课堂实验一：Python数字图像基本处理

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、实验内容及步骤

1．利用 imread( )函数读取一幅图像，假设其名为 flower.tif，存入一个

数组中；

2．利用 whos 命令提取该读入图像 flower.tif 的基本信息；

3．利用 imshow()函数来显示这幅图像；

4．利用 imfinfo 函数来获取图像文件的压缩，颜色等等其他的详细信息；

5．利用 imwrite()函数来压缩这幅图象，将其保存为一幅压缩了像素的

jpg 文件,设为 flower.jpg；语法：imwrite(原图像，新图像，‘quality’,q), q 取

0-100。6．同样利用 imwrite()函数将最初读入的 tif 图象另存为一幅 bmp 图像，

设为 flower.bmp。

7．用 imread()读入图像：Lenna.jpg 和 camema.jpg；

8．用 imfinfo()获取图像 Lenna.jpg 和 camema.jpg 的大小；

9．用 figure,imshow()分别将 Lenna.jpg 和 camema.jpg 显示出来，观察

两幅图像的质量。

10．用 im2bw 将一幅灰度图像转化为二值图像，并且用 imshow 显示

出来观察图像的特征。

11．将每一步的函数执行语句拷贝下来，写入实验报告，并且将得到

第 3、9、10 步得到的图像效果拷贝下来。

三、实验代码及结果

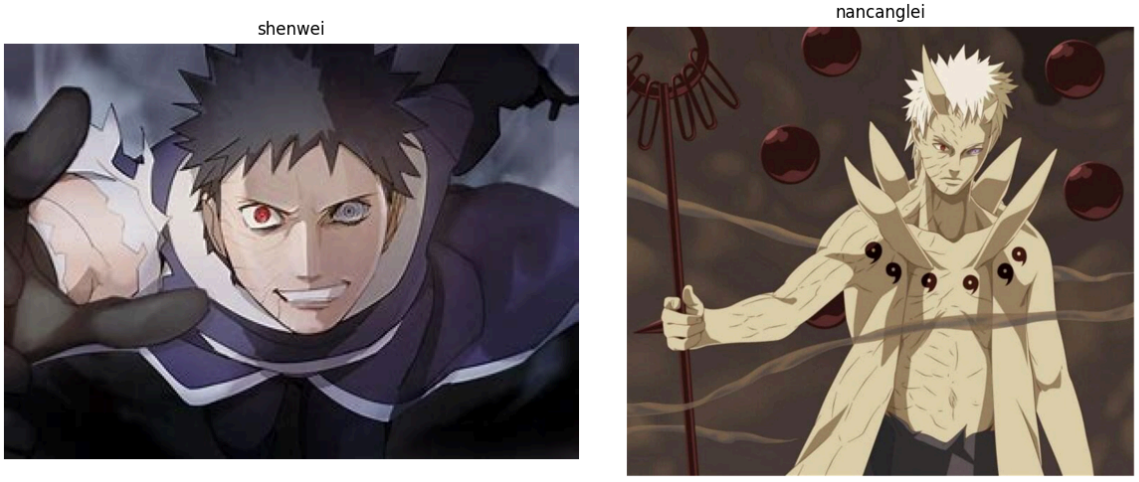
```
import cv2
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt

# 1. 读取图像
flower = cv2.imread('obtu.png')  # 默认BGR格式
```

```python
flower_rgb = cv2.cvtColor(flower, cv2.COLOR_BGR2RGB)  # 转换为RGB显示

# 2. 提取基本信息
print("Image shape:", flower.shape)  # (高度, 宽度, 通道数)
print("Data type:", flower.dtype)    # 数据类型
print("Number of pixels:", flower.size)  # 像素总数

# 3. 显示原始图像
plt.figure(figsize=(8, 6))
plt.imshow(flower_rgb)
plt.title('Original obtu')
plt.axis('off')
plt.show()

# 4. 获取文件详细信息（使用Pillow）
with Image.open('obtu.png') as img:
    info = {
        'format': img.format,
        'mode': img.mode,
        'size': img.size,
        'filename': img.filename
    }
    print("\nImage details from Pillow:")
    for key, value in info.items():
        print(f"{key}: {value}")

# 5. 压缩为JPEG（质量75）
cv2.imwrite('obtu.jpg', flower, [int(cv2.IMWRITE_JPEG_QUALITY), 75])

# 6. 另存为BMP
cv2.imwrite('obtu.bmp', flower)

# 7. 读取其他图像
lenna = cv2.imread('shenwei.jpg')
camema = cv2.imread('nancanglei.jpg')

# 转换为RGB
shenwei_rgb = cv2.cvtColor(lenna, cv2.COLOR_BGR2RGB)
nancanglei_rgb = cv2.cvtColor(camema, cv2.COLOR_BGR2RGB)

# 8. 获取图像尺寸
print("\nLenna size:", lenna.shape[:2][::-1])  # (宽度, 高度)
print("Camema size:", camema.shape[:2][::-1])

# 9. 显示两幅图像
plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.imshow(shenwei_rgb)
plt.title('shenwei')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(nancanglei_rgb)
plt.title('nancanglei')
plt.axis('off')
```

```
plt.tight_layout()
plt.show()

# 10. 灰度转二值图像（以Lenna为例）
# 先转换为灰度
gray_lenna = cv2.cvtColor(lenna, cv2.COLOR_BGR2GRAY)

# 方法1：固定阈值（127）
_, binary_fixed = cv2.threshold(gray_lenna, 127, 255, cv2.THRESH_BINARY)

# 方法2：Otsu's阈值（自动计算最佳阈值）
_, binary_otsu = cv2.threshold(gray_lenna, 0, 255, cv2.THRESH_BINARY +
cv2.THRESH_OTSU)

# 显示结果
plt.figure(figsize=(15, 5))

plt.subplot(1, 3, 1)
plt.imshow(gray_lenna, cmap='gray')
plt.title('Original Gray')
plt.axis('off')

plt.subplot(1, 3, 2)
plt.imshow(binary_fixed, cmap='gray')
plt.title('Binary (Threshold=127)')
plt.axis('off')

plt.subplot(1, 3, 3)
plt.imshow(binary_otsu, cmap='gray')
plt.title('Binary (Otsu)')
plt.axis('off')

plt.tight_layout()
plt.show()
```

显示图片imshow

显示两张图片并对比质量



灰度图转化为二值图像

四、思考题

(1) Python软件的特点

Python是一种高级编程语言，具有以下特点：

1. 简单易学：语法简洁清晰，接近自然语言，适合初学者。
2. 跨平台性：可在Windows、Linux、Mac OS等多种操作系统上运行。
3. 丰富的库支持：拥有大量标准库和第三方库（如NumPy、Pandas、OpenCV等），支持多种功能开发。
4. 解释型语言：代码逐行执行，无需编译，调试方便。
5. 动态类型：变量无需声明类型，运行时自动推断。
6. 面向对象：支持面向对象编程（OOP），同时兼容过程式编程。
7. 强大的社区支持：开源社区活跃，文档丰富，问题容易解决。
8. 可扩展性：可通过C/C++扩展性能，或与其他语言集成。

---

(2) Python支持的图像文件格式

Python通过OpenCV、Pillow（PIL）等库支持多种图像格式，包括：

• 常见格式：

• BMP（无损，未压缩）

• PNG（无损，支持透明度）

• JPEG/JPG（有损压缩，适合照片）

• GIF（动画，支持透明度）

• TIFF（高精度，支持多页）

• WebP（现代格式，高压缩率）

• RAW（相机原始数据）

• 其他格式：

• TGA、DDS、EXR（专业图像处理）

• PDF（需额外库如PyMuPDF）

---

(3) `imread` 函数的用途、格式及图像性质

用途

`imread` 用于从文件加载图像到内存，返回一个NumPy数组（OpenCV）或PIL图像对象（Pillow）。

格式及性质

库名	函数	返回值类型	图像性质
OpenCV	`cv2.imread()`	`numpy.ndarray`	默认BGR格式（非RGB），像素值为0-255的uint8类型，支持多通道（灰度/彩色）。
Pillow	`Image.open()`	`PIL.Image`	RGB格式（默认），像素值为0-255的uint8类型，支持透明度（RGBA）。

示例代码

```
import cv2
from PIL import Image

# OpenCV读取（BGR格式）
img_cv = cv2.imread("image.jpg")  # shape=(H, W, C), dtype=uint8

# Pillow读取（RGB格式）
img_pil = Image.open("image.jpg")  # 需转换为NumPy数组才能计算
img_pil_np = np.array(img_pil)     # 转换为numpy数组后可用
```

---

(4) 为什么`I = imread('lena.bmp')`得到的图像`I`不能直接进行算术运算?
原因分析

1. 数据类型问题：
   • 如果`imread`来自OpenCV，默认返回`uint8`类型（0-255整数），直接进行算术运算可能导致溢出或精度丢失。
   • 例如：`I + 10`会溢出（255 + 10 = 255，而非265）。
2. 未显式转换类型：
   • 需先将图像转换为浮点型（`float32`）再进行运算，例如归一化或滤波：

   ```
   I_float = I.astype(np.float32) / 255.0  # 归一化到[0,1]
   ```

3. Pillow与OpenCV的区别：
   • Pillow的`Image`对象需先转换为NumPy数组才能运算：

   ```
   from PIL import Image
   I_pil = Image.open("lena.bmp")
   I_np = np.array(I_pil)  # 转换为numpy数组后才能运算
   ```

解决方法

```
import cv2
import numpy as np

I = cv2.imread("lena.bmp")  # uint8类型
I_float = I.astype(np.float32)  # 转换为float32

# 示例运算（如高斯模糊后的加法）
blurred = cv2.GaussianBlur(I_float, (5, 5), 0)
result = I_float + blurred  # 需确保类型一致
```

关键点
• OpenCV的`imread`默认返回`uint8`，需显式转换类型才能进行数学运算。

• Pillow的`Image`对象需先转NumPy数组才能参与计算。

• 算术运算前建议归一化到`[0,1]`（浮点型）以避免溢出。