**双边滤波法**

基本原理：

双边滤波(bilateral filter)是一种非线性滤波器，该算法结合空间信息和亮度相似性对图像进行滤波处理，在平滑滤波的同时能大量保留图像的边缘和细节特征。

双边滤波的滤波核由两部分乘积组成：空域核http://img.voidcn.com/vcimg/static/loading.png与值域核http://img.voidcn.com/vcimg/static/loading.png，两个滤波核通常都采用高斯函数形式。

空间域高斯函数其数学形式为：

|  |  |  |
| --- | --- | --- |
|  |  | (1) |

值域高斯函数其数学形式为：

|  |  |  |
| --- | --- | --- |
|  |  | (2) |

相乘得双边滤波函数模板为：

|  |  |  |
| --- | --- | --- |
|  |  | (3) |

以上三式中，为基于空间分布的高斯滤波函数的标准差,为值域高斯函数的标准差。由得到的双边滤波函数模板与噪声图像进行卷积来滤波。

Python代码如下：

**import** cv2  
**import** numpy **as** np  
  
  
**def** bfltGray(img, w, sigma\_d, sigma\_r): # 定义双边滤波函数  
 # 计算关于距离的高斯核矩阵模板  
 D = np.exp(-((x) \*\* 2 + (y) \*\* 2) / (2 \* sigma\_d \*\* 2))  
  
 filted\_img = np.zeros([m, n]) # 定义与原图像大小相同的0矩阵来装滤波后的图像  
 # 计算值域核矩阵R模板，双边滤波器，并滤波  
 **for** i **in** range(m):  
 **for** j **in** range(n):  
 # 抽取与模板大小相同的图像区域，注意边界位置的特殊性  
 iMin = max(i - w, 0)  
 iMax = min(i + w, m - 1)  
 jMin = max(j - w, 0)  
 jMax = min(j + w, n - 1)  
 # 当前模板所作用的区域为(iMin: iMax, jMin: jMax)  
 I = img[iMin:iMax, jMin:jMax] # 提取该区域的图像值赋给I  
 # 值域模板R  
 R = np.exp(-(I - img[i, j]) \*\* 2 / (2 \* sigma\_r \*\* 2))  
 # 两个模板相乘得到双边滤波器的模板F  
 F = R \* D[iMin - i + w + 1:iMax - i + w + 1, jMin - j + w + 1:jMax - j + w + 1]  
 filted\_img[i, j] = np.sum(F[:] \* I[:]) / np.sum(F[:]) # 对[i,j]点进行卷积滤波并装入filted\_img中  
 **return** filted\_img # 返回滤波后的图像  
  
  
img = cv2.imread(**'C:/Circuit\_noise.jpg'**, 0) # 直接读为灰度图像  
img = img / 255 # 归一化到0-1之间，减小计算量  
w = 3 # w是模板宽度的一半，构造一个3x3的模板  
sigma = [5, 0.4]  
sigma\_d = sigma[0] # σd为基于空间分布的高斯滤波函数的标准差  
sigma\_r = sigma[1] # σr为值域高斯函数的标准差  
x\_ticks = np.linspace(-w, w, 2 \* w + 1) # 模板横坐标-w到w  
y\_ticks = np.linspace(-w, w, 2 \* w + 1) # 模板纵坐标-w到w  
x, y = np.meshgrid(x\_ticks, y\_ticks) # 2\*w+1x2\*w+1的模板  
m, n = img.shape # 图片大小  
  
image = bfltGray(img, w, sigma\_d, sigma\_r) # 调用双边滤波函数  
# 写入时一定要乘255，因为之前归一化到0-1之间了  
cv2.imwrite(**"D:/filt.jpg"**, image \* 255, [int(cv2.IMWRITE\_JPEG\_QUALITY), 100]) # 保存图片质量100%  
cv2.imshow(**"filter\_img"**, image) # 显示图片  
cv2.waitKey(0)  
cv2.destroyAllWindows()

实现结果：

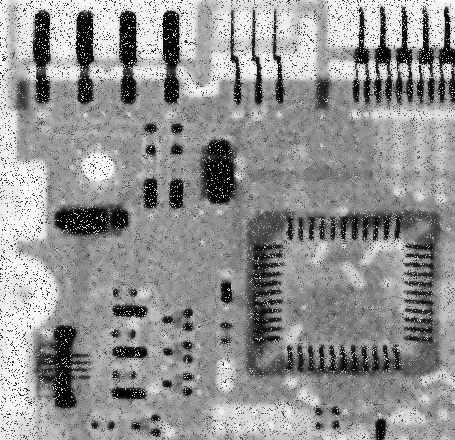
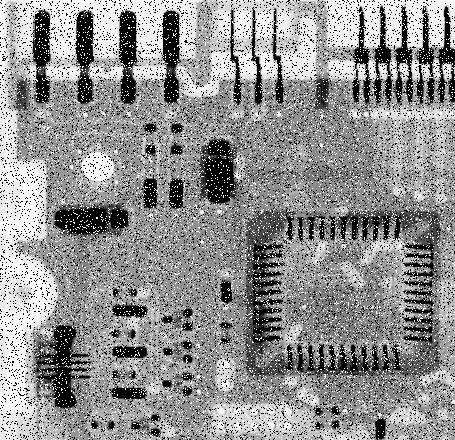


图1 原图像 图2 双边滤波后图像