将数组保存成图像的多种方法 （包含 scipy.misc.toimage）

原创

ciky奇 于 2018-07-05 09:42:00 发布 20302 收藏 40
分类专栏： 计算机视觉 文章标签： scipy.misc.toimage plt.savefig write PIL

本文链接：https://blog.csdn.net/c20081052/article/details/80917841

本文本打算写toimage这个函数的，发现网上有很多将数组保存为图像的方法，故一起整理在一起。其他方法是参考了：
Numpy数组保存为图片

Numpy数组类型的矩阵，如何将它作为图像写入磁盘？任何格式的图像都行(PNG，JPEG，BMP ...)

最佳的解决方法

可以使用scipy.misc,代码如下：

```
import scipy.misc
scipy.misc.imsave('outfile.jpg', image_array)
```

这里顺带说下**scipy.misc.toimage**函数
函数原来形式如下：
scipy.misc.toimage(*args,**kwds)这个函数将在scipy 1.2.0版本中删除，将会用Pillow的Image.formarray替代 。
*args：输入的array；
**kwds：输入的关键字；

功能是输入一个numpy数组array,输出一个PIL图像。
这个功能只有在PIL安装后才能使用。输出的PIL图的模式依赖于array的形状和pal, mode这些关键字。

对于2-D arrays, 如果pal 是有效的（N, 3） byte类的且其值在（0, 255）间的RGB值, 那么mode='P', 否则mode='L', 除非mode给出为'F'或'I', 这种情况下将会创建浮点数 和/或 整数数组。
对于3-D arrays, 'channel_axis'参数指示数组的哪个维度会保存通道数据。

对于3-D arrays, 如果某个维度是3, 默认mode是RGB或YCbCr。
numpy array 必须是2维或3维的。

函数可写成scipy.misc.toimage（*ARR, 高=255, 低= 0, 的Cmin =无, Cmax为无, PAL =无, 模式=无, channel_axis =无*）

上面的scipy版本会标准化所有图像，以便min(数据)变成黑色，max(数据)变成白色。如果数据应该是精确的灰度级或准确的RGB通道，则解决方案为：

```
import scipy.misc
scipy.misc.toimage(image_array,
 cmin= 0 . 0, cmax=...).save(
```

（ SciPy 中包含一些用于输入和输出的实用模块。下面介绍其中两个模块：io 和misc。 以图像形式保存数组

因为我们需要对图像进行操作，并且需要使用数组对象来做运算，所以将数组直接保存为图像文件非常有用。

imsave() 函数可以从scipy.misc 模块中载入。要将数组im 保存到文件中，可以使用下面的命 令：

from scipy.misc import imsave imsave('test.jpg',im) ）

我在程序中的一个实例是这样的：

```
for i in range(20):
    image_array=mnist.train.images[i,:]
    image_array=image_array.reshape(28,28)
    filename=save_dir+'mnist_train_%d.jpg' % i
    scipy.misc.toimage(image_array,cmin=0.0,cmax=1.0).save(filename)
```

第二种解决办法

使用PIL。

给定一个numpy数组"A"：

```
from PIL import Image
im = Image.fromarray(A)
im.save(
```

你可以用几乎任何你想要的格式来替换"jpeg"。有关格式详见here更多细节

第三种办法

纯Python(2& 3)，没有第三方依赖关系的代码片段。

此函数写入压缩的真彩色(每个像素4个字节)RGBA PNG。

```
def write_png(buf, width, height):
    buf: must be bytes or a bytearray in Python3.x,
    a regular string in Python2.x.
    import zlib, struct
    # reverse the vertical line order and add
    width_byte_4 = width * 4
    raw_data = b''.join(b'\x00' + buf[span:span + width_byte_4]
        for span in range(height
    def png_pack(png_tag, data):
        chunk_head = png_tag + data
        return (struct.pack("!I", len(data)) +
            chunk_head +
            struct.pack( "!I", 0xFFFFFFFF &
            zlib.crc32(chunk_head)))
    return b''.join([
        b'\x89PNG\r\n\x1a\n',
```

```
png_pack(
png_pack( b'IDAT',
zlib.compress(raw_data,
png_pack(
```

...数据应直接写入以二进制打开的文件，如下所示：

```
data =
with open(buf,
write_png(
'my_image.png',
```

> Original source
>
> 另见：Rust Port from this question.
>
> 使用示例感谢@Evgeni Sergeev：https://stackoverflow.com/a/21034111/432509

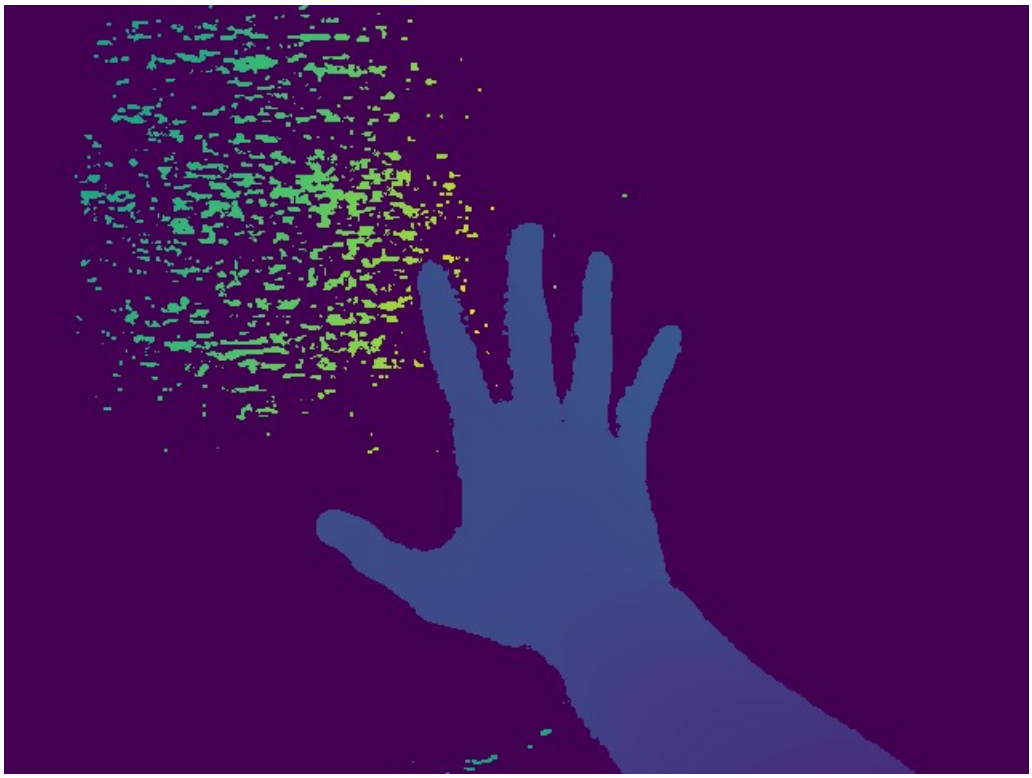
第四种办法

用matplotlib：

```
import
matplotlib
matplotlib.image.imsave(
```

适用于matplotlib 1.3.1，不确定更低的版本是否有效。文档：

```
Arguments:
*fname*:
A string
containing
a path to
format is
```

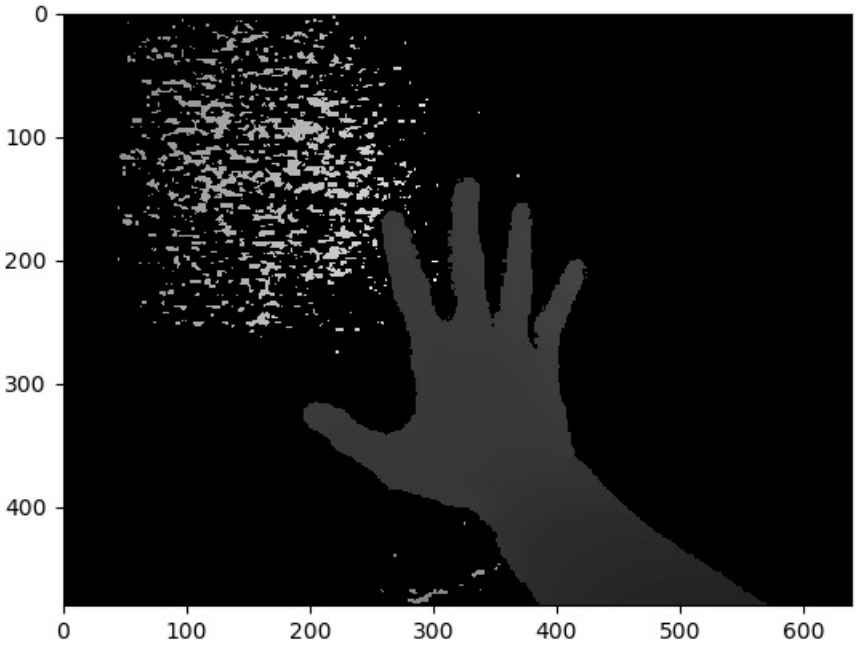


第五种办法

如果使用matplotlib，也可以这样做：

```
import matplotlib.pyplot as plt
plt.imshow(
```

这将保存plot(而不是图像本身)。



第6种办法

python的opencv(http://docs.opencv.org/trunk/doc/py_tutorials/py_tutorials.html)。

```
import cv2
import numpy as np
cv2.imwrite(
```

如果你需要做更多的处理，而不是保存，这个库比较有用。

参考文献

- Saving a Numpy array as an image