getbbox() ⇒ 4-tuple or None

- 计算图像中(非零区域)边界框，返回一个四元元组，如果图像为空则返回None

getcolors
im.getcolors(maxcolors=256) ⇒ a list of (count, color) tuples or None

- 返回一个未排序的(count,color)元组列表，count为对应颜色在图像中出现的次数
- 如果maxcolors值被超标，该方法将会停止计数，返回None
- 默认的maxcolors值为256，因此为了获取所有颜色的，你可以将size[0]*size[1]传递给maxcolors，但你在处理大图像的时候必须保证足够多的内存

getextrema

im.getextrema() ⇒ 2-tuple

- 返回一个二元元组，包含图像最大最小值，当前版本，只工作于单波段图像

split

im.split() ⇒ sequence

- 将多波段图像分成多个单波段的图像，返回一个单波段图像元组

seek
im.seek(frame)

- 在序列文件中搜寻给定的帧，超过范围就会抛出EOFError异常
- 当文件打开时，该库自动定位到第0帧
- 当前版本中，大多数序列格式只能允许搜寻下一帧

show

im.show()

- 显示一个图像，该方法主要用于调试

tell

im.tell()⇒ integer

- 返回当前帧号

thumbnail

im.thumbnail(size, filter)

- 创建图像的缩略图
- filter参数为以下参数之一：
    - NEAREST
    - BILINEAR
    - BICUBIC
    - ANTIALIAS(质量最好)
- 忽略此参数，默认为ANTIALIAS
- 注意：bilinear、bicubic过滤器在当前版本中,并不能很好地生成缩略图，如果不考虑速度问题的话，可以使用ANTIALIAS过滤器
- 此方法在原图像进行修改，最好使用图像的副本进行操作，该方法返回None

getpixel

im.getpixel(xy) ⇒ value or tuple

- 返回给定位置的像素，如果图像是一个多层图像，将会返回一个元组

- 这个方法执行缓慢，如果你需要处理图像较大的部分，你可以使用getdata()和load()访问

histogram

im.histogram() ⇒ list

- 返回这个图像的直方图，返回的是一个像素计数列表，每一个对应其源图像每一个像素值
- mask参数(作用暂时不清楚)

load

im.load()

- 为这个图像分配空间，从文件中载入，大多数情况下，你不需要调用这个方法，因为这个Image类会自动装载这个打开的文件，当它被第一次访问的时候
- 在1.1.6之后的版本，load会返回一个像素访问对象，可以用此进行访问和修改像素
- 这个对象有点像一个二维数组，通过这个对象访问，比getpixel和putpixel速度更快

offset

im.offset(xoffset, yoffset) ⇒ image

- 已经不推荐使用

paste

im.paste(image, colour,box,mask)

- 将一个图像粘贴到这个图像上
- box参数可以为二元元组，定义其左上角的位置，为四元元组，定义其box的坐标和大小，同时被粘贴的图像必须匹配这个区域的大小，为None，等同于(0,0)
- 如果模式不匹配，被粘贴的图像将会转变为这个图像的模式
- colour参数用指定颜色进行填充
- mask参数仅更新由掩码指示的区域。 您可以使用“1”，“L”或“RGBA”图像（在后一种情况下，Alpha波段用作掩码），在掩码为255的情况下，给定图像被原样复制。 在掩码为0的情况下，保留当前值， 中间值可用于透明效果
- 请注意，如果粘贴“RGBA”图像，将忽略Alpha波段，可以使用与源图像和掩码相同的图像来解决此问题

point

im.point(function,table,mode) ⇒ image

- table参数，返回一个图像副本，每个像素映射到颜色查询表上
- function参数，该参数将应用到图像的每个像素上
- mode参数用于为输出对象指定新的模式

tobitmap

im.tobitmap() ⇒ string

- 以字符串的形式，返回转换为Xll bitmap图像

tostring

im.tostring() ⇒ string

- 使用标准的原始编码器，返回包含像素数据的字符串
- 该方法只能返回原始的像素数据，为将图像以标准的文件格式保存为字符串，传递一个类StringIO对象给save方法

transform

putpixel

im.putpixel(xy, colour)

- 修改给定位置的像素值
- 对于多波段图像而言,colour将会一个元组
- 该方法执行相对缓慢，load提供了一个更快的方法

- **quantize**
  im.quantize(colors, **options) ⇒ image
- 已经不推荐使用，使用给定的颜色数，量化，转换'L'或者'RGB'图像为'P'图像

resize

im.resize(size, filter) ⇒ image

- 返回一个改变大小的图像副本
- filter参数： NEAREST BILINEAR BICUBIC ANTIALIAS
- 如果忽略该参数，或者图像的模式为'1'或者'L',将会被设置为NEAREST

rotate
im.rotate(angle, filter=NEAREST, expand=0) ⇒ image

- 返回指定旋转角度逆时针旋转的图像副本，相对于图像的中心
- filter参数同resize方法
- expand参数，指示图像旋转后是否扩展图像以显示整个原图像内容

save

im.save(outfile, format, options…)

- 用于保存一个文件，如果格式被省略，将文件格式将会由文件名的扩展决定
- options参数用于提供额外的指示
- outfile参数值也可以是一个类文件对象
- 保存失败，通常会引发一个IOError异常

putalpha

im.putalpha(band)

- 将给定的波段复制到当前图像alpha层
- 这个图像必须是'RGBA'模式的图像
- 波段的模式必须是'L'或者'1'
- 在PIL版本1.1.5之后，该方法也应用在其他模式上。
- 这个转换在原图像上进行，进行这样的转换，模式匹配当前模式， 但是有个alpha层(这通常意味着'LA'或者'RGBA')
- 这个band参数可以是一个image对象，也可以是一个颜色值

putpalette

im.putpalette(sequence)

- 给'P'或者'L'图像附加一个调色板，对'L'，图像来说， 模式被为'P'，调色板序列应该包含768个整数值，其中每组三个值表示相应像素索引的红色、绿色、和蓝色值
- 你可以使用768字节的字符串替代整数序列

putdata

im.putdata(data, scale, offset)

- 将像素值从一个序列对象上复制到图像上，从左上角(0,0)开始，scale(缩放因子)和offset(偏移量)用于调整序列对象的像素值
- pixel = value * scale + offset
- 如果scale被忽略，将默认为1.0，offset被忽略，默认为0.0

getdata

im.getdata() ⇒ sequence

- 以序列对象返回一个图像的内容，该序列包含像素的值
- 序列对象是扁平的，因此第一行的值紧跟在第零行的值之后，依此类推
- 该方法的返回的序列对象是PIL内部的数据类型，通过list()可以将其转换为普通的序列

transpose

im.transpose(method) ⇒ image

- 返回一个翻转或者旋转的图像副本
- 该参数为下面之一：
  - FLIP_LEFT_RIGHT
  - FLIP_TOP_BOTTOM
  - ROTATE_90
  - ROTATE_180
  - ROTATE_270

属性

Image类的实例对象有以下属性：

format

im.format ⇒ string or None

源文件的文件格式，对于由该库的工厂函数或者已存在图像对象的方法创建的图像，这个属性被设置为None

mode

im.mode ⇒ string

- 图像模式，这个字符串指定了图像使用的像素格式。
- 一般来说，该值可为'1','L'，'RGB','CMYK'

size

im.size ⇒ (width, height)

- 图像大小，是一个二元元组

palette
im.palette ⇒ palette or None

- 颜色表，如果有，且模式为'P'，那么将会是ImagePalette类的实例，否则为None

info

im.info ⇒ dictionary

- 一个保存图像相关数据的字典，文件处理程序将会使用这个字典传递一些非图像信息
- 大多数方法当返回一图像的时候个新的会忽略这个字典，如果需要，在open方法返回后，保留对其的

ImageChops模块

The ImageChops Module

ImageChops模块包含了一些图像的算术操作，称之为'channel operations'(chops)。 该模块可以用来制作特效，图像合成，算法绘图等等

函数 大多数通道操作需要一个或者两个参数，返回一个新的图像。除非另有声明，通道操作的结果总是被剪切到[0,max]范围内 更多的预制操作，查看ImageOps模块

constant

ImageChops.constant(image, value) ⇒ image

- 用给定的灰度值填充一个通道
- 返回一个'L'模式的图像
- 源码如下：

```
return Image.new("L", image.size, value)
```

duplicate

- duplicate为复制的意思
- 为Image.copy()的别名
- 源码如下：

```
return image.copy()
```

invert

ImageChops.invert(image) ⇒ image

- 图像色彩的反转
- out = MAX - image

- 效果如下：

原图：  效果图： 

lighter

ImageChops.lighter(image1, image2) ⇒ image

- 逐个像素地，比较image1和image2
- 返回一个新的图像,每个像素包含较大(亮)的值,max(image1,image2)

- 效果图如下：

Image1：  Image2：  效果图： 

darker

ImageChops.darker(image1, image2) ⇒ image

- 与lighter效果相反

- 效果图如下：

Image1：  Image2：  效果图： 

difference ImageChops.difference(image1, image2) ⇒ image

- 将两幅图像做差值运算，并返回其绝对值的图像
- out = abs(image1 - image2)

- 效果图如下：

Image1：  Image2：  效果图： 

multiply

ImageChops.multiply(image1, image2) ⇒ image

- 两幅图像相互叠加，并不是将整个图像作为整体进行叠加，是像素级的叠加
- out = image1 * image2 \/ MAX

- 效果图如下：

Image1：  Image2：  效果图： 

screen

ImageChops.screen(image1, image2) ⇒ image

- 将两幅相互反转的图像进行相互叠加
- out = MAX - ((MAX - image1) * (MAX - image2) \/ MAX)

add

ImageChops.add(image1, image2, scale, offset) ⇒ image

- 将两幅图像进行相加操作，结果除以缩放因子，在加上偏移量
- scale和offset默认为1和0

- 效果图如下：

Image1：  Image2：  效果图： 

subtract

ImageChops.subtract(image1, image2, scale, offset) ⇒ image

- 与add相反，进行的是相减操作

- 效果图如下：

Image1：  Image2：  效果图： 

blend

ImageChops.blend(image1, image2, alpha) ⇒ image

- 功能与Image模块的blend函数功能相同

composite

ImageChops.composite(image1, image2, mask) ⇒ image

- 功能与Image模块的composite函数相同

offset ImageChops.offset(image, xoffset, yoffset) ⇒ image

- 返回图像的副本，图像的数据由给的距离进行偏移
- 若省略yoffset,其默认等于xoffset
- 效果图如下： 原图：  效果图: 

ImageColor模块

The ImageColor Module

该模块包含了颜色表和从CSS3样式颜色说明符到RGB元组的转换器，该模块由Image.new和ImageDraw模块使用。

颜色名称

ImageColor模块支持下面的字符串格式：

- 16进制颜色说明符，像这样：'#rgb'或者'rrggbb',例如：'#ff0000'指定纯红
- RGB函数，像这样:'rgb(red,green,blue)',颜色值为[0,255]范围的整数值，也可以为一个百分比，像这样rgb(100%,0%,0%)
- HSL函数，像这样hsl(hue, saturation%, lightness%)，参数分别为色相，饱和度，明度；色相为[0,360]中一个值，饱和度和明度为[0%,100%]中的值
- 常见的HTML颜色名称，ImageColor模块提供了140种标准的颜色名称，颜色名不区分大小写，例如：'#Red'和'#red'都代表纯红

函数

getrgb

getrgb(color) ⇒ (red, green, blue)

*将一个字符串颜色，转换为RGB值的元组，若不能解析，将引发ValueError异常

getcolor

getcolor(color, mode) ⇒ (red, green, blue) or integer

- 作用与getrgb类似，但是模式如果不是彩色或者调色板模式，将会把RGB值转换为灰度值

ImageDraw模块

The ImageDraw Module

ImageDraw模块为Image图像提供了2D绘图，你可以使用这个模块创建图像、注释或者润饰已经存在的图像，快速生成用于Web的图形

关于PIL的更高级的绘图的库，查看aggdraw模块

示例

在图像上绘制灰色交叉线

```
import Image, ImageDraw
im = Image.open("lena.pgm")
draw = ImageDraw.Draw(im)
draw.line((0, 0) + im.size, fill=128)
draw.line((0, im.size[1], im.size[0], 0), fill=128)
del draw
# write to stdout
im.save(sys.stdout, "PNG")
```

概念

坐标

图形界面使用和PIL自身相同的坐标系统，左上角为坐标原点(0,0)

颜色

你可以是数字元组来指定颜色，对于'1'、'L'和'I'模式，使用一个整数，对于'RGB'图像，使用一个3元整型元组,对于'F'，使用整型或者浮点型数值。

对于调色板图像（模式'P'），使用整数作为颜色索引。在1.1.4版本后（包括1.1.4），你可以使用表示RGB的3元元组，图层将自动分配索引，只要你使用的颜色不超过256种，也就是在颜色表范围内。

颜色名

该部分介绍与ImageColor模块的颜色名称介绍相同，请参考ImageColor模块的Colour Names。

Fonts

PIL可以使用bitmap字体、OpenType\/TrueType字体。

Bitmap字体以PIL自己的格式保存，一般来讲，每种字体包含两个文件：'.pil'文件,'.pdm'文件(通常)。前者包含字体度量，后者栅格化数据。

使用ImageFont模块的load函数来载入一个bitmap字体。

使用ImageFont模块的truetype函数来载入OpenType\/TrueType字体；注意:这个函数依赖第三方库，并不是在PIL版本中是可用的。

（IronPIL）使用ImageFont模块的Font构造器来创建一个内建的字体。

函数

Draw

Draw(image) ⇒ Draw instance

- 在给定的图像上创建一个绘图对象
- (IronPIL)你可以使用ImageWin模块的HWND和HDC对象，而不是Image对象，你可以直接在屏幕上绘图
- 注意：操作是在原图像进行修改的

方法

arc

draw.arc(xy, start, end, options)

- 在给定的边框中，由给定的起始角和结束角来绘制一个圆弧
- outline参数将指定绘制使用颜色

bitmap

draw.bitmap(xy, bitmap, options)

- 在给定的位置绘制一个位图（蒙版），这个位图应该是一个透明的蒙版(模式'1')或者磨砂（模式'L'或者'RGBA'）

- 这等价于image.paste(xy, color, bitmap)

chord

draw.chord(xy, start, end, options)

- 与arc类似，绘制的是圆的弦线
- outline参数指定矩形使用的轮廓颜色，fill参数指定其内部使用的填充颜色

ellipse

draw.ellipse(xy, options)

- 在给定的边框内部，绘制一个椭圆
- 坐标列表要包含两个坐标，与矩形相似
- outline参数指定矩形使用的轮廓颜色，fill参数指定矩椭圆内部使用的填充颜色

line

draw.line(xy, options)

- 在坐标列表中坐标之间首尾进行连线，不闭合
- 坐标列表是向这样序列对象：[(x, y), …]和[x, y, …]，至少包含两个坐标
- fill参数指定其使用的颜色
- width选项，指定线的宽度，以像素为单位，注意：线的连接和接头处处理的不好，因此当线过宽，该函数绘制的多边形看起来不是很好看

pieslice

draw.pieslice(xy, start, end, options)

- 和arc类似，只不过是绘制的是一个扇形
- outline参数指定矩形使用的轮廓颜色，fill参数指定扇形内部使用的填充颜色

point

draw.point(xy, options)

- 由给定的坐标绘制像素点
- 参数为一个像素列表：[(x, y), …] 或者[x, y, …]
- fill参数指定其使用的填充颜色

polygon

draw.polygon(xy, options)

- 绘制一个多边形
- 多边形轮廓包含了给定坐标多个直线段，该坐标围成一个闭合曲线
- 坐标列表是像这样的序列对象：[(x, y), …] 或者[x, y, …],至少包含三个坐标
- outline参数指定矩形使用的轮廓颜色，fill参数指定多边形内部使用的填充颜色

rectangle draw.rectangle(box, options)

- 绘制一个矩形
- box应该是一个序列对象，像这样[(x, y), (x, y)] 或者[x, y, x, y]，包含两个坐标
- 注意：当矩形不被填充的时候，第二个坐标对定义刚好在矩形外部的点
- outline参数指定矩形使用的轮廓颜色，fill参数指定矩形内部使用的填充颜色

text

draw.text(position, string, options)

- 在给定的位置绘制字符串，这个位置给出了文本字符串的左上角的位置
- font选项被用于指定要使用的字体，它应该是ImageFont类的实例，一般来讲，使用ImageFont模块的load方法从文件中载入

textsize

draw.textsize(string,font,options) ⇒ (width, height)

- 返回给定文本字符串的大小(w,h)形式的元组
- font选项被用于指定要使用的字体，它应该是ImageFont类的实例，一般来讲，使用ImageFont模块的load方法从文件中载入

选项

outline outline 整型或者元组

fill

fill 整型或者元组

font

font ImageFont 实例对象

兼容性

Draw类包含了一个构造器和一些仅用于向后兼容的方法，为了保证正常工作，你应该要么使用绘图图元的选项，要么使用这些下面这些方法。不要混合使用新的和旧的调用约定习惯。在IronPIL中，这些兼容性的方法不被支持。

ImageDraw

ImageDraw(image) ⇒ Draw instance

- 不推荐使用，和Draw相同

setink draw.setink(ink)

- 不推荐使用，设置后续操作的绘制填充操作的颜色

setfill

draw.setfill(mode)

- 不推荐使用（已经被移除），设置填充模式
- 如果模式为0，则描绘形状（多边形和矩形）的轮廓，如果模式为1，它们将会被填充

setfont

draw.setfont(font)

- 不推荐使用，为Text方法设置默认的字体
- font参数应该是ImageFont实例，一般来讲，使用ImageFont模块的load方法载入文件

ImageEnhance模块

The ImageEnhance Module

ImageEnhance模块包含了很多的类，用于图像的增强效果。

Example

改变图像的清晰度

```
import ImageEnhance

enhancer = ImageEnhance.Sharpness(image)
for i in range(8):
    factor = i / 4.0
    enhancer.enhance(factor).show("Sharpness %f" % factor)
```

接口

所有的enhancement类都实现了一个公共的接口，包含了单个方法：

enhancer.enhance(factor) ⇒ image

返回一个增强的图像，factor因子是一个浮点型数值，控制其增强效果。Factor为1.0，总是返回原图像的一个副本，小于1.0，是减弱效果，大于1.0是增强效果

Color类

color enhancement类用于调整图像的颜色平衡，这种方式类似于电视的颜色控制

ImageEnhance.Color(image) ⇒ Color enhancer instance

为一个图像创建一个增强对象调整颜色。Factor为0.0，则返回一个黑白图像，Factor为1.0,则返回原图像。

Brightness类

brightness enhancement类用于控制图像的亮度。

ImageEnhance.Brightness(image) ⇒ Brightness enhancer instance

为一个图像创建一个增强对象调整图像亮度。Factor为0.0返回一个黑色图像，Factor为1.0返回原始图像。

Contrast类

contrast enhancement类用于控制图像的对比度，类似电视上对比度的控制。
ImageEnhance.Contrast(image) ⇒ Contrast enhancer instance

为一个图像创建一个增强的对象调整图像的对比度，Factor为0.0返回一个实体灰色图像，Factor为1.0返回原始图像。

Sharpness类

sharpness enhancement类用于控制图像的清晰度。

ImageEnhance.Sharpness(image) ⇒ Sharpness enhancer instance

为一个图像创建一个增强图像调整图像的清晰度，Factor为0.0，返回一个不清晰的图像，Factor为1.0返回原始图像。

ImageFile模块

ImageFile模块

ImageFile模块提供函数用于图像打开和保存的功能。

此外，它还提供了一个Parser类，可以一片一片对图像进行解码，(例如：当从网络连接中接收数据)，这个类实现了和标准sgmllib和xmllib模块相同的用户接口。

示例

解析一个图像

```
import ImageFile
fp = open("lena.pgm", "rb")
p = ImageFile.Parser()
while 1:
    s = fp.read(1024)
    if not s:
        break
    p.feed(s)
im = p.close()
im.save("copy.jpg")
```

函数

Parser

ImageFile.Parser() ⇒ Parser instance

- 创建一个解释器对象，解释器不能重复使用

方法

feed

parser.feed(data)

- 将字符串数据喂给解释器，该方法可能会引发IOError异常

close

parser.close() ⇒ image or None

- 告知解释器完成解码，如果解释器成功解码将返回一个Image对象，否则返回一个IOError异常

注意：如果文件不能被识别，解释器将会在close方法中引发一个异常，如果文件可以被识别，但是不能解码（例如：数据损坏，或者使用一个不被支持的压缩方法），解释器也会引发一个IOError异常

ImageFileIO模块

ImageFileIO模块

ImageFileIO模块可被用于从一个套接字读取图像，或者其他流装置。

该模块已经不推荐使用了，相反，使用ImageFile模块中Parser类。

函数

ImageFileIO.ImageFileIO(stream)

- 将缓冲添加到流文件对象中，为了提供Image.open方法所需的seek和tell方法，这个流对象必须实现read和close方法

ImageFilter模块

ImageFilter模块

ImageFiter模块包含一组预定义的过滤器的定义，用于Image类的过滤器方法。

示例

过滤一个图像

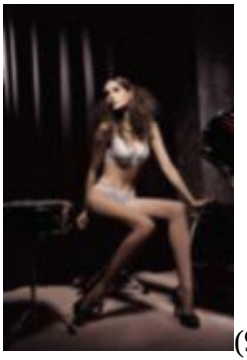
```
import ImageFilter
im1 = im.filter(ImageFilter.BLUR)
im2 = im.filter(ImageFilter.MinFilter(3))
im3 = im.filter(ImageFilter.MinFilter) # same as MinFilter(3)
```

过滤器

当前版本库，提供了下列一组预定义图像增强过滤器(类似PS里面的滤镜)：

BLUR(模糊), CONTOUR（轮廓V等高线）**, DETAIL, EDGE_ENHANCE, EDGE_ENHANCE_MORE, EMBOSS(浮雕), FIND_EDGES, SMOOTH, SMOOTH_MORE, and SHARPEN**（锐化）

效果图依次如下： (0):  (1):  (2):  (3):  (4):  (5):  (6):  (7):

 (8):  (9):  (10): 

Kernel

Kernel(size, kernel, scale=None, offset=0)

- 卷积是图像处理常用的方法,给定输入图像,在输出图像中每一个像素是输入图像中一个小区域中像素的加权平均,其中权值由一个函数定义,这个函数称为卷积核, 比如说卷积公式：R(u，v)=∑∑G（u-i，v-j）f（i，j），其中f为输入，G为卷积核， 卷积核为系数矩阵
- 对图像像素进行卷积遍历（即截取和卷积核同等大小的像素矩阵进行卷积运算），每一个输出像素都是一定区域像素按一定权重组合计算出的结果
- kernel为序列对象包含9/25个整型或者浮点型值，作为权重

RankFilter RankFilter(size, rank)

- 创建给定大小的rank过滤器，针对输入图像的每个像素，等级过滤器根据像素值对(size,size)环境中的像素进行排序，将排序好的值复制输出图像中

MinFilter

MinFilter(size=3)

- 创建给定大小的min过滤器，针对输入图像的每个像素，这个过滤器复制（size,size）环境中的最小的值复制到输出图像中

MedianFilter

MedianFilter(size=3)

- 与MinFilter类似，只不过是取中间值

MaxFilter MaxFilter(size=3)

- 与MaxFilter类似，只不过取最大值

ModeFilter

ModeFilter(size=3)

- 创建给定大小的mode过滤器，针对输入图像的每个像素,这个过滤器复制（size,size）出现最多的像素值到输出图像中
- 如果出现所有的像素值出现的次数不超过一次的，那么将使用原始的像素值

ImageFont模块

ImageFont

ImageFont模块定义了与其相同名字的类，这个类的实例存储着bitmap字体，被用于ImageDraw类的text方法中。

PIL使用自己的字体文件格式来存储位图字体，你可以使用pilfont工具来将BDF和PCF字体描述符（X window字体格式）转化为这种格式。

1.1.4版本开始，PIL可以支持TrueType和OpenType字体，（和其他字体格式一样被FreeType库支持）。早些版本中，TrueType支持仅仅作为Toolkit软件包的一部分提供。

示例

```
import ImageFont, ImageDraw

draw = ImageDraw.Draw(image)

# use a bitmap font
font = ImageFont.load("arial.pil")
draw.text((10, 10), "hello", font=font)

# use a truetype font
font = ImageFont.truetype("arial.ttf", 15)

draw.text((10, 25), "world", font=font)
```

函数

load

ImageFont.load(file) ⇒ Font instance

- 从给定的文件中载入一个字体，并且返回相应的字体对象
- 如果出错，引发一个IOError异常

load_path

ImageFont.load_path(file) ⇒ Font instance

- 作用与load相同，但是如果没有在当前目录中找到，就在sys.path路径中查找。

truetype

ImageFont.truetype(file, size) ⇒ Font instance

- 载入一个TrueType和OpenType字体文件，创建一个字体对象，这个函数从给定的文件中载入一个字体，并且创建一个指定字体大小的字体对象

- 在Windows中，如果给定的文件名不存在，加载器也会在Windows字体目录中搜索

- 这个函数需要_imageinft服务
- 在1.1.5版本中，新增encoding参数，指定编码；常见的编码有：“unic” (Unicode), “symb” (Microsoft Symbol), “ADOB” (Adobe Standard), “ADBE” (Adobe Expert), and “armn” (Apple Roman)。

下面这个例子使用微软的Symbol字体绘制字符，使用'symb'编码，字符串的范围在0xF000和0xF0FF之间：

```
font = ImageFont.truetype("symbol.ttf", 16, encoding="symb")
draw.text((0, 0), unichr(0xF000 + 0xAA))
```

load_default

ImageFont.load_default() ⇒ Font instance

- 载入一个默认字体

方法

字体对象必须实现以下方法，这些方法供ImageDraw绘制层使用。

getsize

font.getsize(text) ⇒ (width, height)

- 返回给定文本字符串的尺寸（长度和宽度）,为一个2元元组。

getmask

font.getmask(text, mode=”“) ⇒ Image object

- 返回文本的位图，位图应该是内部PIL存储器内00存实例（由Image.core接口模块定义）
- 如果使用字体使用抗锯齿，位图应该是模式'L',并且最大值为255，或者是模式'1'
- 在1.1.5版本中，新增可选mode参数,被用于图形驱动来指示驱动更喜欢的模式，如果为空，渲染器返回任意一个模式，mode为一个字符串

ImageGrab模块

The ImageGrab Module

ImageGrab模块被用于复制屏幕的内容和剪切板的内容到一个PIL内存中的图像。 当前版本仅在Windows上实现。

函数

grab

ImageGrab.grab(bbox) ⇒ image

- 捕获屏幕的截图，并返回一个RGB图像，bbox参数用于指定截取屏幕的边框
- 不指定bbox,则截取整个屏幕

grabclipboard

ImageGrab.grabclipboard() ⇒ image or list of strings or None

- 获取剪切板内容的快照，返回一个图像对象和文件名列表
- 如果剪切板不包含任何图像数据，将返回None

ImageMath模块

The ImageMath Module

ImageMath 用于创建计算图像表达式，该模块只提供了一个eval函数，接受一个表达式字符串和一个或者多个图像对象。

示例：

```
im1 = Image.open("image1.jpg")
im2 = Image.open("image2.jpg")

out = ImageMath.eval("convert(min(a, b), 'L')", a=im1, b=im2)
out.save("result.png")
```

函数

eval

eval(expression, environment) ⇒ image or value

- expression参数是使用标准Python表达式语法的字符串。 除了标准操作符之外，还可以使用下面描述的函数
- environment参数是一个将图像名称映射到Image实例的字典,您可以使用一个或多个关键字参数，而不是字典，如上例所示, 注意:名称必须是有效的Python标识符

- 在当前版本中，ImageMath仅支持单层图像;要处理多波段图像，请使用分割和合并功能

- 返回图像，整数值，浮点值或像素元组

表达式语法

表达式是标准Python表达式，但它们在非标准环境中进行计算。 您可以照常使用PIL方法，此外还有下面的一组运算符和函数。

标准操作符 可以使用标准算术运算符进行加法（+），减法（ - ），乘法（*）和除法（\/）。

模块还支持一元负（ - ），模（％）和幂（**）运算符。

注意，根据需要，所有操作都使用32位整数或32位浮点值完成。 例如，如果添加两个8位图像，结果将是一个32位整数图像。 如果向8位图像添加浮点常量，则结果将是32位浮点图像。

您可以使用下面描述的convert，float和int函数强制转换。

位运算符 该模块还提供位操作。 这包括and（＆），or（|）和exclusive or（^）。

你也可以取反（～）所有像素位。请注意，在应用按位操作之前，操作数将转换为32位有符号整数。

这意味着，如果你反转一个普通的灰度图像，你会得到负值。 您可以使用和（和）运算符来屏蔽不需要的位。

按位运算符不适用于浮点型图像。

逻辑运算符

像and，or这样的，不是工作于整个图像，而是工作于单个像素的逻辑运算符。

一个空图像（所有像素为零）被视为假。 所有其他图像都视为true。

请注意，and和or 返回最后一个求值的操作数，而不总是返回一个布尔值。

内建函数

这些函数应用于每个单个像素上。

abs(image) 绝对值

convert(image, mode) 将图像转换为给定模式。 模式必须作为字符串常量给出

float(image) 将图像转换为32位浮点。 这相当于convert（image，“F”）

int(image) 将图像转换为32位整数，这相当于convert（image，“I”）

Note that 1-bit and 8-bit images are automatically converted to 32-bit integers if necessary to get a correct result.

max(image1, image2) 最大值 min(image1, image2) 最小值

ImageOps模块

The ImageOps Module

ImageOps模块包含多个“现成”图像处理操作。 这个模块有点实验性，大多数操作函数方法只适用于L和RGB图像。

函数

autocontrast

ImageOps.autocontrast(image, cutoff=0) ⇒ image

- 最大化（标准化）图像对比度
- 该函数计算输入图像的直方图，从直方图中去除最亮和最暗像素的截止百分比，并重新映射图像，使得剩余最黑的像素变为黑色（0），最亮的变为白色（255）
- cutoff参数用来指定被截断的百分比

- 效果图如下：

原图：  效果图： 

colorize

ImageOps.colorize(image, black, white) ⇒ image

- 对灰度图像进行着色处理
- white和black参数指定为RGB元组或者颜色名称，这个函数计算一个颜色楔将所有黑色像素映射到第一颜色上，将所用白色像素映射到第二个颜色上
- 效果图如下：

原图：  着色图： 

deform

ImageOps.deform(image, deformer, filter=Image.BILINEAR) ⇒ image

- 使用给定的变形器对象变形图像。
- 变形器应该提供getmesh方法，该方法返回适合于图像变换方法的MESH列表
- 还不清楚如何使用

equalize

ImageOps.equalize(image) ⇒ image

- 均衡图像直方图
- 该函数对输入图像应用一个非线性映射，以便创建一个灰度值均匀分布输出图像
- 效果图如下：

原图：  均衡图： 

expand ImageOps.expand(image, border=0, fill=0) ⇒ image

- 在所有边缘进行扩展边框，与crop相反，也就是加边框
- fill参数填充色
- 效果图如下：

原图：  边框图： 

fit

ImageOps.fit(image, size, method, bleed, centering) ⇒ image

- 返回裁剪为所需宽高比和大小的图片的版本
- size参数是请求的输出大小（以像素为单位），以（width，height）元组给出
- method参数是指定使用的重采样方法。默认值为Image.NEAREST
- bleed(渗出)参数允许您删除图像外部的边框,该值为十进制百分比（对于百分之一，使用0.01）,默认值为0（无边框）
- centering参数用于控制裁剪位置，（0.5,0.5）是中心裁剪
- 效果图如下：

原图：  效果图： 

crop ImageOps.crop(image, border=0) ⇒ image

- 移除所有边缘的像素，也就是裁边
- border为指定裁边的宽度
- 效果图如下：原图：  裁边： 

flip ImageOps.flip(image) ⇒ image

- 垂直方向上翻转图像
- 效果图如下：原图：  垂直翻转： 

grayscale ImageOps.grayscale(image) ⇒ image

- 将图像转化为灰度
- 效果图如下：

原图：  灰度图： 

invert

ImageOps.invert(image) ⇒ image

- 反转（取反）图像
- 效果图如下：

原图：  反转图： 

mirror ImageOps.mirror(image) ⇒ image

- 水平翻转图像（从左往右）
- 效果图如下：原图：  水平翻转： 

posterize

ImageOps.posterize(image, bits) ⇒ image

- 色调分离
- 减少每个颜色通道上的颜色位
- 对于'RGB图像而言，bit为8,返回原图像，为0，返回黑色图像
- 效果图如下：原图：  色调分离： 

solarize

ImageOps.solarize(image, threshold=128) ⇒ image

- 过度曝光
- 将高于某一域值的所有像素进行反转

- 效果图如下：

原图：  曝光图： 

ImagePalette模块

The ImagePalette Module

FIXME： 当前版本并不完全匹配当前的实现，当前，最安全的方式是使用putpalette方法将一个调色板附加到图像上。

示例

将一个调色板附加到图像上

```
palette = []
for i in range(256):
    palette.extend((i, i, i)) # grayscale wedge
assert len(palette) == 768
im.putpalette(palette)

#当前还没被支持
import ImagePalette

palette = ImagePalette.ImagePalette("RGB")
palette.putdata(...)

im.putpalette(palette)
```

使用resize和convert方法来调色板的内容

```
assert im.mode == "P"

lut = im.resize((256, 1))
lut.putdata(range(256))
lut = lut.convert("RGB").getdata()
```

类

ImagePalette

ImagePalette.ImagePalette(mode=”RGB”) ⇒ palette instance

- 这个构造器创建一个调色板，从'P'模式映射到给定的模式，这个调色板被初始化为线性灰度梯度

ImagePath模块

The ImagePath Module

ImagePath（图像路径）模块被用于存储和操作二维向量数据，Path对象可以被传递给ImageDraw模块中的方法。

函数

Path

ImagePath.Path(coordinates) ⇒ Path instance

- 创建一个坐标对象，坐标列表是一个序列对象：[(x, y), …] 或者 [x, y, …]
- 你也可从其他path对象创建一个path对象
- path对象实现了实现了Python序列接口的大部分实现，其行为类似于一个二元元组列表，因此你可以len函数、访问项元素、切片， 然而当前版本并不支持切片赋值，项元素和切片删除

Path方法

compact

p.compact(distance=2) ⇒ count

- 通过移除彼此靠近的点来压缩路径，这个方法是在原路径对象上修改的，返回路径剩余点的计数
- distance距离被测量为曼哈顿距离，Distance = |x2-x1|+|y2-y1|，默认为两个像素

getbbox

p.getbbox() ⇒ 4-tuple

- 获取路径的边框

map

p.map(function)

- 通过一个函数映射路径

tolist

- 将路径转换为一个Python列表，[(x, y), …]
- flat（0 or 1）选项控制列表返回的形式，前者是一个二元元组列表[(x, y), …]，后者为列表（[x, y, …]）

transform

p.transform(matrix)

- 使用仿射变换在原路径上进行转换

- 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)
- matrix参数为一个6元元组(a, b, c, d, e, f)
- 每个点像下面这样被映射：

```
xOut = xIn * a + yIn * b + c
yOut = xIn * d + yIn * e + f
```

ImageQt模块

The ImageQt Module

（在版本中1.1.6中的新功能）Image Qt模块提供了从PIL images中创建PyQt4 QImage对象的支持。

ImageSequence模块

The ImageSequence Module

ImageSequnce模块包含一个包装类，能够使你能够遍历图像序列帧。

从一个动画中抽取帧

```
import Image,ImageSequence
im=Image.open('animation.fli')

index=1
for frame in ImageSequence.Iterator(im):
    frame.save('frame%d.png'%index)
    index=index+1
```

函数

- 创建一个迭代器对象，使你能够遍历序列中所有的帧

Methods

这个Iterator类实现[]操作符：

操作符：**[]**

- 你可以调用这个[]操作，完成元素的访问，这类似数组元素的访问（即：array[6]）
- 当迭代结束后，会引发一个IndexError异常

ImageStat模块

The ImageStat Module

ImageStat模块计算一个图像和一个图像区域的全局统计数据。

函数

Stat

ImageStat.Stat(image,mask) ⇒ Stat instance

- 计算给定图像的统计信息
- 如果包括一个蒙版，则统计中仅包括由该蒙版覆盖的区域 ImageStat.Stat(list) ⇒ Stat instance
- 与上述相同，但计算先前计算的直方图的统计数据

属性

下面的属性包含一个元素序列，图像的每个图层都有一个，所有属性都被延迟计算; 如果你不需要一个值，它不会被计算。

extrema

stat.extrema

- 获取图像中每个波段的最小/最大值

count

stat.count

- 获取像素的总数

sum stat.sum

- 获取所有像素的总和

sum2

stat.sum2

- 所有像素的平方和

mean

stat.mean

- 像素平均值

median

stat.median

- 像素中值

rms

stat.rms

- 均方根

stddev

stat.stddev

- 标准的偏差

var

stat.var

- 方差

ImageTk模块

The ImageTk Module

ImageTk模块提供这样的支持：从PIL images中创建和修改Tkinter BitmapImage和PhotoImage对象。

ImageWin模块

The ImageWin Module

这个ImageWin模块提供了在windws上创建和显示图像的支持。

ImageWin可以与PythonWin和其他用户界面工具包一起使用，提供对Windows设备上下文或窗口句柄的访问。 例如，Tkinter通过winfo_id方法使窗口句柄可用：

```
dib = ImageWin.Dib(...)

hwnd = ImageWin.HWND(widget.winfo_id())
dib.draw(hwnd, xy)
```

PSDraw模块

The PSDraw Module

PSDraw模块提供了简单地打印输出功能,支持Postscript printers。你可以通过这个模块打印文本，图形，和图像。 PostScript是一种编程语言，最适用于列印图像和文字（无论是在纸、胶片或非物质的CRT都可）,现今的行话讲，它是页面描述语言

类

PSDraw

PSDraw.PSDraw(file) ⇒ PSDraw instance

- 设置打印到的给定文件，如果忽略此参数，默认为sys.stdout

PSDraw方法

begin

ps.begin_document()

- 开始打印文档

end

ps.end_document()

- 结束打印

line

ps.line(from, to)

- 在两点之间画一条线。 坐标以Postscript的点坐标形式给出（每英寸72点，（0,0）是页面的左下角）

rectangle

ps.rectangle(box)

- 绘制一个矩形

text

ps.text(position, text)

ps.text(position, text, alignment)

- 在给定的位置绘制文本，在调用这个方法前必须指定字体

image

ps.image(box, image, dpi=None)

- 绘制一个图像，在给定的边框中居中

setfont

ps.setfont(font, size)

- 指定要使用的文字，font参数为Postscript字体名

setink

ps.setink(ink)

- 选择后续操作使用的像素值

setfill

ps.setfill(onoff)

- 选择后续操作矩形是绘制填充矩形还是仅绘制轮廓