哈爾濱工業大學

视听觉与信号处理实验报告

实验 2

学　　院　计算机科学与技术学院

专　　业　视听觉信息处理

学　　号　1170300511

学　　生　易 亚 玲

任 课 教 师　姚 鸿 勋

哈尔滨工业大学计算机科学与技术学院

2019 年秋季

一、 实验内容（contents）

1. 实现图像直方图均衡化，要求显示均衡化前、后直方图以及均衡化后图像。
2. 对单通道图像进行 DFT 变换，要求显示幅度图和相位图，并设计理想高通滤波器和高斯低通滤波器对图像进行频域滤波，并显示滤波之后的图像。

二、 实验目的（purposes）

1. 掌握图像直方图的概念以及直方图均衡化的方法。
2. 理解图像变换和重建的思想和方法。
3. 掌握图像处理中频域滤波的概念、方法以及理想、高斯等频域滤波器。

三、 实验设计、算法和流程(Design, algorithm and procedure)

3.1 图像的直方图

3.1.1 概念

图片的直方图可以反映图片各个像素点的像素值的分布，如果图片整体较暗，则在直方图的表现为像素值小的地方分布较为集中，像素值大的地方分布较稀疏；如果图片整体较亮，则图片的像素点主要集中分布在像素值较大的区域；如果图片的亮度相对平均，则直方图分布相对均衡。

3.1.2 直方图均衡化

对于一些直方图分布不均衡的图像，往往表现出细节地方较暗，无法辨别；或者过度曝光、图片太亮，无法识别图片的细节。于是，我们希望改善直方图分布不均匀的情况，从而使图片出现更多的细节。由直方图分布不均匀的图片变成一个直方图分布相对均衡的图片，我们需要寻找一个映射，将原本小范围像素值的直方图，映射到分布更为广泛的直方图。要寻找这个映射，我们必须先了解原本的图像参数，即了解图片中哪些像素值分布密集，哪些像素值分布较少甚至没有分布，所以我们首先要统计图像中各个像素值出现的次数，然后根据总像素点的个数，求出每个像素值（0-255）出现的频率。接下来的思路就比较巧妙：由于图片的像素由 0 到 255 是均匀变化的，假如图片的直方图分布是完全均匀的，那么概率分布函数（以像素值为自变量）和图片像素点的像素值是线性相关的。因此，我们可以用像素值的概率分布函数的变换来完成图片的直方图的变换。在这种情况下，均衡化后的图片的像素值可以表示为原概率分布函数的值乘以新的像素值范围，这样就完成了将图片的像素值范围扩大，从而达到均衡的目的。由上面的分析，图片的直方图均衡化主要可以分为一下几个步骤：

1. 统计图片中各个像素值的个数（像素值范围是 0-255）
2. 根据上一步统计的个数，进行归一化得到频率
3. 根据统计得到的频率，按像素值的从小到大求得概率分布函数
4. 根据概率分布函数，求出新的图片的各点的像素值，新的像素值的计算公式为：新的范围大小乘以旧的像素值的累计概率值
5. 至此图片均衡化已经结束，可以对得到的均衡化的图片进行一些统计分析，比较新旧直方图的区别。

根据以上步骤，代码的实现如下：

```
# 1. 统计各灰度值的像素数目
p = [0 for i in range(MAX_NUM)]  # 初始化各个灰度值的个数为0
len_x = len(img_array[:, 0])
len_y = len(img_array[0, :])
for i in range(len_x):
    for j in range(len_y):
        p[img_array[i][j]] += 1  # 计数
# 2.计算各灰度值出现的概率
p = [x / (len_x * len_y) for x in p]
# 3.计算累计分布函数
c = [0 for i in range(MAX_NUM)]
c[0] = p[0]
for i in range(1, MAX_NUM):
    c[i] = c[i - 1] + p[i]
# 4.计算映射后的灰度图
after_hist = np.zeros((len_x, len_y), dtype='uint8')
for i in range(len_x):
    for j in range(len_y):
        after_hist[i][j] = (MAX_NUM - 1) * c[img_array[i][j]] + 0.5
# 5.统计变换后各灰度值出现的次数
p_after = [0 for i in range(MAX_NUM)]
for i in range(len_x):
    for j in range(len_y):
        p_after[after_hist[i][j]] += 1
# 6.统计变换后各点出现的频率
p_after = [x / (len_x * len_y) for x in p_after]
x = [i for i in range(MAX_NUM)]
```

3.2 DFT 变换

3.2.1 DFT 的概述

DFT 是将空域的信号变换到频域的一种方法，有些噪声在空域直接去除效果并不好，但是转化到频域就能很好的去除，特别是对于乘性噪声。我们通过 log 和频域变换后可以直接将乘性噪声转换为加性噪声，这样更加方便去除，DFT 为图像处理提供了一种新的思路。但是由于 DFT 转化过程中存在舍入，可能会导致微小偏差。

3.2.2 DFT 的实现

在进行 DFT 变换之前，首先要将图片的像素值转化为 float32 类型的数，然后调用 cv2 的 DFT 函数，得到 DFT 变换后的实部和虚部的信息。如果要显示图片的幅度图，可以使用 np.fft.fftshift 将图片的低频部分变换到中心，高频部分在四周，这样显示更加清楚一些。根据 DFT 得到的每个像素点的实部和虚部，可以求得每个像素点对应的相位角，并进行归一化，然后显示相位角对应的图片，可以得到相位图。代码实现如下：

```
dft_array = cv2.dft(np.float32(img_array), flags=cv2.DFT_COMPLEX_OUTPUT)
# 将图片的低频部分移动到中心位置
dft_shift = np.fft.fftshift(dft_array)
# cv2.magnitude()计算矩阵的加权平方根，将实部和虚部投影到空间域
magnitude_spectrum = np.log(cv2.magnitude(dft_shift[:, :, 0], dft_shift[:, :, 1]))
# 计算相角
len_x = len(img_array[:, 0])
len_y = len(img_array[0, :])
phase = []
for i in range(len_x):
    for j in range(len_y):
        phase.append(np.complex(dft_shift[i, j, 0], dft_shift[i, j, 1]))
phase = (180.0 * np.angle(phase) / np.pi + 2 * np.pi) % 360 / 360  # 相位信息,归一化
phase = np.array(phase).reshape(len_x, len_y)  # 重构矩阵
```

3.3 频域滤波

首先，频域滤波和空域滤波最大的区别是：空域是执行卷积运算，而频率是执行乘积运算。所以在频域操作能减少计算乘积的次数（不计空域频域的转换操作）。

3.3.1 理想高通滤波

首先进行 DFT 变换并将低频信号移动到中心位置(shift 操作)，然后构造一个掩膜，这个掩膜的尺寸和图片尺寸相同，但中心低频部分的取值为 0，其余部分取值为 1，让这个掩膜与 shift 操作后的图像做乘积运算完成理想高通变换。将掩膜作用后的图像进行 ifftshift 操作变换回正常的高低频关系，再执行反傅里叶变换并求幅值映射到空间域，最后得到的这张图片就是理想高通滤波作用后的图片。代码实现如下：

```
dft_array = cv2.dft(np.float32(img_array), flags=cv2.DFT_COMPLEX_OUTPUT)
# 将图片的低频部分移动到中心位置
dft_shift = np.fft.fftshift(dft_array)
# 构造掩膜（中心为0，其余为1）
len_x = len(img_array[:, 0])
len_y = len(img_array[0, :])
mask = np.ones((len_x, len_y, 2))
for i in range(int(len_x / 2 - 10), int(len_x / 2 + 10)):
    for j in range(int(len_y / 2 - 10), int(len_y / 2 + 10)):
        mask[i][j][0] = 0  # 中心置0
        mask[i][j][1] = 0
# 添加掩膜
mask_img = np.multiply(mask, dft_shift)
# 高低频恢复正常排布
de_shift = np.fft.ifftshift(mask_img)
# 傅里叶反变换
de_dft = cv2.idft(de_shift)
# 将图像转为空间域
dft_img = cv2.magnitude(de_dft[:, :, 0], de_dft[:, :, 1])
```

3.3.2 高斯低通滤波

高斯低通滤波的函数表达式为

$$H(u,v) = e^{-[(u-\frac{M}{2})^2+(v-\frac{N}{2})^2]/2\sigma^2}$$

当 sigma 确定的时候，离中心点较近的点高斯滤波的值较大，而离中心点较远的点高斯滤波的值较小。而经过 shift 操作后的图像中心为低频信息，边缘为高频信息，所以高斯滤波是低通滤波。高斯低通滤波相当于对图像进行平滑操作，其中 sigma 越小，图像的平滑效果越明显，当 sigma 太小的时候，图片会完全丢失以前的特征信息。

代码实现如下：

```
dft_array = cv2.dft(np.float32(img_array), flags=cv2.DFT_COMPLEX_OUTPUT)
# 将图片的低频部分移动到中心位置
dft_shift = np.fft.fftshift(dft_array)
# 构造掩膜（中心为0，其余为1）
len_x = len(img_array[:, 0])
len_y = len(img_array[0, :])
# 添加高斯低通滤波
for i in range(len_x):
    for j in range(len_y):
        d = np.exp(-((i - len_x / 2) ** 2 + (j - len_y / 2) ** 2) / (2 * sigma ** 2))
        dft_shift[i][j][0] *= d
        dft_shift[i][j][1] *= d
# 高低频归位
de_shift = np.fft.ifftshift(dft_shift)
# 反傅里叶变换
de_dft = cv2.idft(de_shift)
# 将图像映射至空间域
dft_img = cv2.magnitude(de_dft[:, :, 0], de_dft[:, :, 1])
```

四、 实验结果(results)

4.1 直方图均衡化

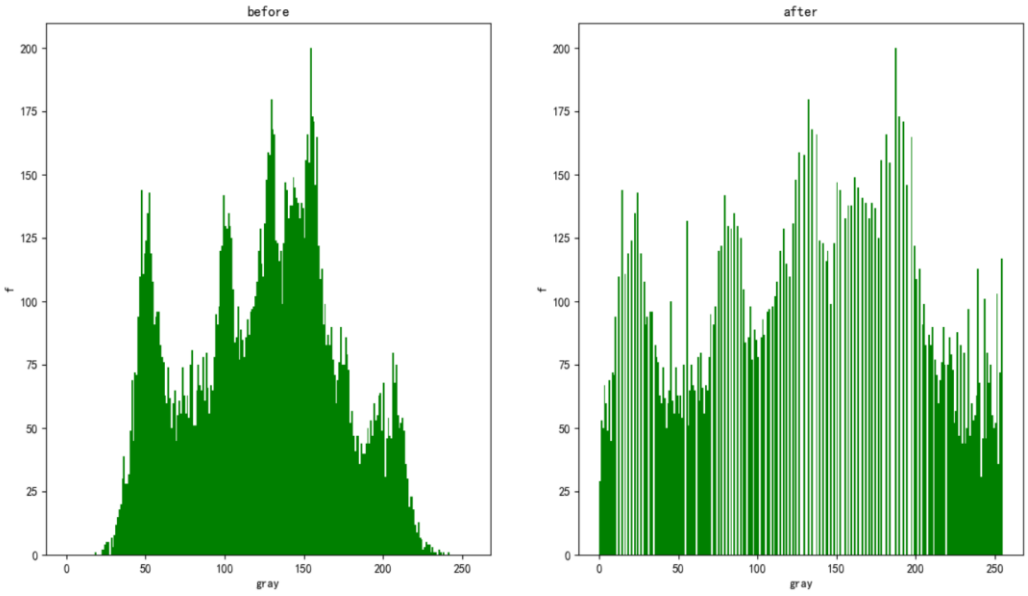

图 1 直方图均衡化前后的直方图对比。由上图可知，原图像的灰度值主要集中分布在中间位置，而较小和较大的像素值分布较少；均衡化的图片整体分布比较均匀，虽然没有达到绝对均匀，但是分布的范围明显比均衡化以前要广。

图 2 直方图均衡化前后的图像对比。根据对比可以发现，原图整体较暗，暗区域和亮区域的界限不那么明显，对比比较柔和，图片整体呈现灰色；而直方图均衡化的图片，对比度明显增强了，原本比较暗的区域更暗了，亮的区域更白了，图片不同灰度值的边界更加清晰，图片也显得更有层次感。

4.2 DFT 变换的幅度图、相位图

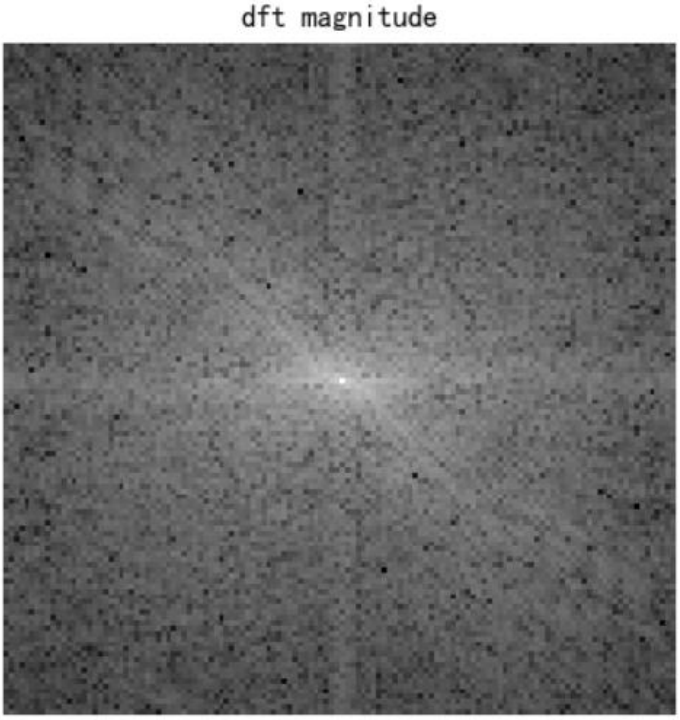

图 3 DFT 变换后图片的幅度图（由于原幅度值太大，在原幅度值的基础上增加了进了 log 运算，否则图片只能观察到一个白点）。从图片的幅度图来看，图片具有明显的堆成性，因此图片存在正弦分量。

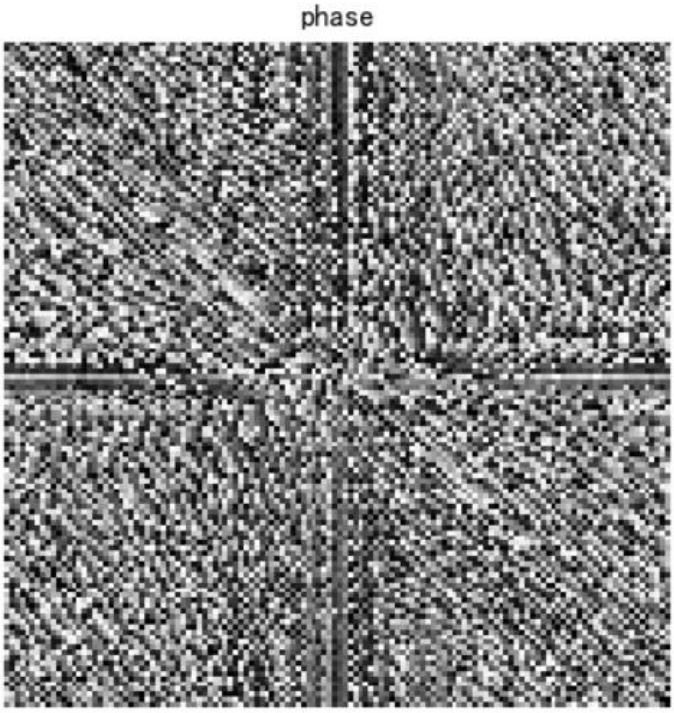

图 4 DFT 变换后的相位图。横竖有两条分界线，不知道是不是和 shift 变换有关系，或者说幅度图和相位图本身存在某种联系。

4.3 理想高通滤波

图 5 理想高通滤波前后图像对比。上图是滤掉了 100 个低频的效果，整体图像相比之前变暗了很多，边界部分保留较好。由此可以初步判断，图片的边界信息为高频信号，其余的信号相对为低频信号。过滤的低频信号越多，图片保留的低频信号越少，少到一定程度时，图片几乎只剩边界信息。

4.4 高斯低通滤波

高斯滤波相当于平滑，受 sigma 的影响，sigma 取值的不同，平滑效果差别很大。观察下面一组图：

图 6 sigma = 1。图片完全失去原本的特征。

图 7 sigma=5。图片很模糊，但是能勉强分辨出图片的轮廓。

图 8 sigma=10。图片较模糊，图中人物的眼部轮廓和头发轮廓都可以分辨。

图 9 sigma = 20。图片有点模糊，眼睛的细节部分不太清晰，头发上糊成一团，但是大多数轮廓都很清楚。

图 10 sigma=30。图片有点模糊，但是头发和帽子的细节有保留，整体模糊感可以接受。

图 11 sigma=40。图片已经没有明显的模糊感了，但是面部仍然有轻微涂抹的痕迹。

图 12 sigma=50。整张图片效果很好，由于平滑的原因，图片噪声变少了，脸部看起来

比原图更舒服一些。

图 13 sigma=70。已经完全看不出原图和高斯滤波的区别了，当 sigma 大于 70 时，肉眼几乎看不出区别，也就是高斯滤波失去效果

。

根据这组图可以看出，sigma 的值越小，图片越模糊，晕影更重。当 sigma < 5 时，图片的轮廓完全不可见；当 sigma 为 5-20 时，图片只能勉强看到轮廓，几乎没有什么细节；当 sigma 为 20-50 时，图片有涂抹痕迹，但是形状是完全清晰的，且随着 sigma 的增大，保留的细节越多；当 sigma>70 时，肉眼几乎无法分辨两张图片的区别。

五、 结论(conclusion)

1. 图片的直方图反映图片的像素值分布规律，当图片的直方图分布均衡时，图片的熵更大，往往意味着图片有更多的细节。我们可以通过直方图均衡化对一些直方图分布不那么均衡的图片做处理，往往可以拥有更多细节或者有利于进一步对图像做去噪、平滑等处理。
2. 图片空域的卷积操作空域变换到频域做乘积操作，能够减少运算次数，而且不改变图片的尺寸。有一些空域不易去除的噪声（往往是高频的），变换到频域进行低通滤波往往能很好地去除。
3. 理想高通滤波的中心块设置较大的时候能够粗略地求取图片的边缘，也许能用于简单的边缘检测，比空域算子进行边缘检测计算的乘积次数更少。
4. 高斯低通滤波的效果受 sigma 的影响，当 sigma 取值合适时，能够去除一些顽固的噪声，达到很好的去噪效果。高斯低通滤波还能通过改变中心点，去除一些中等频率的滤波，应用很广泛。