同态滤波原理及代码实现

同步滤波的含义

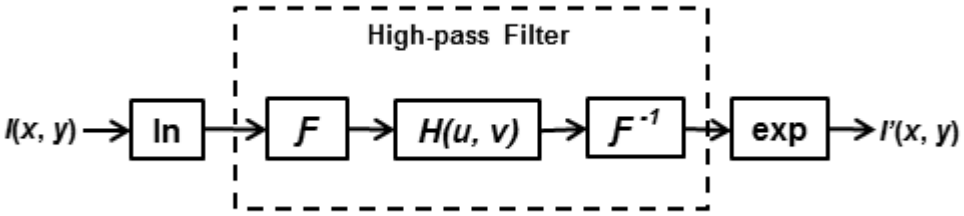
同态滤波是一种数字图像处理技术，它主要用于图像增强，它可以通过去除乘性噪声，同时增加图片对比度以及标准化亮度。

同态滤波原理

一幅图像可以看作照度分量和反射分量的乘积，空间滤波难以将两者分离。而由于照度分量变化缓慢，反射分量相对变化大，两者在频域上可以分离。但两者是乘性叠加，直接进行傅立叶变换仍无法简单分离。

于是，同态滤波采用先取对数，再进行傅立叶变换的方式。由于取对数之后乘性叠加会变为加性叠加，从而使得傅立叶变换之后两者可以分离。在使用高斯滤波器或巴特沃斯滤波器对傅立叶变换后的图片矩阵进行滤波之后，在进行傅立叶反变换和取自然对数指数的操作，便可得到滤除反射分量的图片。

同态滤波的原理如下图：



同态滤波的实现

我们使用OpenCV的Python版本对同态滤波过程进行了实现，代码如下：

```
import cv2
import numpy as np

# 高斯滤波器
def gaussian_filter(I_shape, filter_params):
    P = I_shape[0]/2                                    # 确定图片水平中心
    Q = I_shape[1]/2                                    # 确定图片垂直中心
    H = np.zeros(I_shape)
    U, V = np.meshgrid(range(I_shape[0]), range(        # 生成对应于图片的坐标矩阵
        I_shape[1]), sparse=False, indexing='ij')
    Duv = (((U-P)**2+(V-Q)**2)).astype(float)           # 计算出高斯函数的幂次分子项
    H = np.exp((-Duv/(2*(filter_params)**2)))           # 生成对应于图片的高斯矩阵
    return (1 - H)

# 同态滤波函数
def filter(I, filter_params, H=None, a=1.0, b=1.0):
    H = gaussian_filter(
        I_shape=I.shape, filter_params=filter_params)   # 根据输入参数生成高斯矩阵
    I_log = np.log1p(np.array(I, dtype="float"))        # 图片矩阵求自然对数
```

```
    I_fft = np.fft.fft2(I_log)                              # 傅立叶变换
    I_fft_filt = (a + b * np.fft.fftshift(H)) * I_fft       # 与生成的高斯矩阵乘性叠加
    I_filt = np.fft.ifft2(I_fft_filt)                       # 傅立叶反变换
    I = np.exp(np.real(I_filt))-1                           # 用指数函数复原自然对数
    return np.uint8(I)                                      # 转为图片像素格式

#主函数
if __name__ == "__main__":
    img = cv2.split(cv2.imread('./original.jpg'))           # 读取图片并分离颜色通道
    for i, color in enumerate(img):
        img[i] = filter(I=color, filter_params=50)          # 每种颜色各自进行同态滤波
    cv2.imwrite('./filtered.png', cv2.merge(img))           # 合并颜色通道并输出
```

代码效果对比如下：

- 原图片

- 滤波后