

**视听觉信号处理**

**实验报告**

题 目 Experiment1

学 院 计算学部

专 业 视听觉信息处理

学 号

学 生

任 课 教 师 姚鸿勋

哈尔滨工业大学计算机科学与技术学院

2021秋季

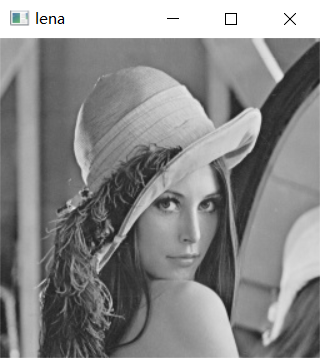
1. **实验目标**

**1.掌握图像处理中读取、显示、保存。**

**2.掌握图像处理中空域的增强算子。**

**3.掌握图像直方图概念，实现图像的直方图均衡化。**

1. **实验内容**
2. 实现图像的读取、显示、保存操作。
3. 实现图像的空域增强算子，包括中值滤波和均值滤波算法，显示并保存结果图像。
4. 实现图像的直方图均衡化，显示并保存结果图像。
5. **实验结果**
6. 图像的读取、显示、保存



1. 图像的空域增强算子

中值滤波：

原图

滤波后

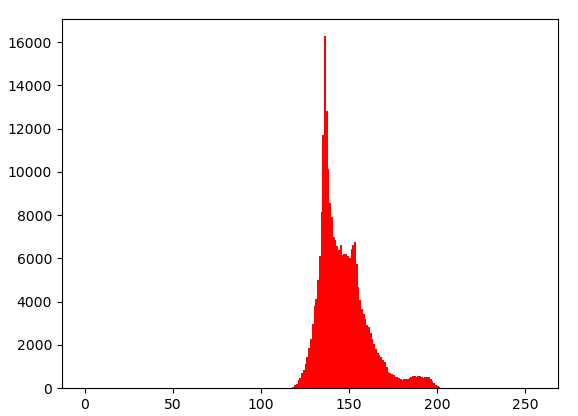
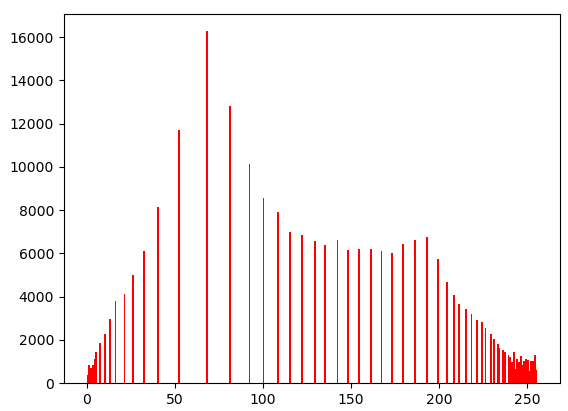
均值滤波：

原图

滤波后

1. 直方图均衡化

原图的直方图 直方图均衡化之后的直方图



原图



直方图均衡化后

1. **实验分析**
2. 图像的读取、显示、保存

图像读取：

img = cv.imread('image/lena256color.jpg', 0)

以灰度图形式读取对应路径的图像。

图像显示：

cv.imshow('lena', img)

将图像对象显示在相应的窗口中。

图像保存：

cv.imwrite('image/new\_save.jpg', img)

将图像对象保存在指定路径。

1. 图像的空域增强算子

中值滤波和均值滤波都是以一个小矩阵遍历整个图像，将某一像素附近的几个像素值经过某种运算，赋值到新的图像中。

以3\*3滤波器为例，在图像的(i,j)处，包括自身在内，周围共有9个像素值，将这9个像素值进行某种运算得到一个新的值，赋值到新的图像的对应位置。

中值滤波：取上述9个像素值的中位数。其中np.median输入一个数组或矩阵，能够返回它们的中位数。

|  |
| --- |
| for i in range(r // 2, R - r // 2):  for j in range(c // 2, C - c // 2):  k = img[i - r // 2:i + r // 2 + 1, j - c // 2:j + c // 2 + 1]  ret\_img[i, j] = np.median(k) |

均值滤波：取上述9个像素值的平均数。其中np.mean输入一个数组或矩阵，能够返回它们的均值。

|  |
| --- |
| for i in range(r // 2, R - r // 2):  for j in range(c // 2, C - c // 2):  k = img[i - r // 2:i + r // 2 + 1, j - c // 2:j + c // 2 + 1]  ret\_img[i, j] = np.mean(k) |

一般来说，对于椒盐噪声的图像，使用中值滤波效果更佳。因为对于一个滤波器，某一个像素值变成0或255对这一列数字的中值几乎没有影响，所以有中值滤波处理椒盐噪声图像基本可以将图像复原。对于高斯噪声图像，使用均值滤波效果更佳。因为高斯噪声图像是对每一个像素值加上一个高斯噪声均值为0，使用均值滤波器处理之后，由概率论与数理统计相关知识，这一列数的噪声均值仍然为0，所以使用均值滤波处理高斯噪声图像较为合适，更加贴近使用均值滤波器降噪的效果。

1. 直方图均衡化

直方图均衡化通过为每个像素值计算出一个新的像素值映射，然后将原图映射到有新像素值组成的图像上实现直方图均衡化。

步骤：

1. 统计原始图像各灰度级的像素数目nj。np.zeros是将数组初始化为全0。

|  |
| --- |
| R, C = img.shape  nj = np.zeros(shape=256)  for i in range(R):  for j in range(C):  nj[img[i, j]] += 1 |

1. 作灰度分布直方图pf。

|  |
| --- |
| pf = nj / (R \* C) |

1. 计算累积分布函数。

|  |
| --- |
| for i in range(1, 256):  cf[i] = cf[i - 1] + pf[i] |

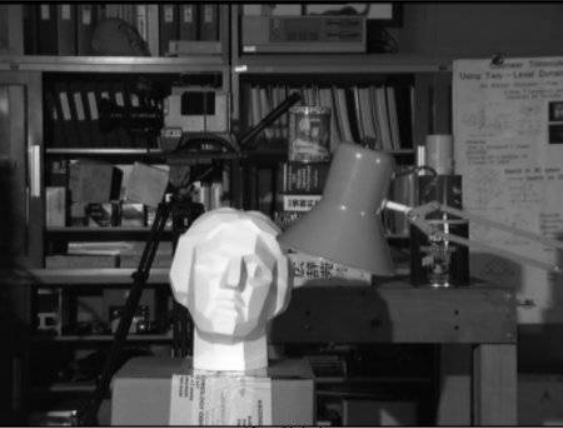
1. 计算映射后的。

|  |
| --- |
| gj = np.zeros(shape=256)  for i in range(256):  gj[i] = int(np.floor(255 \* cf[i] + 0.5)) |

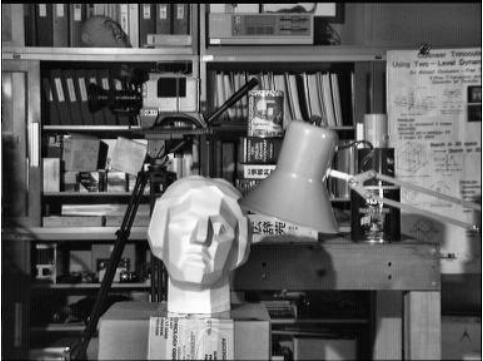
1. 计算原图像每个像素点的映射并输出图像。

|  |
| --- |
| for i in range(R):  for j in range(C):  ret\_img[i, j] = gj[img[i, j]] |

有时普通的直方图均衡化可能效果不佳，比如下图：



左图是原图，右图是通过直方图均衡化处理之后的图。结果发现，图中雕塑的效果较差，其余地方效果都较好。对于这种情况，可以使用自适应直方图均衡化的方法。首先将图像分为若干个小子图，如8\*8，然后对于每一个子图进行普通的直方图均衡化，然后拼接起来即可。结果如下图所示：



1. **实验总结**

通过本次实验，我掌握了图像的一些简单操作，对图像处理的一些基本方法，包括均值滤波、中值滤波、直方图均衡化，以及这些方法的一般适用情况。通过这次实验，我对数字图像的一些基本处理操作有了一个了解。