Introduction to Mathematical Image Processing
Image Processing Toolbox: Part 3
(spatial filtering and image denoising)

Suh-Yuh Yang (楊肅煜)

Department of Mathematics, National Central University
Jhongli District, Taoyuan City 32001, Taiwan

http://www.math.ncu.edu.tw/∼syyang/

Linear and non-linear spatial filtering

Spatial filtering techniques refer to those operations where *the resulting value of a pixel at a given coordinate is a function of the pixel values at that point and some of its neighbors.*

- **Linear filter:** the resulting output pixel is computed as a sum of products of the pixel values and mask coefficients in the pixel's neighborhood in the original image.

  Example: *Mean filter is an example of linear filter.*

- **Non-linear filter:** the resulting output pixel is selected from an ordered sequence of pixel values in the pixel's neighborhood in the original image.

  Example: *Median filter is an example of non-linear filter.*

Convolution

- *Convolution (卷積) and correlation (相關) are the two fundamental mathematical operations* involved in linear filters based on neighborhood-oriented image processing algorithms.
- Convolution processes an image by computing, for each pixel, a weighted sum of the values of that pixel and its neighbors.
- Different convolution masks produce different results when applied to the same input image. These operations are referred to as filtering operations and the masks as spatial filters.

Low-pass filters (LPFs)

- Low-pass filters (LPFs) are those spatial filters whose effect on the output image is equivalent to attenuating the high-frequency components and preserving the low-frequency components.
- High frequency: *fine details in the image.*

  Low frequency: *coarser details and homogeneous areas in the image.*
- These filters are typically used to either blur an image or reduce the amount of noise present in the image.
- *Linear LPFs can be implemented using 2D convolution masks with non-negative coefficients.*

High-pass filters (HPFs)

- A complementary way to LPFs. These preserve or enhance high-frequency components with the possible side-effect of enhancing noisy pixels as well.
- *High-frequency components include fine details, points, lines and edges.* In other words, these highlight transitions in intensity within the image.

  +————————————

- There are two in-built functions in MATLAB's Image Processing Toolbox (IPT) that can be used to implement 2D convolution: `conv2, filter2.`
- `conv2` computes 2D convolution between two matrices. `C = conv2(A, B)` computes the 2-D convolution of matrices A and B. If one of these matrices describes a 2-D *finite impulse response (FIR) filter,* the other matrix is filtered in two dimensions.

Command `filter2`

`filter2` function rotates the convolution mask, that is, 2D FIR filter, by 180° in each direction to create a convolution kernel and then calls `conv2` to perform the convolution operation.

- `Y=filter2(h,X)` filters the data in X with the 2-D FIR filter h. It computes the result Y using 2-D correlation, and returns the central part of the correlation that is of the same size as X.
- `Y=filter2(h,X,'shape')` returns the part of Y specified by the shape parameter. `shape` is a string with one of these values:
  (1) `full`: returns the full 2-D correlation. Y is larger than X.
  (2) `same`: (default) returns the central part of the correlation. Y is of the same size as X.
  (3) `valid`: returns only those parts of the correlation that are computed without zero-padded edges. Y is smaller than X.
- *Please see the M-file for an example:* `conv_filter.m`

Creating filters

One can create filter by hand or by using the `fspecial` function.

```
>> h=fspecial('type',parameters);
```

- `h` is the filter mask
- `'type'` is one of these:
  (1) `'average'`: averaging filter
  (2) `'disk'`: circular averaging filter
  (3) `'gaussian'`: Gaussian low-pass filter
  (4) `'laplacian'`: 2D Laplacian operator
  (5) `'log'`: Laplacian of Gaussian (LoG) filter
  (6) `'motion'`: approximates the linear motion of a camera
  (7) `'prewitt'` and `'sobel'`: edge-emphasizing filters
  (8) `'unsharp'`: unsharp contrast-enhancement filter
- `parameters`: optional and vary depending on the type of filter; e.g., mask size, standard deviation (for `'gaussian'` filter), etc.

Linear low-pass filters

The averaging filter *(spatial smoothing filter)* is a linear LPF implemented using `'average'` option in the `fspecial` function.

```
>>B=fspecial('average',[5,7]); % default [3,3], 3 × 3 matrix
```

```
    0.0286    0.0286    0.0286    0.0286    0.0286    0.0286    0.0286
    0.0286    0.0286    0.0286    0.0286    0.0286    0.0286    0.0286
    0.0286    0.0286    0.0286    0.0286    0.0286    0.0286    0.0286
    0.0286    0.0286    0.0286    0.0286    0.0286    0.0286    0.0286
    0.0286    0.0286    0.0286    0.0286    0.0286    0.0286    0.0286
```

```
>>A=imread('Penguins_gray.jpg');
>>B=fspecial('average');    % 3 × 3 matrix
>>C=filter2(B,A);   % C has data of type 'double.'
>>figure,imshow(A),figure,imshow(C/255)
```

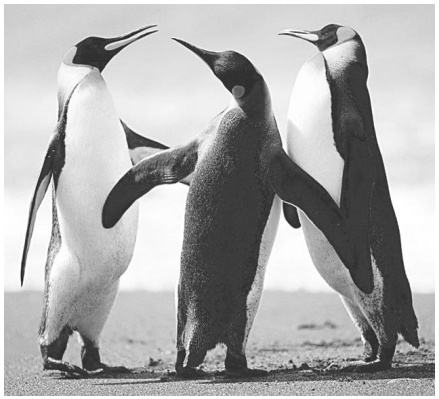


image A

filtered image C

Larger averaging filter

As you can see from the filtered image, the averaging filter blurs the image and the edges in the image are less distinct than in the original image. *Larger the size of the averaging filter, more is the blurring effect!*

```
>>A=imread('Penguins_gray.jpg');
>>B=fspecial('average',[9, 9]);    % 9 × 9 matrix
>>C=filter2(B,A);   % C has data of type 'double.'
>>figure,imshow(A),figure,imshow(C/255)
```

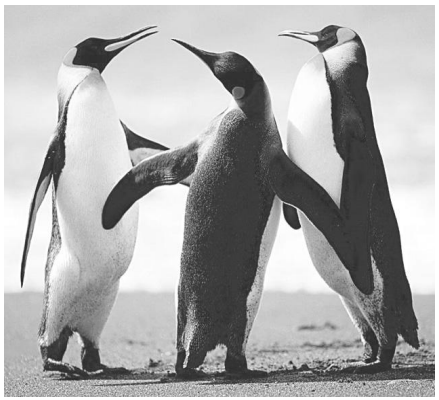


image A

filtered image C

The averaging filter operates on an $m \times n$ sliding window by calculating the average of all pixel values within the window and replacing the centre pixel value in the destination image with the result.

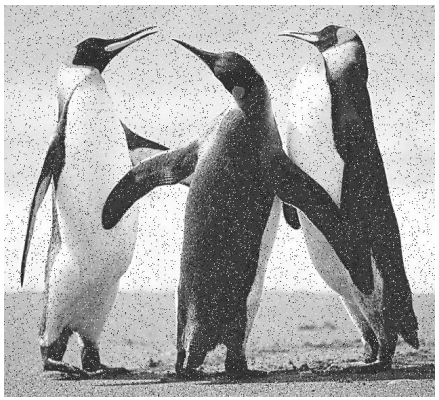
Reduction of noise

The averaging filter causes the blurring effect in the image. This helps in reduction of noise, especially Gaussian or uniform noise.

```
A=imread('Penguins_gray.jpg');
Noise_snp=imnoise(A,'salt & pepper'); % adding noise
```

Following commands remove the salt and pepper noise:

```
>> B=fspecial('average',[9,9]);
>> C=filter2(B,Noise_snp);
>> imshow(C/255)
```



noised image A

image denoising C

MRI image

Below are the MRI image A, noised image B with salt and pepper noise, and the image C after denoising:

```
>> A=imread('MRI_image.jpg');
>> B=imnoise(A,'salt & pepper');
>> E=fspecial('average', [9,9]);
>> D=filter2(E,B);
>> C=D/255;
>> imshow(A),figure,imshow(B),figure,imshow(C)
```

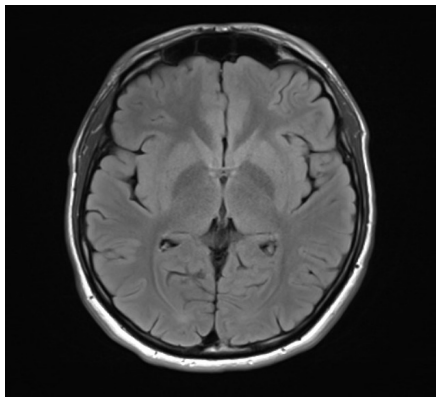


image A

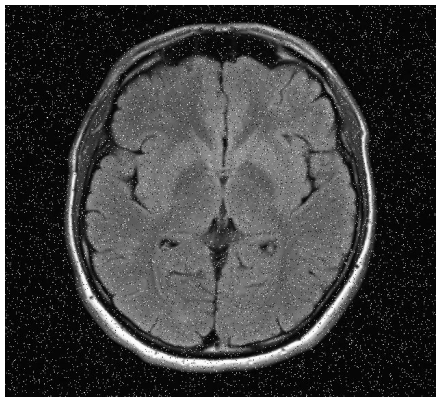


image B

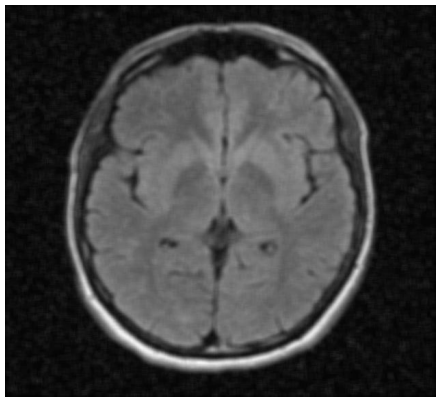


image C

Another type of linear filter: Gaussian filter

Gaussian filters are low-pass filters, based on the Gaussian probability distribution function:

$$f(x,y) = ae^{-\frac{x^2+y^2}{2\sigma^2}},$$

where σ is the standard deviation. Gaussian filters have a blurring effect which looks very similar to averaging.

```
>> A=imread('Penguins_gray.jpg');
>> F=fspecial('gaussian',[11,11], 5);
% The final parameter is the standard deviation σ, default 0.5
>> B=filter2(F,A)/255;
```

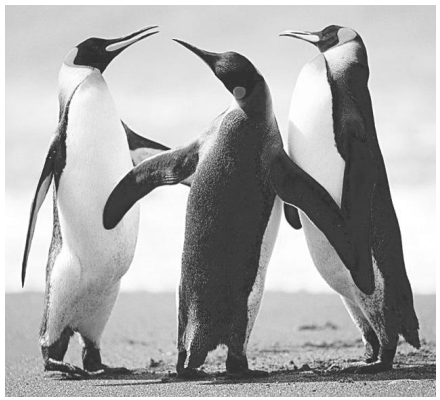
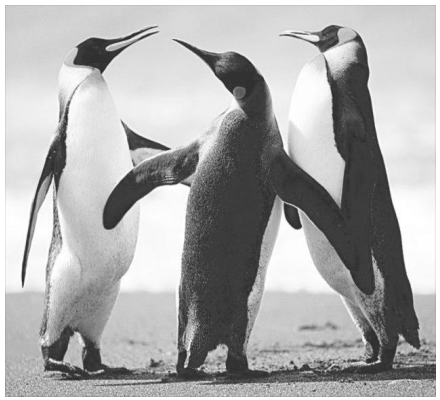


image A

$\sigma = 5$



$\sigma = 0.5$

Linear high-pass filters (HPFs)

High-pass filters are used for edge detection and edge enhancement operations. Edge extraction includes a variety of methods that aim at identifying points in an image at which the brightness changes sharply.

- **Laplacian filter:** Linear HPFs can be implemented using 2D convolution masks with positive and negative coefficients, which correspond to a digital approximation of the Laplacian. A typical convolution mask is given below:

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}.$$

- **Directional difference filters:** These filters are usually called emboss (凸起) filters. There are four representative masks that can be used to implement the emboss effect:

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & -1 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & 0 \end{bmatrix}.$$

Prewitt and Sobel operators for edge detection

Linear high-pass filters include Prewitt and Sobel operators. These are also referred to as first-order edge-detection methods. Following code extracts edges in the image using Prewitt operator:

```
>> A=imread('Penguins_gray.jpg');
>> B=edge(A,'Prewitt'); % or using 'Sobel'
>> imshow(B)
```

Let us now add Gaussian noise to the `Penguins_gray.jpg` image. We then process it using the Prewitt edge detector:

```
>> C=imnoise(A,'gaussian');
>> D=edge(C,'Prewitt'); % or using 'Sobel'
```

image B

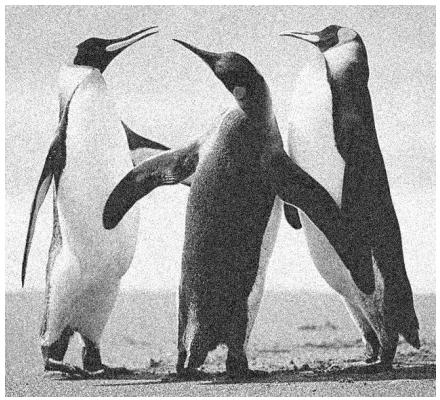


image C

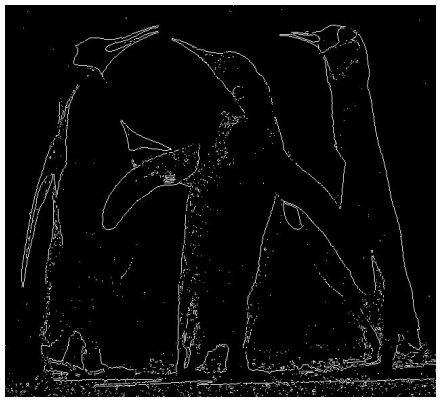


image D

Non-linear filters (sometimes called rank filters)

These usually apply *a ranking (sorting) function* to pixel values within the neighborhood and select a value from the sorted list. *Non-linear filters include, e.g., median filter, max filter, and min filter.*

Median filter works by sorting the pixel values within a neighborhood, finding the median value, and replacing the original pixel value with the median of that neighborhood.

Median filter has its own function, `medfilt2`, provided by the IPT.

```
>> A=imread('Penguins_gray.jpg');
>> B=imnoise(A,'salt & pepper');
>> C=medfilt2(B,[3,3]);
```

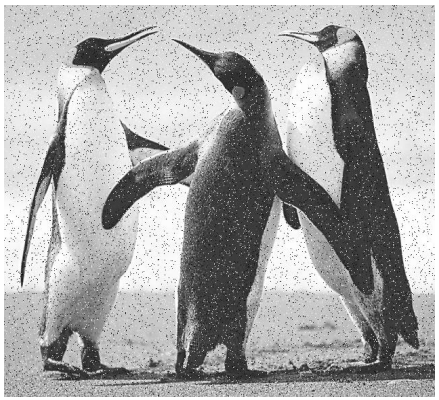


image B

median denoising C