问题 1 黑白图像灰度扫描

实现思路：

采用 cv2.imread 函数以灰度图像方式读取图像路径。当图像的行数或列数为奇数时，直接取中心行和中心列；当行数或列数为偶数时，取中间 2 行或 2 列的灰度平均值作为中心灰度值。

extract(path)函数读取图像路径 path，并通过调用 scanLine4e 函数提取灰度图像中心行和中心列的灰度值，最后返回灰度值矢量并可视化输出。

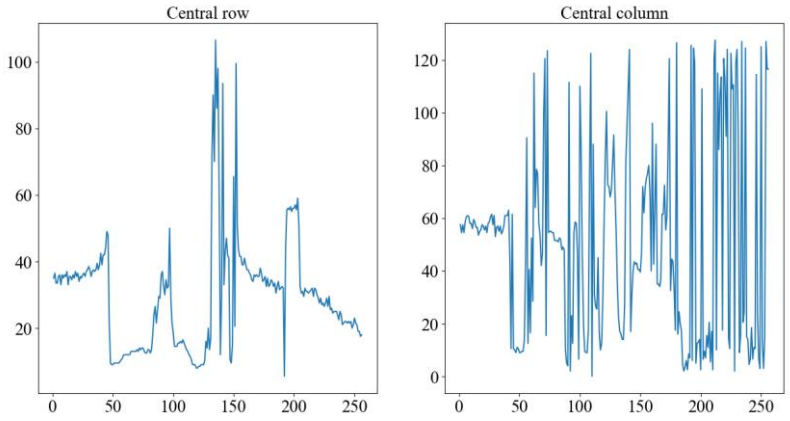


图 1：图像 cameraman.tif 中心行和中心列灰度值

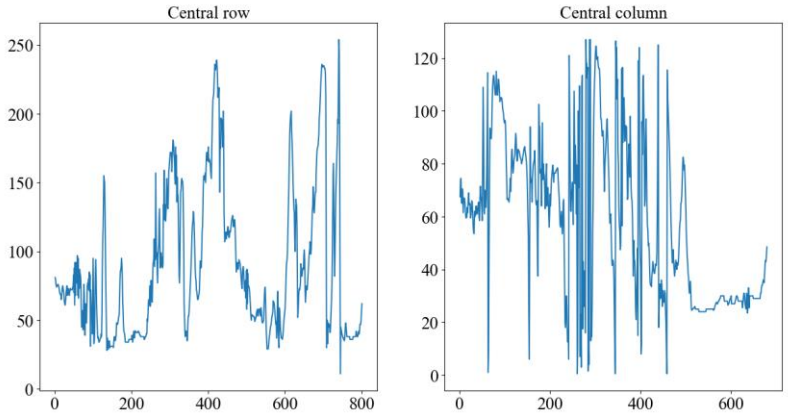


图 2：图像 einstein.tif 中心行和中心列灰度值

问题 2 彩色图像转换为黑白图像

图 3：average、NTSC 与库函数转换结果对比

由图 3 可知，对于两种将彩色 RGB 图像转换成单色灰度图像的方法，average 方法对 R、G、B 三个通道取均值，转换后的灰度图像并不符合人类视觉习惯，与 opencv 库函数转换结果相差较大；NTSC 方法对 RGB 三通道进行加权，转换后的灰度图像和库函数结果相近，但并不完全相同，说明 opencv 自身采用的并不完全是 NTSC 方法。

问题 3 图像二维卷积函数

首先注意到，卷积操作需要对卷积核 w 进行上下左右翻转。

根据卷积后图像尺寸的计算公式

$$L' = \frac{L + 2 * padding - kernel\_size}{stride} + 1$$

在 $kernel\_size$ 为奇数且 $stride = 1$ 的前提下，若卷积后尺寸不变，则填充大小满足

$$padding = \frac{kernel\_size - 1}{2}$$

在填充时，将待填充区域分为 8 个部分，即上、下、左、右、左上、左下、右上和右下，分别取和最近的像素值相等。

在计算卷积时，应先设定数据格式为 float，全部计算完成后，再转为 np.uint8 数据格式输出结果。

问题 4 归一化二维高斯滤波核函数

首先需要确定高斯滤波器的大小 M

$$M = 1 + 2 \times round(3\sigma)$$

对以下几种情况展开讨论：

①m 没有给定：取高斯滤波核大小为 M ；

②m 给定但 $m < M$ ：程序给出 Warning，提示高斯滤波核太小；

③m 给定且 $m \geq M$ ：最终高斯滤波核大小为 m。

高斯核函数的表达形式为

$$G(s,t) = Ke^{-\frac{s^2+t^2}{2\sigma^2}}$$

其中 K 为归一化因子。

问题 5 灰度图像的高斯滤波

调用 gaussKernel 和 twodConv 函数，采用 0 填充方式对图像进行高斯滤波，并和 opencv 官方高斯滤波函数 cv2.GaussianBlur 进行对比，结果如图 4 所示。

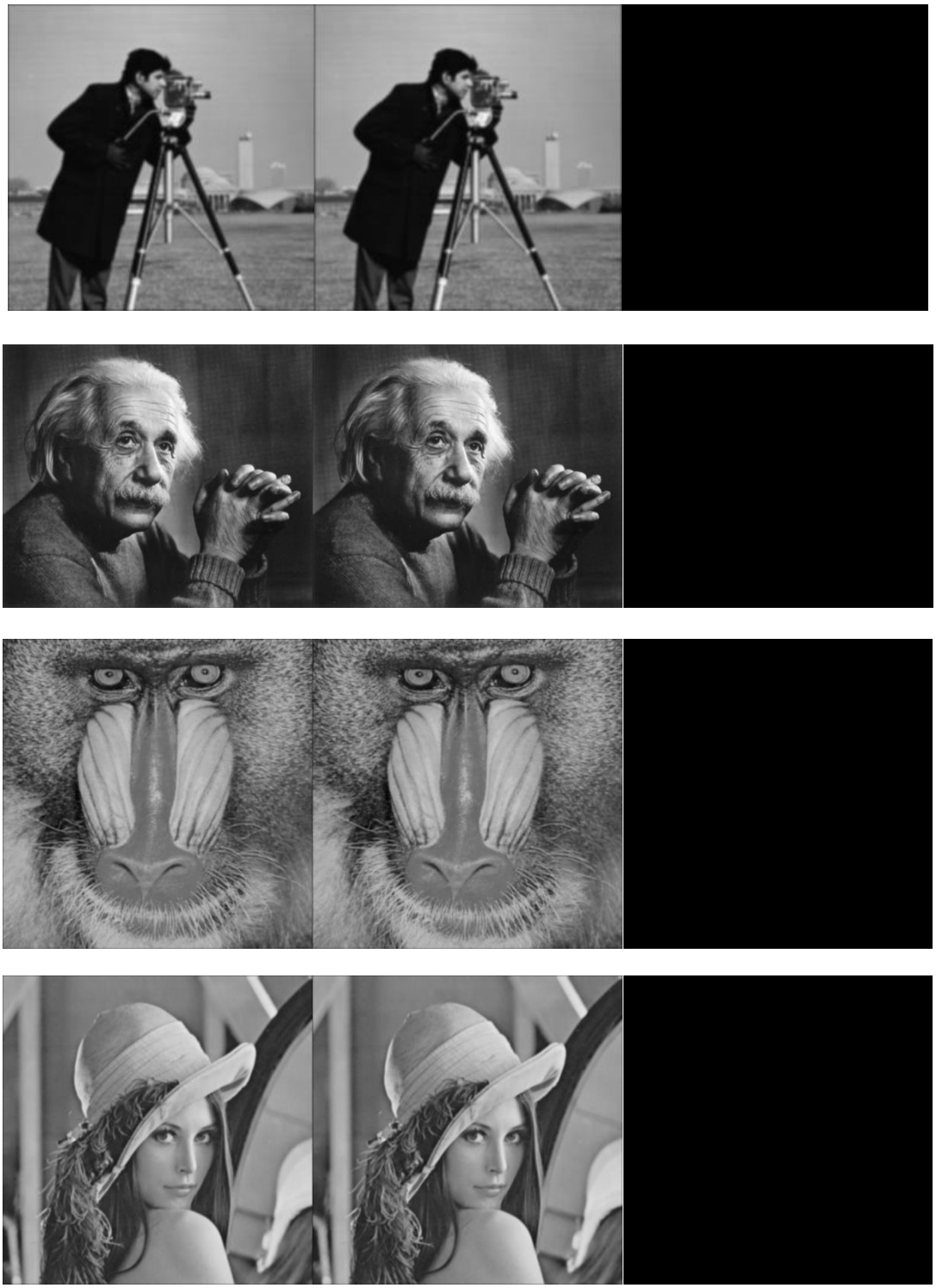
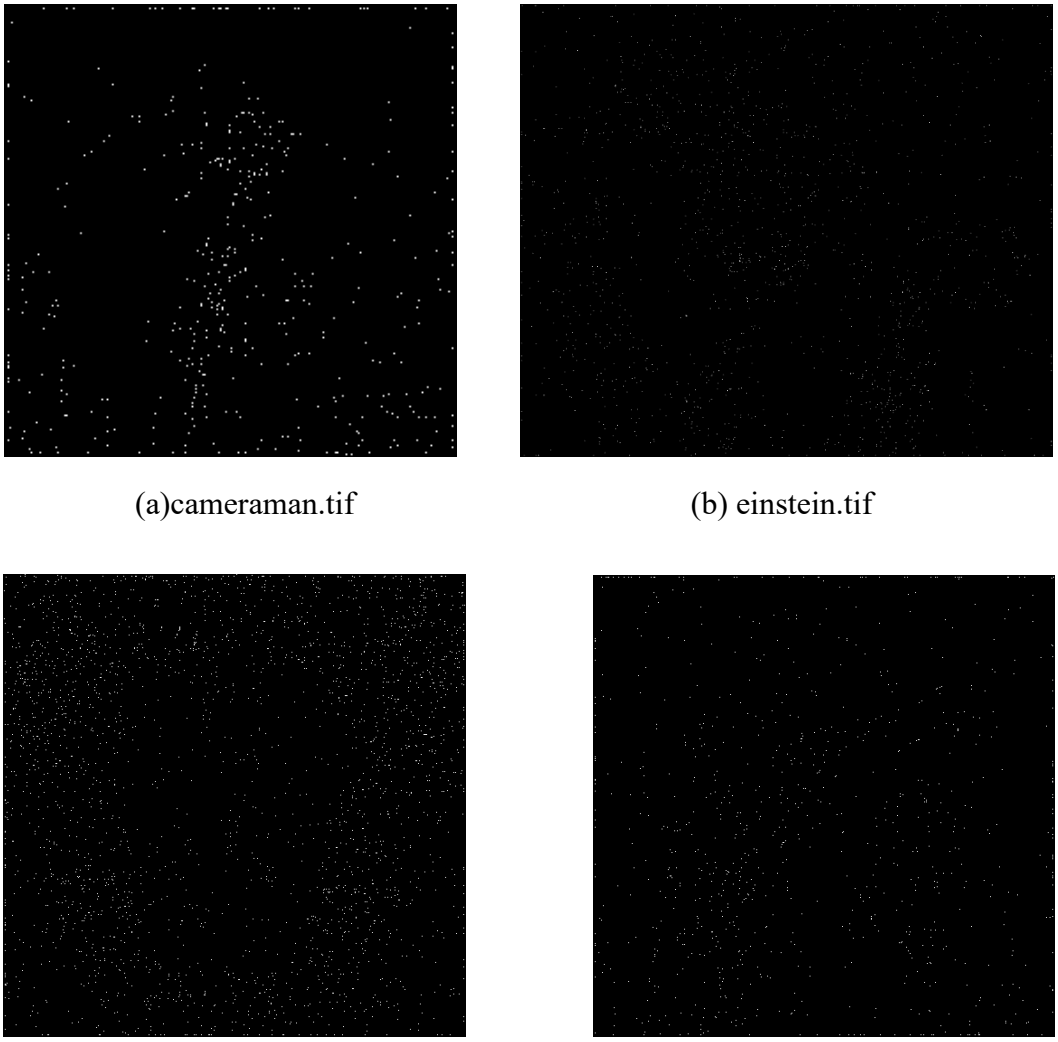


图 4：gaussKernel+twodConv 滤波、cv2.GaussianBlur 滤波和差值图像

通过计算可知在自定义 gaussKernel+twodConv 滤波和 cv2.GaussianBlur 滤波结果中，灰度值相差最大为 1，因此可以认为两者实现了相同的功能。为可视化差异点，将相差为 1 的灰度映射为 255 的灰度值，差异可视化结果如下所示。



(a)cameraman.tif　　(b) einstein.tif

(c)mandril_color.tif　　(d) lena512color.tiff

图 5：可视化差量图像

下面比较边界填充方式为 0 填充和 replicate 填充的区别，选择 cameraman.tif 作为比较对象。

(a) 0 填充的滤波图像　　(b)replicate 填充的滤波图像

图 6：0 填充和 replicate 填充滤波图像

对比图 6(a)和图 6(b) 可知，0 填充的图像边缘有 1 圈黑色边框，而 replicate 填充不存在黑色边框。这是由于边缘填充的灰度值计算更大，0 填充在计算得出的边缘灰度值较小，由此导致了边框的区别。