Image filtering

In Fourier domain
- Linear filters

In spatial domain
- Non-linear filters
Image filtering in spectrum domain

\[ g(x,y) = \text{IF} \{ H(u,v) F\{f(x,y)\} \} \]
Ideal low-pass filter

\[ H(u, v) = \begin{cases} 1 & D(u, v) \leq D_0 \\ 0 & D(u, v) > D_0 \end{cases} \]

\[ D(u, v) = \sqrt{u^2 + v^2} \]
Image after ideal low-pass filtering, $D_0=70$
Image after ideal low-pass filtering, $D_0=10$
Ideal low-pass filter - example

- $D_0 = 10$
- $D_0 = 30$
- $D_0 = 70$

ringing
Butterworth filter

\[ H(u,v) = \frac{1}{1 + (\sqrt{2} - 1)\left[ \frac{D(u,v)}{D_0} \right]^{2n}} \]

\[ D(u,v) = \sqrt{u^2 + v^2}, \quad n = 1, 2, ... \]

\( n \)-filter order
Low-pass filtering

Butterworth filter

n = 1
Low-pass Butterworth filter - examples

\[ D_0 = 10 \]

\[ n = 1 \]

\[ D_0 = 30 \]

\[ D_0 = 70 \]
Ideal high-pass filter

\[ H(u,v) = \begin{cases} 
0 & D(u,v) \leq D_0 \\
1 & D(u,v) > D_0 
\end{cases} \]

\[ D(u,v) = \sqrt{u^2 + v^2} \]
Ideal high-pass filter - example

$D_0 = 10$  

$D_0 = 70$
Image after ideal high-pass filtering, $D_0=10$
High-pass Butterworth filter 2

\[ H(u,v) = \frac{1}{1 + (\sqrt{2} - 1) \left[ \frac{D_0}{D(u,v)} \right]^{2n}} \]

\[ D(u,v) = \sqrt{u^2 + v^2} \quad n = 1, 2, ... \]
High-pass Butterworth filter - examples

$D_0 = 10$  $\quad n = 1 \quad D_0 = 70$
Image after Butterworth high-pass filtering
High-pass Butterworth filter - examples

\[ n = 1 \]

\[ D_0 = 10 \]

\[ D_0 = 70 \]
Image filtering in spatial domain

\[ g(x,y) = IF\{ H(u,v) \ F\{f(x,y)\} \} = IF \{H(u,v)\} ** IF \{F \{f(x,y)\}\} = h(x,y)**f(x,y) \]
Filter definition in spatial domain

\[ IF \{ H(u,v) \} = h(x,y) \]

\[ \hat{h}(x,y) \]

\[ \hat{h} \text{ is selected so that } F(\hat{h}(x,y)) = \hat{H}(x,y) \approx H(x,y) \]
Image and the filer mask convolution

This is true for symmetric masks only!

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$h(-1,-1)$</td>
<td>$h(-1,0)$</td>
<td>$h(-1,1)$</td>
</tr>
<tr>
<td>$l(i-1,j-1)$</td>
<td>$l(i-1,j)$</td>
<td>$l(i-1,j+1)$</td>
</tr>
<tr>
<td>$h(0,-1)$</td>
<td>$h(0,0)$</td>
<td>$h(0,1)$</td>
</tr>
<tr>
<td>$l(i,j-1)$</td>
<td>$l(i,j)$</td>
<td>$l(i,j+1)$</td>
</tr>
<tr>
<td>$h(1,-1)$</td>
<td>$h(1,0)$</td>
<td>$h(1,1)$</td>
</tr>
<tr>
<td>$l(i+1,j-1)$</td>
<td>$l(i+1,j)$</td>
<td>$l(i+1,j+1)$</td>
</tr>
</tbody>
</table>

$$G(i,j) = l(i-1,j-1)h(-1,-1) + l(i-1,j)h(-1,0) + l(i-1,j+1)h(-1,1) + l(i,j-1)h(0,-1) + l(i,j)h(0,0) + l(i,j+1)h(0,1) + l(i+1,j-1)h(1,-1) + l(i+1,j)h(1,0) + l(i+1,j+1)h(1,1)$$
Computing the filtered image

<table>
<thead>
<tr>
<th>h</th>
<th>f(x,y)</th>
</tr>
</thead>
</table>

$g(x,y) = h(x,y)^{f(x,y)}$

source image $f$  
output image $g$
Boundary effects

source image $f$

output image $g$
Boundary effects – 3x3 mask

Boundary columns and rows of (N×N) image are neglected and the filtered image is of size \((N-2)\times(N-2)\).
f, g : array[0..N-1, 0..N-1] of byte;
{ size2 – half size of the mask }
h : array[-size2..size2,-size2..size2] of integer;
...
for i:=1 to N-2 do  for j:=1 to N-2 do 
begin
  g[i,j]:=0;
  for k:=-size2 to size2 do  for l:=-size2 to size2 do 
    g[i,j]:=g[i,j] + f[i+k,j+l] * h[i+k,j+l];
end;
...

Range check g[i,j] !!!
Low pass filter

\[ h_1 = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad h_2 = \frac{1}{25} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \]

Can one use mask of even size?
Frequency characteristics of low pass filters

for 5x5 mask

for 3x3 mask
Low-pass filtering the image

Source image

3x3 mask

5x5 mask
Gaussian filter

\[ h = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \]

\[ h(x, y) = e^{-\pi(x^2 + y^2)} \]

\[ H(u, v) = e^{-\pi d_0^2 (u^2 + v^2)} \]
Image filtering using the Gaussian filter

source image

filtered image
Image low-pass filters - examples

Image distorted by the Gaussian noise $N(0, 0.01)$

Low pass filter $3 \times 3$

Gaussian filter $3 \times 3$

Butterworth filter $D_0 = 50$
Image low-pass filters - examples

- Image distorted by the Gaussian noise $N(0, 0.01)$
- low-pass filter 5x5
- Gaussian filter 5x5
- Butterworth filter $D_0=30$
Image low-pass filters - examples

- Image distorted by the Gaussian noise $N(0, 0.002)$
- Low pass filter 3x3
- Gaussian filter 3x3
- Butterworth filter $D_0=50$
High-pass filters (derivative filters)

$$h_1 = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \quad h_2 = \begin{bmatrix} 0.17 & 0.67 & 0.17 \\ 0.67 & -3.33 & 0.67 \\ 0.17 & 0.67 & 0.17 \end{bmatrix}$$

[Graphs showing frequency response with labels: laplacian]
High-pass filtering the image

mask $h_1$

mask $h_2$
The „high boost” filter

\[ f(x, y) = f_L(x, y) + f_H(x, y) \]

\[ f_{HB}(x, y) = Af(x, y) - f_L(x, y) = \]
\[ = (A - 1)f(x, y) + f(x, y) - f_L(x, y) = \]
\[ = (A - 1)f(x, y) + f_H(x, y), \quad A \geq 1 \]

\[ h_{HB} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9A - 1 & -1 \\ -1 & -1 & -1 \end{bmatrix} \]

\[ A=? \]
High boost filter - example

Laplace filter

A=1.1
A=1.5
In order to keep the average value of the image
add 1 do the centre element of the Laplace mask
Other high-pass filters

\[ h'_3 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} \]

\[ h'_4 = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix} \]
High-pass filters

Blurred image

Sharpened image

MATLAB

out_image = filter2(filter_mask, in_image);
Nonlinear filters

The filtered image is defined by a non-linear function of the source image.

Can we compute spectral characteristics for nonlinear filters?

**NO**

Because transfer characteristics of nonlinear filters depend on image content itself!
The median $m$ of a set of values (e.g. image pixels in the filtering mask) is such that half the elements in the set are less than $m$ and other half are greater than $m$.

\[ x(n) = \{1, 5, -7, 101, -25, 3, 0, 11, 7\} \]

Sorted sequence of elements:

\[ x_s(n) = \{-25, -7, 0, 1, 3, 5, 7, 11, 101\} \]
Median filtering the image

source image $f$

output image $g$

$g(x,y) = \text{median}\{f(x,y); \ (x,y) \in h\}$
Buble-sort algorithm

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>(iterations)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>06</td>
<td>06</td>
<td>06</td>
<td>06</td>
<td>06</td>
<td>06</td>
<td>06</td>
<td>06</td>
<td></td>
</tr>
<tr>
<td>55</td>
<td>44</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>55</td>
<td>44</td>
<td>18</td>
<td>18</td>
<td>18</td>
<td>18</td>
<td>18</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>12</td>
<td>55</td>
<td>44</td>
<td>42</td>
<td>42</td>
<td>42</td>
<td>42</td>
<td>42</td>
<td></td>
</tr>
<tr>
<td>94</td>
<td>42</td>
<td>18</td>
<td>55</td>
<td>44</td>
<td>44</td>
<td>44</td>
<td>44</td>
<td>44</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>94</td>
<td>42</td>
<td>42</td>
<td>55</td>
<td>55</td>
<td>55</td>
<td>55</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>06</td>
<td>18</td>
<td>94</td>
<td>67</td>
<td>67</td>
<td>67</td>
<td>67</td>
<td>67</td>
<td>67</td>
<td></td>
</tr>
<tr>
<td>67</td>
<td>67</td>
<td>67</td>
<td>94</td>
<td>94</td>
<td>94</td>
<td>94</td>
<td>94</td>
<td>94</td>
<td></td>
</tr>
</tbody>
</table>

Unsorted sequence

© N. Wirth, „Algorithms+Data Structures=Programs”
a[k], k=1..N – unsorted sequence
for i:=2 to N do
begin
    for j:=N downto i do
    if a[j-1]>a[j] then
    begin
        x=a[j-1]; a[j-1]:=a[j]; a[j]:=x;
    end;
end;
Demo – median filter

Source image distorted by „salt and pepper noise”

Enhanced image using the median filter (3x3)

%MATLAB
out_image = medfilt2(in_image, [m n]);
Median filter:

1. Excellent in reducing impulsive noise (odd size smaller than half size of the filtering mask)

2. Keeps sharpness of image edges (as opposed to linear smoothing filters)

3. Values of the output image are equal or smaller than the values of the input image (no rescaling)

4. Large computing cost involved
Median filter

[1 x 3] 1/3*[1 1 1]

median

average