问题 1 图像处理软件包比较 (20 分)

Pillow库

支持的图像类型和格式:

- Pillow支持多种光栅文件格式，例如PNG, JPEG, PPM, GIF, TIFF, 和BMP等。

读取，显示和写入图像:

```
from PIL import Image

# 读取图像
# 使用`Image.open()`函数，通过传递图像文件的路径来读取图像。
image = Image.open('example.jpg')

# 显示图像
# 使用`show()`方法，此方法会调用系统默认的图像查看器来显示图像。
image.show()

# 保存图像
# 使用`save()`方法，并传递新的图像文件名和路径来保存图像。
image.save('new_example.png')
```

图像处理功能:

```
# 图像裁剪
# 使用`crop()`方法，并传递一个元组，该元组包含4个值：左，上，右和下，来定义裁剪区域。
cropped_image = image.crop((left, upper, right, lower))

# 图像缩放
# 使用`resize()`方法，并传递一个元组，该元组包含2个值：宽度和高度，来定义新图像的尺寸。
resized_image = image.resize((width, height))

# 图像旋转
# 使用`rotate()`方法，并传递一个角度值来旋转图像。
rotated_image = image.rotate(angle)
```

Scikit-image库

支持的图像类型和格式:

- Scikit-image支持常见的图像格式，如BMP, PNG, JPEG, 和TIFF等。

读取，显示和写入图像:

```
import skimage.io as io
import matplotlib.pyplot as plt

# 读取图像
# 使用`io.imread()`函数，通过传递图像文件的路径来读取图像。
image = io.imread('example.jpg')

# 显示图像
# 使用`io.imshow()`函数来显示图像，然后使用matplotlib的`plt.show()`来显示图像窗口。
io.imshow(image)
plt.show()

# 保存图像
# 使用`io.imsave()`函数，通过传递新的图像文件名和路径以及图像数据来保存图像。
io.imsave('new_example.png', image)
```

图像处理功能:

- Scikit-image包括用于图像分割、几何变换、颜色空间处理、分析、过滤、形态学、特征检测等多种图像处理算法.

```
from skimage import filters, color

# 转为灰度图像
# 使用`color.rgb2gray()`函数，通过传递彩色图像数据来获取灰度图像。
gray_image = color.rgb2gray(image)

# 应用Sobel边缘检测过滤器
# 使用`filters.sobel()`函数，通过传递灰度图像数据来应用Sobel边缘检测过滤器。
edges = filters.sobel(gray_image)
```

OpenCV库

支持的图像类型和格式:

- OpenCV支持多种图像格式，如Windows位图（*.bmp, *.dib*）、*JPEG文件*（.jpeg, *.jpg, *.jpe*）、*JPEG 2000文件*（.jp2）、可移植网络图形（*.png*）*和WebP*（.webp）等。

读取，显示和写入图像:

```
import cv2

# 读取图像
# 使用`cv2.imread()`函数，通过传递图像文件的路径来读取图像。
image = cv2.imread('example.jpg')

# 显示图像
# 使用`cv2.imshow()`函数来显示图像，在图像窗口上等待用户按下任何键，然后使用
`cv2.destroyAllWindows()`关闭图像窗口。
cv2.imshow('Image', image)
cv2.waitKey(0)
cv2.destroyAllWindows()

# 保存图像
# 使用`cv2.imwrite()`函数，通过传递新的图像文件名和路径以及图像数据来保存图像。
cv2.imwrite('new_example.png', image)
```

图像处理功能:

- OpenCV提供了广泛的图像处理功能，包括图像处理、图像文件读写、视频I/O、高级GUI、视频分析、相机校准和3D重建、2D特征框架、对象检测、深度神经网络模块、机器学习等。

```
# 转为灰度图像
# 使用`cv2.cvtColor()`函数，通过传递图像数据和颜色转换代码来将图像转换为灰度图像。
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# 应用Canny边缘检测
# 使用`cv2.Canny()`函数，通过传递灰度图像数据以及边缘检测的两个阈值来应用Canny边缘检测。
edges = cv2.Canny(gray_image, threshold1=100, threshold2=200)
```

问题 2 黑白图像灰度扫描 (20 分)

运行方式：将其使用安装有Jupyter notebook的VS code 打开，选择版本恰当（>3.8 为佳）的python kernel，依次序运行ipynb文件中问题2对应的代码块。

实验及结果

问题3 彩色图像转换为黑白图像

依次序运行ipynb文件中对应的问题3的三个代码块。

实验结果

从实验结果图中可以看出，NTSC的方法考虑到了人眼的彩色感知体验，加权计算RGB三通道的计算方式更好地保留了细节，对比度更好，相较于平均计算效果更优。

问题 4 图像二维卷积函数 (20 分)

运行ipynb中问题对应代码块即可。

问题 5 归一化二维高斯滤波核函数 (20 分)

运行ipynb中问题对应的代码即可。

问题 6 灰度图像的高斯滤波 (20 分)

实验结果：

- **与直接调用opencv中的函数的效果对比：**

对于σ=1 下的结果，自己实现的高斯滤波和opencv中实现的高斯滤波对各个图像的结果有一定的噪声。

- **选择两幅图像的填充方式对比**

在sigma为1的情况下，选定cameraman, einstein两幅图像，其replicate和zero padding的区别如图所示：

使用了replicate复制策略的图像相较于使用了补零的图像在边界更加平滑，补零的图像有明显的黑边，而replicate是没有显示出来的。