

数字图像处理

第四次作业



摘要

本报告通过了解各种滤波器，以实现对图片进行平滑处理和边缘提取，用到的滤波器包括了中值滤波，高斯滤波。边缘检测用到了Sobel，Laplace以及Canny算子。

杨致远

电信钱51

2150405061

2018/4/19

## 任务1—中值滤波

**实验介绍：**

中值滤波的意思就是对一幅图像的某点进行中值变换，根据模板大小，将范围内的所有像素值按从小到大排列，取中值，用这个中间值去替换原来的像素值。

在处理的过程中我发现这次给的图片没有调色盘的值，因为并不是bmp格式，所以图片的读取就用最简单的方式就行了。

该程序完全由我自己编写，并与Matlab自带的函数进行了对比。我写的完全是一个活的程序，无论是图片的尺寸大小还是滤波器的大小都是动态可调的。

**结果展示：**













**结果讨论：**

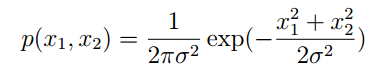
由上图可以看出，中值滤波器的窗口越大，平滑效果越明显，图像的细节越模糊。而自己编写的函数和matlab函数表现一致，验证了我代码的正确性。

与高斯滤波器的对比将在后面的结果中讨论。

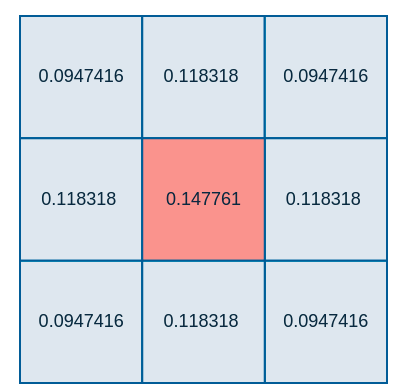
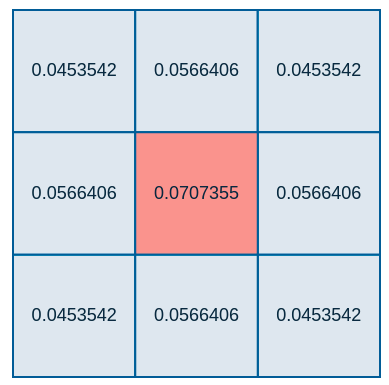
## 任务2—高斯滤波

**实验介绍：**

高斯滤波就不如前面的中值滤波容易了。计算所用到的是二维的高斯函数，见下。



所以我们首先需要确定一个滤波器模板，利用该公式将这个模板中的每一个位置都填入值，需要注意的是，Matlab的矩阵格式左上方是（1，1），而高斯公式带进去的值是（-1，1），所以需要做一个转换。这是3×3的高斯滤波器的矩阵值。



但这9个权重值和为0.47，所以还需要进行一步归一化，也就是这9个值都要除以0.47，然后就得到了一个新的矩阵。用这个新的矩阵去和原来的图片矩阵做卷积，最终就实现了高斯滤波。结果展示为原图，我自己写的程序，Matlab自带函数实现。

**结果展示：**













**结果讨论：**

通过以上对比，滤波器的平滑效果就不说了，这里着重强调中值滤波和高斯滤波的差异，可以发现高斯滤波更好的保留了图像的细节，图像的模糊程度低于中值滤波。而我自己的函数与现有函数对比差异不大，证明了我代码的正确性。

## 任务3—高通滤波器滤波

**实验介绍：**

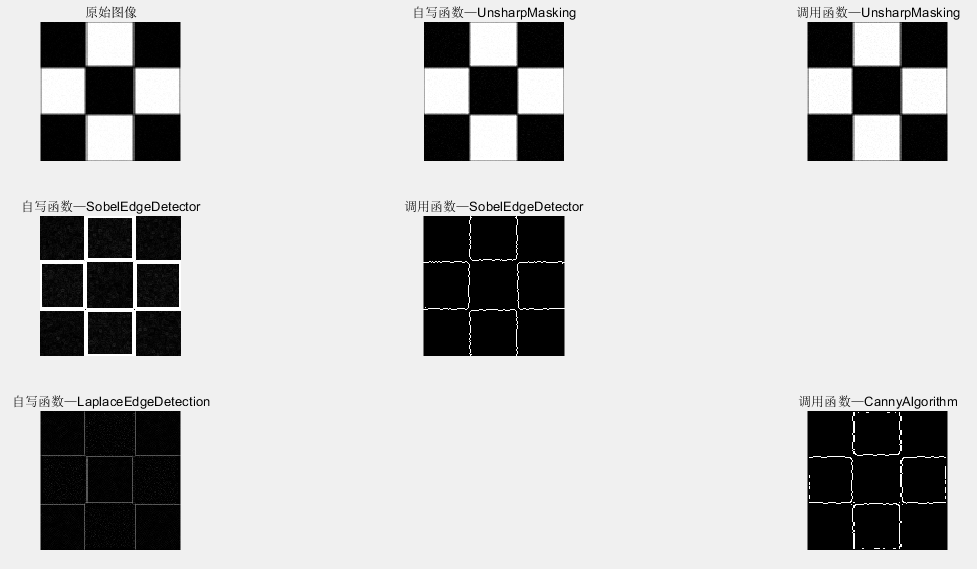
这个实验包括了4个部分

Unsharp masking：根据教材介绍的，需要先将原始图像进行一个模糊处理，然后将原图像与该模糊图像相减得到保留高频成分的图像，再将高频图像用参数放大后与原图像叠加，这就产生了一个增强了边缘的图像。

Sobel edge detector：首先需要求出两个横向和纵向的Sobel算子矩阵，然后用这两个算子去和原始图像做卷积。接下来用到的是一个近似计算，，得到的新矩阵G就是得到的边缘图像。

Laplace edge detection：拉普拉斯算子同上面一样也是用一个算子矩阵去和原图像做卷积，拉普拉斯本来是有一个方程的，但为了运算简单，用[0 1 0; 1 -4 1; 0 1 0]这样一个矩阵去做卷积是更好的选择。

**结果展示：**





**结果讨论：**

通过边缘的增强，图像的边缘显得更加清晰，但是引入了一些不需要的噪声。

三个边缘检测方法获得的效果各有差异，其中Canny算子的边缘提取较多，Laplace算子的边缘提取较少，Sobel算子的效果最好。

## 代码：

%中值滤波

clc;clear;

picture = imread('test1.pgm');

subplot(1,3,1);

imshow(picture);

title('原始图像7×7');

%%自己写的滤波器

blockSize = 7; %设置滤波器模板大小n\*n

pictureSize = size(picture); %获得图像的大小

pictureFilter = zeros(pictureSize(1),pictureSize(2));

pictureExpand = imresize(picture,[pictureSize(1) + ((blockSize - 1)),pictureSize(2) + ((blockSize - 1))],'bicubic'); %将原图像进行扩充

for i = 1:pictureSize(1)

for j = 1:pictureSize(2)

block = pictureExpand(i:i + blockSize - 1,j:j + blockSize - 1); %提取一个拓展后的图像块

block\_sort = sort(block(:)); %先将block中的每一个元素从小到大排列

value\_middle = block\_sort((blockSize^2 - 1) / 2); %然后取中值

pictureFilter(i,j) = value\_middle;

end

end

subplot(1,3,2);

imshow(uint8(pictureFilter));

title('自己写—中值滤波')

%%调用函数medfilt2()实现的中值滤波

picture1 = medfilt2(picture,[blockSize blockSize]);

subplot(1,3,3);

imshow(picture1);

title('调用函数—中值滤波');

%高斯滤波

clc;clear;

picture = imread('test2.tif');

subplot(1,3,1);

imshow(picture);

title('原始图像3×3');

%%自己写的函数

sigma = 1.5; %设定高斯滤波器的标准差

blockSize = 3; %设置滤波器模板大小n\*n

blockSize1 = (blockSize - 1) / 2; %做一个预处理，后面经常会用到

blockSize2 = (blockSize + 1) / 2; %做一个预处理，后面经常会用到

%%%生成高斯滤波器模板

sum1 = 0; %为了高斯模板归一化

gauss = @(x,y)(1 / (2 \* pi \* sigma^2)) \* exp(-(x^2 + y^2) / (2 \* sigma^2)); %二维高斯函数

gaussMatrix = zeros(blockSize,blockSize); %声明矩阵大小

for i = blockSize1 : -1 : -blockSize1 %倒着循环

for j = -blockSize1 : blockSize1 %例如3\*3的就是从-1到1

gaussMatrix(blockSize2 - i,j + blockSize2) = gauss(j,i);

sum1 = sum1 + gaussMatrix(blockSize2 - i,j + blockSize2);

end

end

gaussMatrix = gaussMatrix / sum1; %归一化处理

pictureSize = size(picture); %获得图像的大小

pictureFilter = zeros(pictureSize(1),pictureSize(2)); %滤波后的图像

pictureExpand = imresize(picture,[pictureSize(1) + ((blockSize - 1)),pictureSize(2) + ((blockSize - 1))],'bicubic'); %将原图像进行扩充

for i = 1:pictureSize(1)

for j = 1:pictureSize(2)

block = pictureExpand(i:i + blockSize - 1,j:j + blockSize - 1); %提取一个拓展后的图像块

value = gaussMatrix .\* double(block); %做卷积运算

pictureFilter(i,j) = sum(value(:));

end

end

subplot(1,3,2);

imshow(uint8(pictureFilter));

title('自己写—高斯滤波')

%%调用自带的函数

picture1 = imgaussfilt(picture, 1.5); %高斯滤波，指定标准差为1.5

subplot(1,3,3);

imshow(picture1);

title('调用函数—高斯滤波');

%%实验三

clc;clear;

picture = imread('test4 copy.bmp');

pictureSize = size(picture); %获得图像的大小

subplot(3,3,1);

imshow(picture);

title('原始图像');

% %Unsharp masking

picture1 = imgaussfilt(picture, 1.5); %高斯滤波，指定标准差为1.5

unsharpMask = picture - picture1;

picture2 = picture + unsharpMask;

subplot(3,3,2);

imshow(picture2);

title('自写函数—UnsharpMasking');

picture3 = imsharpen(picture);

subplot(3,3,3);

imshow(picture3);

title('调用函数—UnsharpMasking');

%Sobel edge detector

hx = [-1 0 1; -2 0 2; -1 0 1]; %横向算子

hy = [1 2 1; 0 0 0; -1 -2 -1]; %纵向算子

% picture = double(picture);

Gx = conv2(picture,hx,'same');

Gy = conv2(picture,hy,'same');

F = abs(Gx) + abs(Gy);

subplot(3,3,4);

imshow(uint8(F));

title('自写函数—SobelEdgeDetector');

picture4 = edge(picture,'sobel');

subplot(3,3,5);

imshow(picture4);

title('调用函数—SobelEdgeDetector');

%Laplace edge detection

laplaceModel = [0 1 0; 1 -4 1; 0 1 0];

laplaceTra = conv2(picture,laplaceModel,'same');

subplot(3,3,7);

imshow(uint8(laplaceTra));

title('自写函数—LaplaceEdgeDetection');

%Canny algorithm

picture5 = edge(picture,'canny');

subplot(3,3,9);

imshow(picture5);

title('调用函数—CannyAlgorithm');

## 参考文献

[1] 冈萨雷斯.数字图像处理（第三版）北京：电子工业出版社，2011

[2] 维基百科