Spatial Transforms and Filtering

Histogram equalization, Histogram matching, Local Histogram equalization and Image filtering

11712610 马思清

1. **Histogram equalization**

(1). Introduction

(2). Algorithm description

(3). The pseudo code

(4). Results

(5). Analysis and conclusions

1. **Histogram matching**

(1). Introduction

(2). Algorithm description

(3). The pseudo code

(4). Results

(5). Analysis and conclusions

1. **Local Histogram equalization**

(1). Introduction

(2). Algorithm description

(3). The pseudo code

(4). Results

(5). Analysis and conclusions

1. **Reduce SAP noise**
2. Introduction

Salt-and-pepper noise is a form of noise sometimes seen on images. It is caused by sudden disturbances in the image signal, where white and black pixels appear in the image. A type of image enhancement technique called linear spatial filtering can solve this problem. The linear spatial filtering use a type of defined templete called filter mask to do convolution with the image.Using different filter mask, we can realize different filtered result.

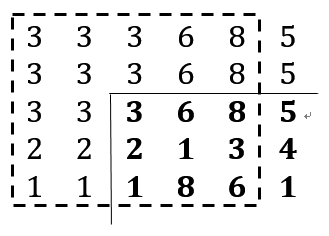
1. Algorithm description

The filter response is defined as g = ω ⅹ f. For every pixel in a picture, the filter response g(x, y) is the The product of the filter coefficient and the intensity of the pixel which was covered by the filter mask. The mathematical expression is shown as follow:

Where m = 2a+1, n = 2b+1, m and n are the dimension of the filter mask, and both m and n should be odd number, since we can easily define the filter centre. In this experiment, the filter mask is set to a square which means m equals n. A new parameter named *msize* is set to express the size of this square filter mask.

In this experiment, we use a averaging filter to do image enhancement. Here give a example of a averaging filter of size 3 x 3 (msize = 3):

For the pixel in the edge of the image, we will extend the picture to do calculation. Here give two example:



Filter mask

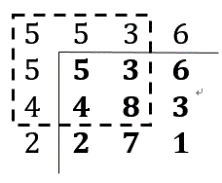


Image edge

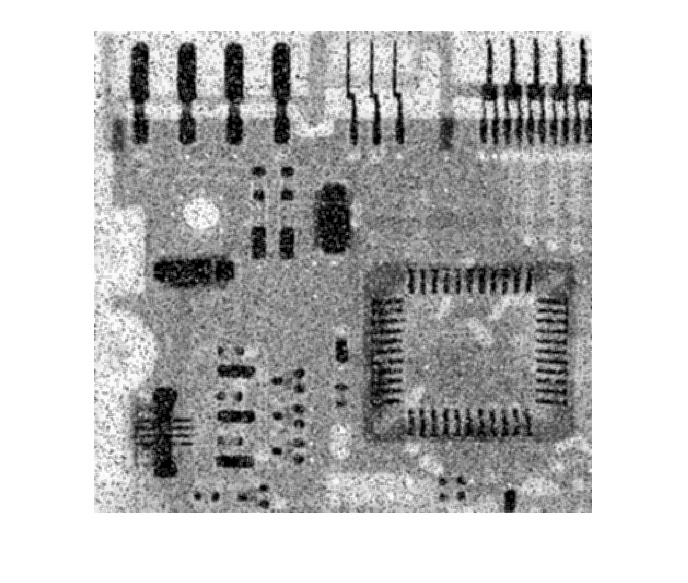
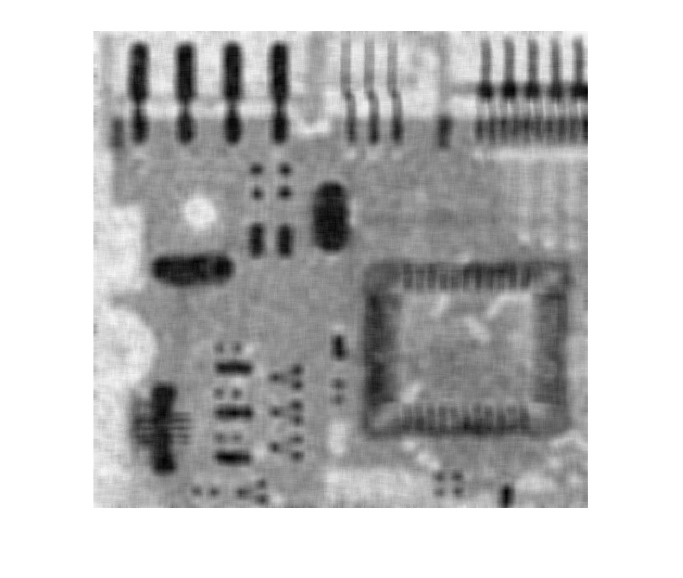
3 x 3 edge extension 5 x 5 edge extension

1. The pseudo code

* *Read the image file into a matrix I with size [m, n]*
* *Assign a new zero matrix OutputImage with the same size as I*
* *Extened I’s size to [m + msize -1, n + msize -1]*
* *For the extension part, interpolate by the nearest edge value*
* *For every pixel which not in the extension part of I*
* *use equation (1) to calculate the neighborhood sum, then do normalization*
* *then asssign it to the corresponding OutputImage pixel.*

***The MATLAB code is attached in appendix***

1. Results

D:\Documents B\Computer Vision\数字图像处理\Digital-Image-Processing-Template\Lab2\Q2_4.tif  

The origin image with SAP noise Enhanced picture with msize=3 Enhanced picture with msize=9

1. Analysis and conclusions

Actually, to get each pixel’s filtered value, what average linear spatial filtering really do is to calculate the sum of the neighborhood intensity and then do normalization, this give a result like a lowpass filter. By changing the size of the filter mask (msize), we find that when msize goes larger, the filtered image become more smoother and hazy, it seems that the cutoff frequency has become smaller.

1. **Appendix**

**Linear filter:**

**function[**OutputImage**]** **=** ReduceSAP\_11712610**(**InputImage**,** nsize**)**

image**=** im2double**(**imread**(**InputImage**));**

step **=** floor**(**nsize**/**2**);**

OutputImage **=** zeros**(**size**(**image**));**

left **=** **[];** right **=** **[];** upper **=** **[];** lower **=** **[];**

**for** i **=** 1 **:** step

left **=** image**(:,** 1**);**

right **=** image**(:,** **end);**

upper **=** **[**image**(**1**,** 1**),** image**(**1**,** **:),** image**(**1**,** **end)];**

lower **=** **[**image**(end,** 1**),** image**(end,** **:),** image**(end,** **end)];**

image **=** **[**left**,** image**,** right**];**

image **=** **[**upper**;** image**;** lower**];**

**end**

w **=** ones**(**nsize**,** nsize**);**

w **=** w**./**sum**(**sum**(**w**));**

**for** x1 **=** 1 **:** size**(**OutputImage**,** 1**)**

**for** y1 **=** 1 **:** size**(**OutputImage**,** 2**)**

sub\_image **=** image**(**x1 **:** x1**+**2 **\*** step**,** y1 **:** y1**+**2 **\*** step**);**

OutputImage**(**x1**,**y1**)** **=** sum**(**sum**(**sub\_image **.\*** w**));**

**end**

**end**