同态滤波——一种基于傅里叶变换的图片处理操作

## 同态滤波简介：

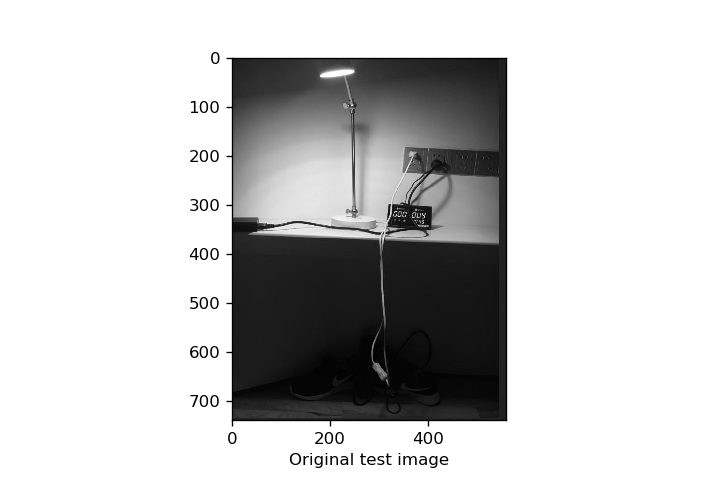


Figure 1 一个测试图片

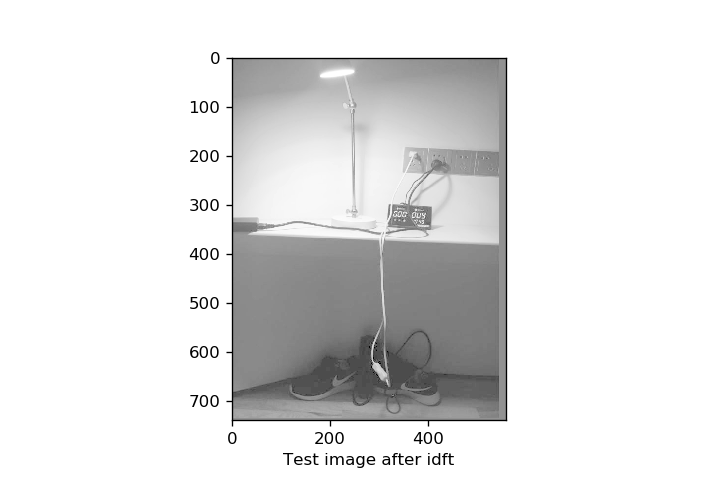


Figure 2 同态滤波处理后的测试图片（注意下方的鞋和黑色的吹风机）

同态滤波是一种在频域中同时将图像亮度范围进行压缩和将图像对比度进行增强的方法，是基于图像成像模型进行的。

为了分离加性组合的信号，常采用线性滤波的方法，而非加性信号组合常用同态滤波的技术将非线性问题转化成线性问题处理，即先对非线性（乘性或者卷积性）混杂信号作某种数学运算，变换成加性的。然后用线性滤波方法处理，最后作反变换运算,恢复处理后图像。

同态滤波中会涉及到傅里叶变换。傅里叶变换把图像变换到了频率域：

有关傅里叶变换的推导详见教材128页。

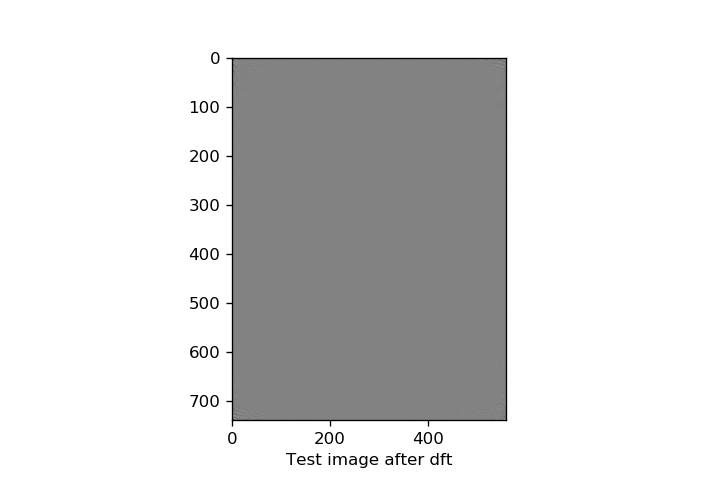


Figure 3 傅里叶变换后测试图像的频率图

**同态滤波器的公式推导**

公式推导详见教材152页。

公式的核心就是推导出同态滤波器，而同态滤波器在实际图像处理程序中就是一个矩阵。矩阵的值由公式给出。对于图像进行处理就是用这个矩阵来乘以图像傅里叶变换后的频率图。

同态滤波器既然是个矩阵，那么它也可以画出来：

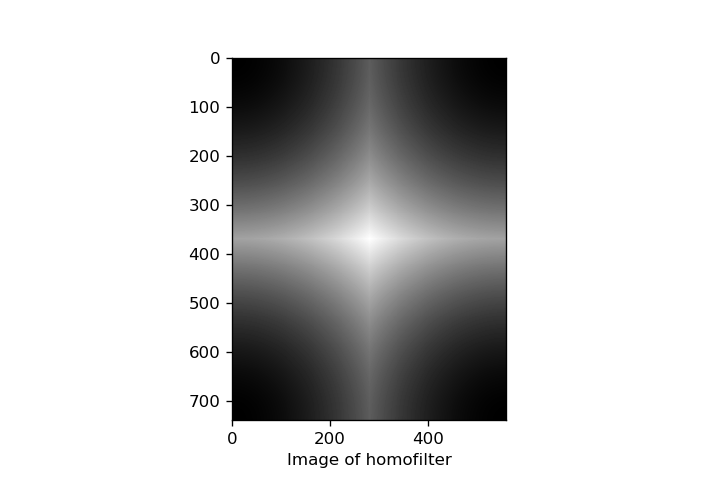


Figure 4 上面那个测试图中用到的同态滤波器

**同态滤波的步骤：**

-> -> DFT -> H(u,v) -> (DFT)` -> ->

简单来说分以下5步：

1. 取对数
2. 傅里叶变换（DFT）
3. 同态滤波器处理
4. 逆傅里叶变换
5. 取指数

**同态滤波的python实现：**

1. python准备

下载python

找到python目录下的pip.exe文件（可以ctrl+F搜索）

在pip.exe的目录下使用命令框，输入以下命令安装包裹：

pip.exe install opencv

pip.exe install matplotlib

pip.exe install numpy

以上三个包分别用来读取图片、绘制图片、dft的可选替代

1. 读取图片



这个是读取图片，读下来的图片会灰度化（变成单通道），最后变成double类型（方便取对数）。

最后的image是一个 width\*height的double类型的二维数组。

1. 傅里叶变换

接口函数：

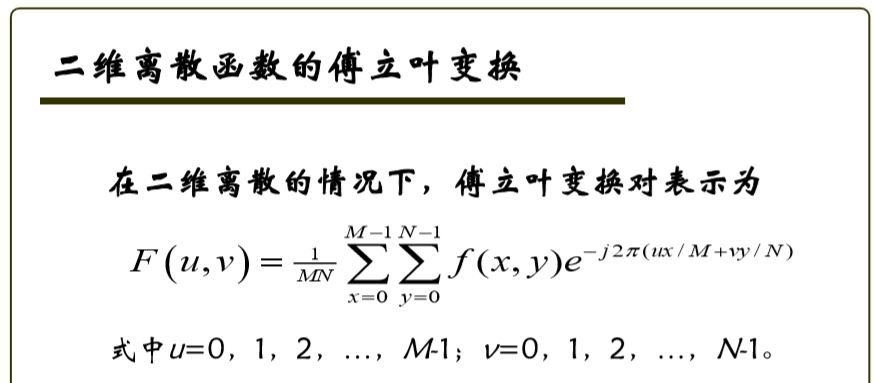
dft\_image = mydft.dft(image)

或dft\_image = npfft.fft(image)

这里有两种傅里叶变换，一种是书上的普通傅里叶变换，一种是numpy实现的快速傅里叶变换。

傅里叶变换实现（DFT）：

先看ppt , dip6 P47：

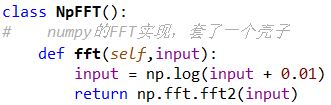


这是对应的实现：



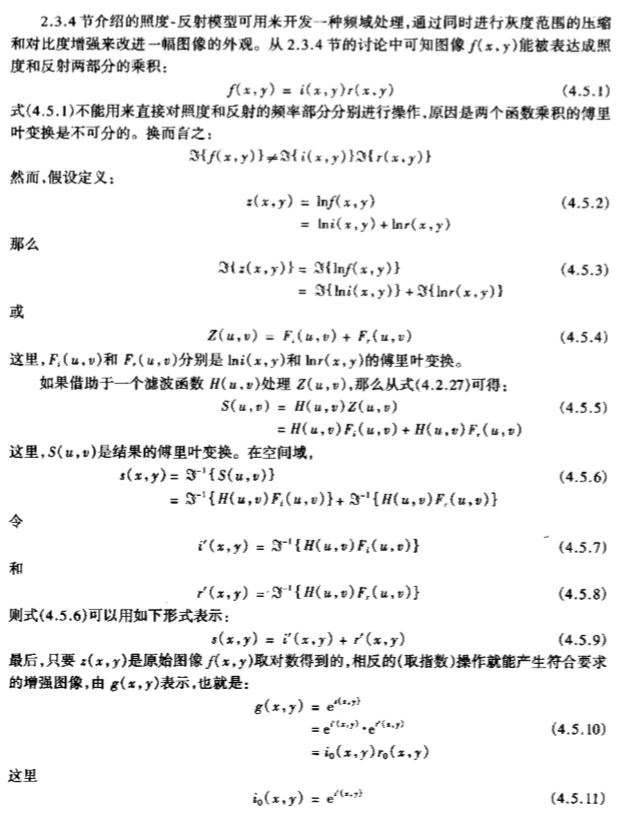
基本上就是按照ppt的来做，参数命名也按ppt的来做。

numpy的快速傅里叶变换：



1. 得到同态滤波器

参考教材：



实现：



注意同态滤波器的四个参数：C、D0、rh、rl

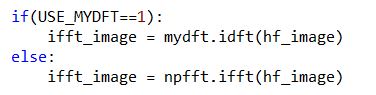
其中调整参数的经验是：D0越大越好（但是效果有极限，就是D0大过一定范围后，就差别不打了），C在rl~rh之间（锐利值，越大越锐利，不要太小即可），rh和rl尽量框住想要看清的范围（框住就行，rh不能太大，容易过亮，rl不能太小会过暗）。

1. 用同态滤波器处理图像：

相乘即可

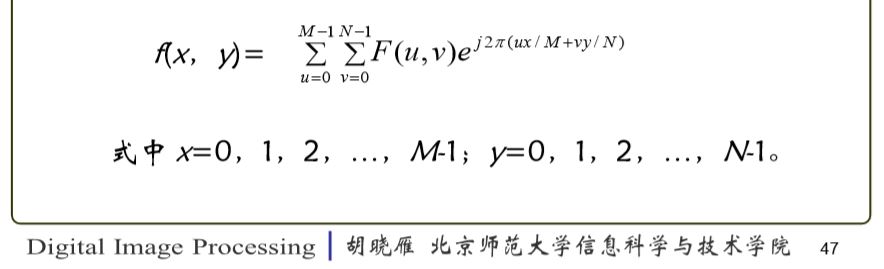


1. 逆傅里叶变换



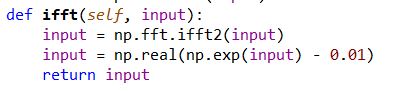
还是两种：一种是书上的普通傅里叶逆变换，一种是numpy实现的快速傅里叶逆变换。

普通傅里叶逆变换ppt:





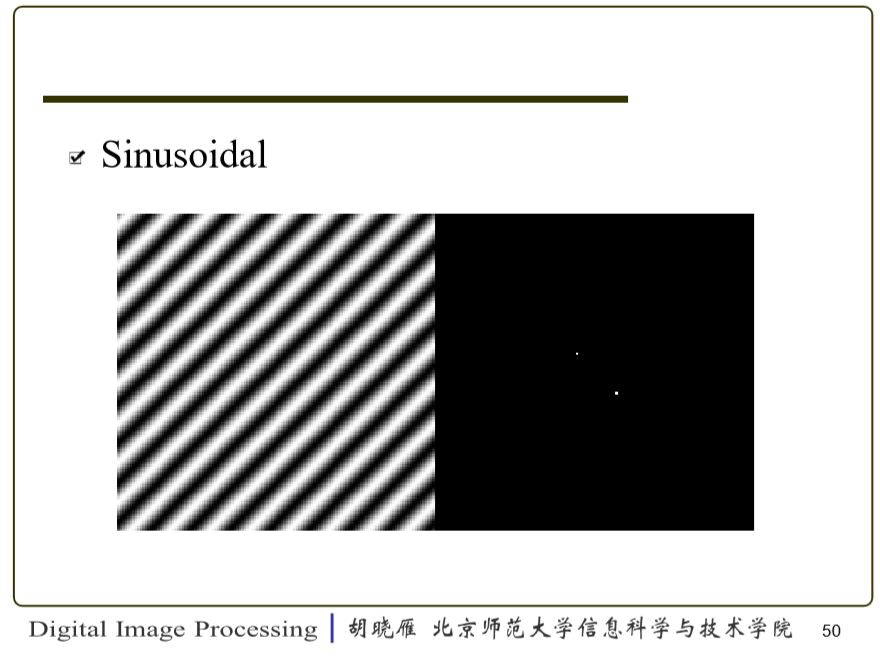
numpy的快速逆变换：



**测试效果**

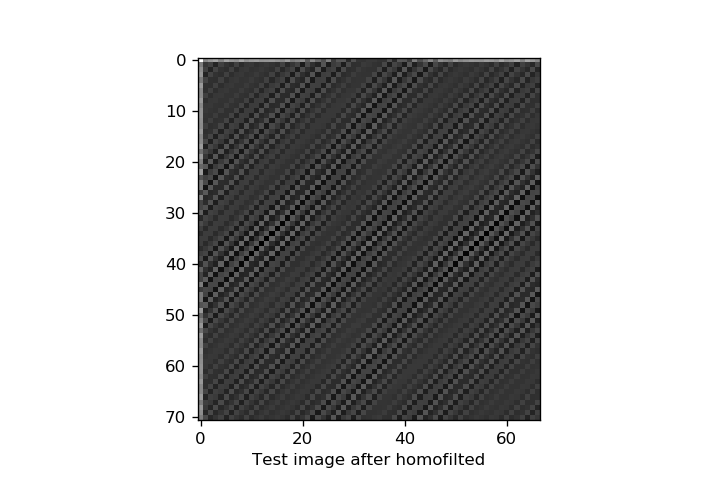
先测试以下DFT对不对：

拿这个ppt上的图来测一下：



左边就是频率域，右边是原图。

测试自己写的dtf，结果如下：



由于像素不是很高（太高运行过于慢），所以只能看个大概的走势。

另外，在只是查看频率图的时候，不要作对数、指数变换，效果会更好。

测试实际图片，由于自己写的dft运行太慢，以下都用numpy的fft：

