Basic Algorithms for Digital Image Analysis

Dmitrij Csetverikov

Eötvös Loránd University, Budapest, Hungary
csetverikov@inf.elte.hu
csetverikov@sztaki.hu

Faculty of Informatics

Image filters

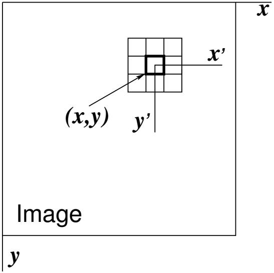
1. Image filtering
   - Correlation and convolution
   - Basics of noise filtering
2. Frequently used filters
   - Linear smoothing filters
   - Median filter
   - Laplace filter
   - Unsharp masking
3. Fast and adaptive filters
   - Separable filters
   - Run filtering
   - Adaptive noise filtering

Neighbourhood operators

- Output value in (x, y) determined by neighbourhood of (x, y):

$$g(x, y) = T[f(x, y)]$$

  - $f(x, y)$ is input, $g(x, y)$ output image
  - T is operator on f, defined over neighbourhood of (x, y)
- Window sampling (observation) assuming *local* dependence between pixels
  - correlation decreases with distance
  - not true for periodic patterns



3×3 window
in point (x, y)
x', y': local coord.

Non-recursive and recursive operators

- **Non-recursive** neighbourhood operator
  - output only depends on input image neighbourhood
  - output separated from input: input *not* modified during operation
  - action limited to neighbourhood
- **Recursive** neighbourhood operator
  - output depends in part on previously generated output values
  - output *not* separated from input: input modified during operation
  - action extends beyond neighbourhood
  - useful but much more complicated
- We only consider non-recursive operators

General non-recursive neighbourhood operator

$$g(x, y) = \phi[x, y, f(x', y') : (x', y') \in N(x, y)]$$

- $f(x, y)$ is input image, $g(x, y)$ output image
- $N(x, y)$ is neighbourhood of point (x, y)
- (x', y') are *local coordinates* within $N(x, y)$
- $f(x', y') : (x', y') \in N(x, y)$ is list of pixel values in $N(x, y)$
  - scan $N(x, y)$ in certain order
  - for each $(x', y') \in N(x, y)$, pick $f(x', y')$ and place into list
- ϕ may depend on position (x, y) within input image
  - neighbourhood $N(x, y)$ may depend on (x, y)
  - procedure computing output value may depend on (x, y)
- ϕ may be nonlinear
  - linear operator A: $A(\alpha p + \beta q) = \alpha A p + \beta A q$

Outline

1. Image filtering
   - Correlation and convolution
   - Basics of noise filtering
2. Frequently used filters
   - Linear smoothing filters
   - Median filter
   - Laplace filter
   - Unsharp masking
3. Fast and adaptive filters
   - Separable filters
   - Run filtering
   - Adaptive noise filtering

Correlation

Linear shift-invariant operator is linear combination of input pixels: **cross-correlation** of image *f* with mask *w*

$$g(x, y) = (f \otimes w)(x, y) \doteq \sum_{\substack{(x',y') \in W \\ (x+x',y+y') \in F}} f(x + x', y + y') \cdot w(x', y')$$

- W is set of positions in window, F in image
- neighbourhood W and weights $w(x', y')$ are shift-invariant
- w called *kernel* or *mask* of weights

Convolution

Convolution of image f with kernel w:

$$g(x,y) = (f * w)(x,y) \doteq \sum_{\substack{(x',y') \in W \\ (x-x',y-y') \in F}} f(x-x', y-y') \cdot w(x',y')$$

- Window W is scanned in reversed order.
- We will work with symmetric masks.
  - $\Rightarrow$ no difference between correlation and convolution

Basic properties of convolution

1. Correlation is convolution by reflected mask:
   $f \otimes w = f * w^{\sim}$
   - $w^{\sim}(x, y) \doteq w(-x, -y)$ is reflection of w
2. Commutative: $w * v = v * w$ (order is arbitrary)
3. Associative: $(f * w) * v = f * (w * v)$
4. Distributive: $(f + g) * w = f * w + g * w$
5. Homogeneousi: $(\alpha f) * w = \alpha(f * w)$ for any constant α
6. Reflection of composition: $(w * v)^{\sim} = w^{\sim} * v^{\sim}$

- f and g are images, w and v masks
- $w * v$: mask w is treated as image and convolved with v
  - result is a larger mask
  - associativity can be used to speed up filtering

Examples: 3 × 3 mean filters

$$\frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}\quad\frac{1}{6}\begin{bmatrix}0&1&0\\1&2&1\\0&1&0\end{bmatrix}\quad\frac{1}{8}\begin{bmatrix}0&1&0\\1&4&1\\0&1&0\end{bmatrix}\quad\frac{1}{16}\begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix}$$

box filter   mean filter 1   mean filter 2   mean filter 3

- *Box filter*: mean filter with uniform weights
- Otherwise, weights decrease with distance from center
  - contribution to result decreases with distance
- Normalising factors are sums of mask coefficients
  - output range: [minval, maxval]
- Filter size is normally odd

5 × 5 mean filters 1/2

$$\begin{bmatrix} 1 & 2 & 3 & 2 & 1 \\ 2 & 4 & 6 & 4 & 2 \\ 3 & 6 & 9 & 6 & 3 \\ 2 & 4 & 6 & 4 & 2 \\ 1 & 2 & 3 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

- This 5×5 filter is convolution of two 3×3 filters.
- Allows for faster implementation:
  - 5×5 filter: $5 \times 5 = 25$ multiplications, 24 additions
  - two 3×3 filters: $2 \times 3 \times 3 = 18$ multiplications, $2 \times 8 = 16$ additions

5 × 5 mean filters 2/2

$$\frac{1}{100}\begin{bmatrix} 0 & 3 & 4 & 3 & 0 \\ 3 & 6 & 7 & 6 & 3 \\ 4 & 7 & \mathbf{8} & 7 & 4 \\ 3 & 6 & 7 & 6 & 3 \\ 0 & 3 & 4 & 3 & 0 \end{bmatrix}$$

- This filter is discrete version of

$$w(r) = 8 - r^2,$$

where $r = \sqrt{x^2 + y^2}$ is distance from center (**8**).

  - for example, $4 = 8 - 2^2$
  - note rotation symmetry

Application of convolution filter: numerical example 1/2

$$\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix} * \begin{array}{|c|c|c|c|c|c|}\hline \mathbf{3} & \mathbf{2} & \mathbf{8} & 7 & 8 & 8\\ \hline \mathbf{2} & \mathbf{2} & \mathbf{7} & 8 & 7 & 7\\ \hline \mathbf{2} & \mathbf{3} & \mathbf{9} & 9 & 8 & 8\\ \hline 1 & 2 & 9 & 9 & 7 & 8\\ \hline 2 & 2 & 8 & 8 & 8 & 8\\ \hline 2 & 3 & 7 & 7 & 9 & 7\\ \hline \end{array} = \begin{array}{|c|c|c|c|c|c|}\hline - & - & - & - & - & -\\ \hline - & 4 & .. & .. & .. & -\\ \hline - & .. & .. & .. & .. & -\\ \hline - & .. & .. & .. & .. & -\\ \hline - & .. & .. & .. & .. & -\\ \hline - & - & - & - & - & -\\ \hline \end{array}$$

$$\frac{1\cdot 3+2\cdot 2+1\cdot 8+2\cdot 2+4\cdot 2+2\cdot 7+1\cdot 2+2\cdot 3+1\cdot 9}{16}=\frac{58}{16}\approx 4$$

- Current (initial) position of filter in input image is in bold.
- Result is written in corresponding position in output image.

Application of convolution filter: numerical example 2/2

$$\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix} * \begin{array}{|c|c|c|c|c|c|}\hline 3 & \mathbf{2} & \mathbf{8} & \mathbf{7} & 8 & 8\\ \hline 2 & \mathbf{2} & \mathbf{7} & \mathbf{8} & 7 & 7\\ \hline 2 & \mathbf{3} & \mathbf{9} & \mathbf{9} & 8 & 8\\ \hline 1 & 2 & 9 & 9 & 7 & 8\\ \hline 2 & 2 & 8 & 8 & 8 & 8\\ \hline 2 & 3 & 7 & 7 & 9 & 7\\ \hline\end{array} = \begin{array}{|c|c|c|c|c|c|}\hline - & - & - & - & - & -\\ \hline - & 4 & 6 & .. & .. & -\\ \hline - & .. & .. & .. & .. & -\\ \hline - & .. & .. & .. & .. & -\\ \hline - & .. & .. & .. & .. & -\\ \hline - & - & - & - & - & -\\ \hline\end{array}$$

- Current (next) position of filter in input image is in bold.
- Result is written in corresponding position of output image.
- Input and output are separate matrices!

Handling border pixels

- For $D_W \times D_W$ mask, width of border margin is $\lfloor D_W/2 \rfloor$ (odd D_W)
  - $\Rightarrow$ margin grows with filter size

Options:

- Fill with zeros
  - may introduce strong artificial edges
  - may disturb greyscale normalisation (rescaling to [0,255])
- Fill with the mean value of output image
  - less strong artificial edges
  - does not influence grey-scale normalisation
- Fill with nearest computed value
- Treat input image as periodic (like cylinder), compute result for all pixels

Outline

1. Image filtering
   - Correlation and convolution
   - Basics of noise filtering
2. Frequently used filters
   - Linear smoothing filters
   - Median filter
   - Laplace filter
   - Unsharp masking
3. Fast and adaptive filters
   - Separable filters
   - Run filtering
   - Adaptive noise filtering

Types of noise

- **Additive picture-independent** (white) noise:

$$g(x, y) = f(x, y) + v(x, y)$$

  - $f(x, y)$ is input, $g(x, y)$ output image, $v(x, y)$ noise
  - typical channel (transmission) noise

- **Uncorrelated multiplicative noise**:

$$g(x, y) = f(x, y) \cdot v(x, y)$$

  - amplitude modulation (variation)
  - typical for TV raster lines

- **Quantisation noise** (error):

$$v_{noise}(x, y) = g_{quantised}(x, y) - f_{original}(x, y)$$

- **Salt-and-pepper, or peak noise**: Pointwise, uncorrelated random noise

Heuristic noise filtering

- Image enhancement often means ‘heuristic’ image restoration
  - no explicit noise model assumed
- However, different filters are best suitable for different types of noise
  - *mean filter* for additive zero-mean noise
  - *median filter* for salt-and-pepper noise

⇒ Analysis of noise is desirable

- Small groups of noisy pixels are easier to remove
  - good estimate of noise-free value when ‘good’ pixels dominate in window
  - bad estimate when noisy values dominate

Outline

1. Image filtering
   - Correlation and convolution
   - Basics of noise filtering
2. Frequently used filters
   - Linear smoothing filters
   - Median filter
   - Laplace filter
   - Unsharp masking
3. Fast and adaptive filters
   - Separable filters
   - Run filtering
   - Adaptive noise filtering

Mean filter and box filter

Mean filter:

- Spatial averaging (smoothing) filter
- Non-negative weights that sum to 1

$$0 \leq w_{mean}(x, y) \leq 1, \quad \sum_{x,y} w_{mean}(x, y) = 1$$

  - in practice, use integer weights, then normalise

- Weights do not grow with distance from filter center:

$$w_{mean}(x_1, y_1) \leq w_{mean}(x_2, y_2), \text{ if } x_1^2 + y_1^2 > x_2^2 + y_2^2$$

Box filter

- Mean filter with uniform weights
  - simplest and fastest mean filter
- For $(2M+1) \times (2N+1)$ size window

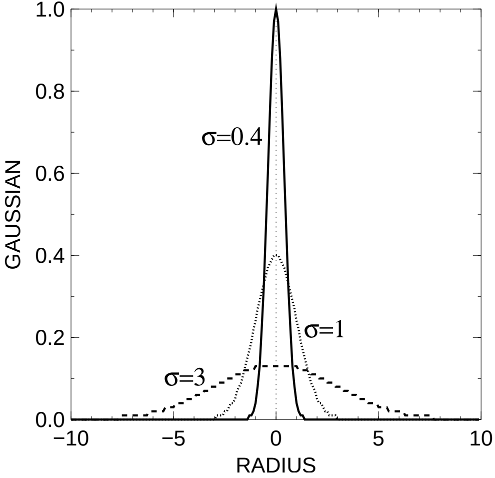
$$g(x,y) = \frac{1}{(2M+1) \times (2N+1)} \sum_{x'=-M}^{M} \sum_{y'=-N}^{N} f(x+x', y+y')$$

Gaussian filter 1/2

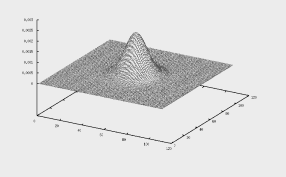
$$w_G(x,y) = \frac{1}{\sum\limits_{(x,y)\in W} e^{-\frac{r^2(x,y)}{2\sigma^2}}} e^{-\frac{r^2(x,y)}{2\sigma^2}}$$

- Weights provided by 2D Gaussian (normal) distribution function.
- $r^2(x,y) = x^2 + y^2$ is squared distance from mask center
  - does not depenge on angle, on r only
  - bell-like, rotation-symmetric shape
- Parameter σ controls size of filter
  - larger $\sigma \Rightarrow$ larger filter and stronger smoothing

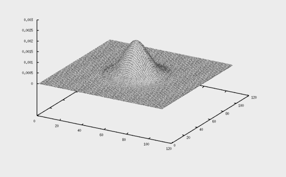
Shape of Gaussian filter for growing σ-ra: 2D



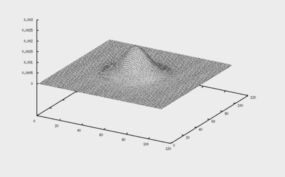
Shape of Gaussian filter for growing σ-ra: 3D



$\sigma = 9$



$\sigma = 10$



$\sigma = 11$

Gaussian filter 2/2

$$w_G(x,y) = \frac{1}{\sum\limits_{(x,y)\in W} e^{-\frac{x^2+y^2}{2\sigma^2}}} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

- When discretised, $w_G(r)$ is cut at $r_{max} = k\sigma$.
  - typically, $k = 2.5$
  - includes most of bell volume

- Gaussian filter is separable:

$$w_G(x,y) = w_G(x) \cdot w_G(y) \Leftarrow \exp(a+b) = \exp(a) \cdot \exp(b)$$

  - fast implementation: two 1D filters instead of one 2D filter
  - $O\left((2r_{max})^2\right)$ ops in 2D, $O(2 \cdot 2r_{max})$ ops in 1D

Use of smoothing

- **Noise filtering**
  - box filter reduces zero-mean white noise as positive and negative values nullify each other
  - large filter size ⇒ greater noise reduction
- Removing fine details
- **Subsampling**: going to lower resolution
  - average, then decimate (discard rows/columns)
- Obtaining **scale-space** representation of image
  - sequence of Gaussian-filtered images for growing σ
  - image analysis at varying degree of detail

Basic properties of smoothing

- Decreases contrast and blurs edges
- Output greylevel range is within input range
- Can produce new greyvalues that did not exist in input
  - smoothing binary image gives greyscale image
- Outliers can strongly affect mean value

  ⇒ mean is not robust
  - outliers are wrong values, such as peak noise
- Number of operations required by box filter
  - direct implementation: $O(N \cdot N_W)$
  - run filter implementation: $O(N)$
  - N is image size (area), N_W window size

Outline

1. Image filtering
   - Correlation and convolution
   - Basics of noise filtering
2. Frequently used filters
   - Linear smoothing filters
   - Median filter
   - Laplace filter
   - Unsharp masking
3. Fast and adaptive filters
   - Separable filters
   - Run filtering
   - Adaptive noise filtering

Nonlinear median filter

- Median filter outputs median of greyvalues in window:
  - sort (rank) the pixels by greyvalue
  - select value which is in centre (middle) of sorted sequence
  - normally, window size is odd: 3 × 3, 5 × 5, etc.
- Example:
  - nine greyvalues in 3 × 3 window are

$$(1, 1, 3, 2, 5, 4, 4, 12, 11)$$

  - the ordered sequence is

$$(1, 1, 2, 3, \mathbf{4}, 4, 5, 11, 12)$$

  - median value is **4**

Properties of median 1/2

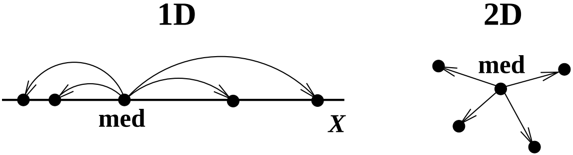
- Calculating the median is *non-linear* operation: For two sequences P and Q,

  $$Med(\alpha P) = \alpha Med(P) \;\text{ but }\; Med(P+Q) \neq Med(P)+Med(Q)$$

- Selecting the median can be viewed as *voting procedure*
  - during sorting, each pixels votes for a grayvalue
  - median is selected from majority, from the ’middle’
  - extremal values are rejected as not belonging to majority

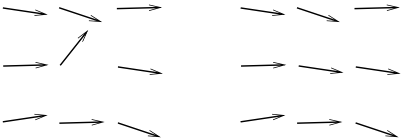
Properties of median 2/2

- Median is a *robust statistics*
  - outliers do not bias result
  - the *breakdown point* is when outliers form 50% or more
- Consider numbers as points on X. Sum of distances from median to other points is minimal for any 1D point set
  - in other words, median is the *innermost* point of set
  - this property is equivalent to definition of median
  - used to extend median to higher dimensions, **vectors**



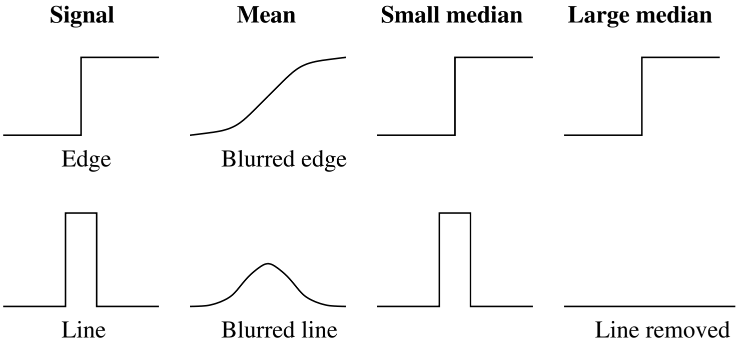
Properties of median filter

- Removes isolated noise pixels
- Does not blur image, but rounds off corners
- Removes thin lines when *filtersize* $> 2 \times$ *linewidth*
  - background pixels form majority
- Number of operations required
  - rirect implementation: $O(N \cdot N_W \cdot \log N_W)$
  - run filter implementation: $O(N \cdot \log N_W)$
- Vector median filter enhances vector fields
  - removes vectors incompatible with surrounding vectors



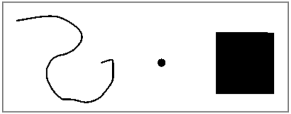
before filtering after filtering

Mean and median filtering of step edge and line

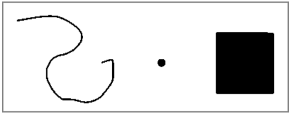


- Line removed when median filter size exceeds twice the line width

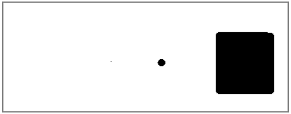
Comparing median and box filters for bilevel image



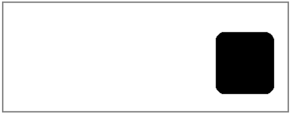
input image



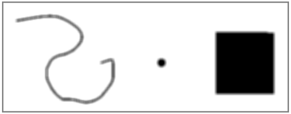
median 3 × 3



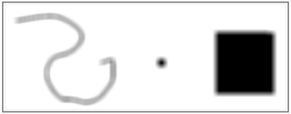
median 7 × 7



median 17 × 17

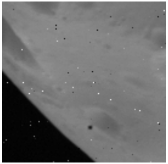


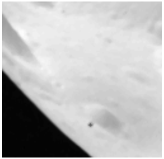
box 5 × 5

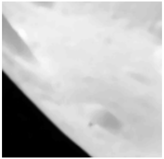


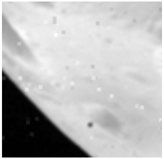
box 9 × 9

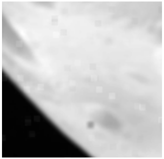
Comparison for image with salt-and-pepper noise

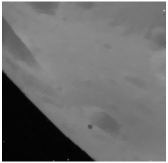

input image


median 3 × 3


median 5 × 5


box 3 × 3


box 5 × 5


symm. box 5 × 5

- The images are gray-scale normalised
- symm. box: adaptive symmetric box filter

Outline

1. Image filtering
   - Correlation and convolution
   - Basics of noise filtering
2. Frequently used filters
   - Linear smoothing filters
   - Median filter
   - Laplace filter
   - Unsharp masking
3. Fast and adaptive filters
   - Separable filters
   - Run filtering
   - Adaptive noise filtering

Laplace operator and its approximation

- Definition of Laplace operator

$$g(x,y) = \Delta f(x,y) \doteq \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) f$$

- Design simple 3 × 3 kernel w_L for Laplace operator
  - approximate derivatives by differences

$$\frac{\partial f}{\partial x} \longrightarrow \begin{array}{|c|c|c|}\hline -1 & 1 & 0 \\ \hline \end{array}$$

$$\frac{\partial^2 f}{\partial x^2} \longrightarrow \begin{array}{|c|c|c|}\hline -1 & 1 & 0 \\ \hline \end{array} - \begin{array}{|c|c|c|}\hline 0 & -1 & 1 \\ \hline \end{array} = \begin{array}{|c|c|c|}\hline -1 & 2 & -1 \\ \hline \end{array}$$

$$\Delta f \longrightarrow \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline -1 & 2 & -1 \\ \hline 0 & 0 & 0 \\ \hline \end{array} + \begin{array}{|c|c|c|}\hline 0 & -1 & 0 \\ \hline 0 & 2 & 0 \\ \hline 0 & -1 & 0 \\ \hline \end{array} = \begin{array}{|c|c|c|}\hline 0 & -1 & 0 \\ \hline -1 & 4 & -1 \\ \hline 0 & -1 & 0 \\ \hline \end{array}$$

Laplace filter and averaging

Normalising the kernel by 4, we have

$$w_L = \Delta f(x, y) \approx f(x, y) - Av(x, y),$$

where *Av* is average of four neighbours

$$Av(x, y) \doteq \frac{1}{4}\Big[f(x-1, y) + f(x, y-1) + f(x+1, y) + f(x, y+1)\Big]$$

$$\frac{1}{4}\begin{array}{|c|c|c|}\hline 0 & -1 & 0 \\ \hline -1 & 4 & -1 \\ \hline 0 & -1 & 0 \\ \hline \end{array}$$

4-neighbour version

$$\frac{1}{8}\begin{array}{|c|c|c|}\hline -1 & -1 & -1 \\ \hline -1 & 8 & -1 \\ \hline -1 & -1 & -1 \\ \hline \end{array}$$

8-neighbour version

Simple masks used for Laplace filtering

Properties of Laplace filter 1/2

- Close to difference of original image and smoothed image
  - gradual variations subtracted, fine variations remain
  - zero response to non-varying parts of image
- Formally, output range is $[-255, 255]$
  - difference between pixel and its neighbours is small

  $\Rightarrow$ in practice, range is narrow
- Enhances intensity variations, fine details
  - contours, spots, thin lines

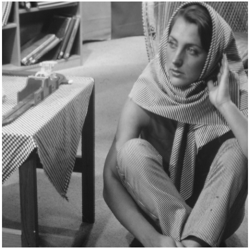
Properties of Laplace filter 2/2

- Noise-sensitive: contains second order derivatives
- Usually, used in combinations with smoothing filters
- Laplacian-of-Gaussian (LoG)

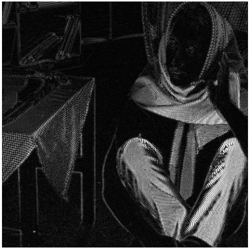
$$w_{LoG} = w_G * w_L$$

  - obtain smooth function before taking derivatives
  - less noise-sensitive than Laplace filter
  - zero-crossings of LoG are *edges*

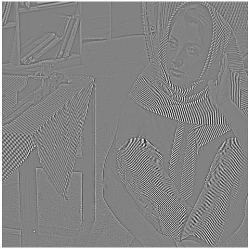
Examples of Laplace filtering 1/3



input



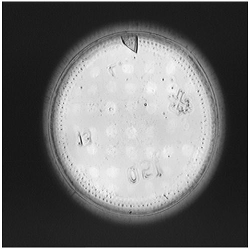
Laplace absolute

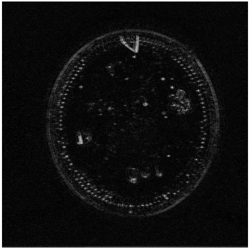


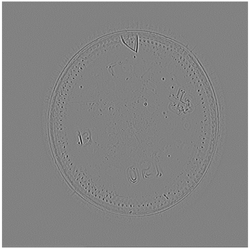
Laplace shift

- Two different visualisations of output are shown
  - absolute value mapping: $-127 \rightarrow 127$, $127 \rightarrow 127$
  - shifted value mapping: $-127 \rightarrow 0$, $127 \rightarrow 254$
- Depending on mapping, different details are visible

Examples of Laplace filtering 2/3

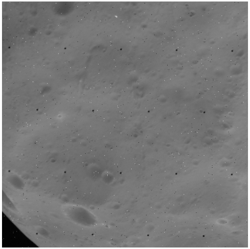

input


Laplace absolute


Laplace shift

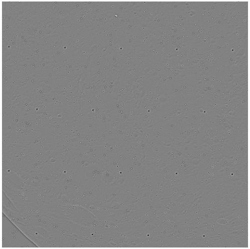
- Fine details are enhanced, including piece of glass and symbols
- Gradual variations are suppressed

Examples of Laplace filtering 3/3



input

Laplace absolute



Laplace shift

- Laplace filter is noise-sensitive
- For peak-noisy input without contrast details, the output is mostly noise

Outline

1. Image filtering
   - Correlation and convolution
   - Basics of noise filtering
2. Frequently used filters
   - Linear smoothing filters
   - Median filter
   - Laplace filter
   - Unsharp masking
3. Fast and adaptive filters
   - Separable filters
   - Run filtering
   - Adaptive noise filtering

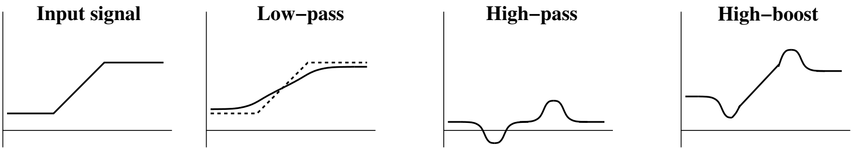
Unsharp masking filter

- Goal: Enhance contours and other high-frequency features
- Solution: Add to input image a part of Laplace output
  - Laplace filter amplifies image variations
- Definition:

$$g(x, y) = f(x, y) + \lambda \cdot \Delta f(x, y)$$

- Parameter $\lambda > 0$
  - greater $\lambda \Rightarrow$ stronger emphasis of high-frequency features
  - unsharp masking is a **high-boost** filter

Meaning of unsharp masking



$f(x)$ $\quad f * w_G$ $\quad f * w_L = f - f * w_G$ $\quad f + \lambda \cdot f * w_L$

- *Low-pass* (mean): Image smoothing
- *High-pass* (Laplace): Difference between image and output of low-pass
- *High-boost* (unsharp masking): Part of high-pass added to image
- ‘Low-’ and ‘high-’ refer to filter action in *frequency domain*

Simple convolution kernel for unsharp masking

Using 8-neighbour version of Laplace filter, for $f + \lambda \Delta f$ we have

$$\begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} + \lambda \begin{array}{|c|c|c|}\hline -1 & -1 & -1 \\ \hline -1 & 8 & -1 \\ \hline -1 & -1 & -1 \\ \hline \end{array} = \lambda \begin{array}{|c|c|c|}\hline -1 & -1 & -1 \\ \hline -1 & 8 + 1/\lambda & -1 \\ \hline -1 & -1 & -1 \\ \hline \end{array}$$

Introducing parameter $\beta = 1/\lambda$ and normalising, we obtain kernel

$$w_U = \frac{1}{9} \begin{array}{|c|c|c|}\hline -1 & -1 & -1 \\ \hline -1 & 8 + \beta & -1 \\ \hline -1 & -1 & -1 \\ \hline \end{array}$$

- $0 \leq \beta \leq 1$; typical values are 0.1–0.2
- Normalisation: largest possible output value is G_{max} (255)
  - other normalisations can also be used

Examples and summary of unsharp masking

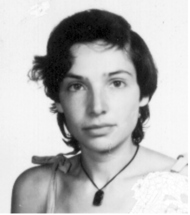
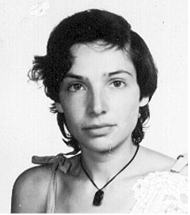


image 1


result 1

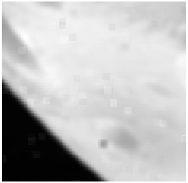
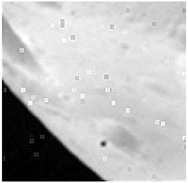


image 2


result 2

- Used to enhance contrast, especially in photography
- Enhances high-frequency features, such as edges
- Can amplify noise

Outline

1. Image filtering
   - Correlation and convolution
   - Basics of noise filtering
2. Frequently used filters
   - Linear smoothing filters
   - Median filter
   - Laplace filter
   - Unsharp masking
3. Fast and adaptive filters
   - Separable filters
   - Run filtering
   - Adaptive noise filtering

Filter separability 1/2

- Filter can be decomposed into product of 1D filters

$$w(x,y) = u(x) \cdot v(y)$$

- Example of separable filter
    - Each entry of 2D filter matrix is product of corresponding entries of 1D filters

$$\begin{array}{c} w(x,y) \\ \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array} \end{array} \qquad \begin{array}{ccc} & & u(x) \\ & & \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline \end{array} \\ v(y) & \begin{array}{|c|}\hline 1 \\ \hline 2 \\ \hline 1 \\ \hline \end{array} & \end{array}$$

Filter separability 2/2

- Number of operations N_{ops} in each point for $D_W \times D_W$ window
  - original filter: $N_{ops} = O(D_W^2)$
  - separable filter: $N_{ops} = 2 \cdot O(D_W)$
- Gaussian filter and box filter are separable
  - Gaussian $w_G(x, y) = w_G(x) \cdot w_G(y)$, $w_G(x) \propto \exp\left\{-\frac{x^2}{2\sigma^2}\right\}$
  - box filter is product of two unit 1D filters
  - running implementation of box filter is even faster
- Decomposing 2D filter into linear combination of 1D filters
  - use Singular Value Decomposition (SVD)
  - not necessarily faster: depends on number of 1D filters

Outline

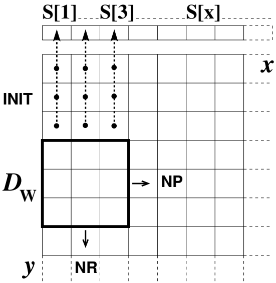
1. Image filtering
   - Correlation and convolution
   - Basics of noise filtering
2. Frequently used filters
   - Linear smoothing filters
   - Median filter
   - Laplace filter
   - Unsharp masking
3. Fast and adaptive filters
   - Separable filters
   - Run filtering
   - Adaptive noise filtering

Notion of run filtering

- When window moves to next position,
  - do not compute output value *from scratch*
  - instead, *update* output value obtained in previous position
- Run filtering solutions exist for different filters
  - box filter
  - median filter
- Efficiency depends on simplicity of updating
  - additive quantities like average are easy to update
  - nonlinear median is more difficult to update
- Run filtering can be extended to windows of more complex shape

Run filtering for box filter

- Data structure: array $S[x]$
- Initialisation (INIT)
    - for starting row, compute column sums $S[x]$
- First position in row
    - compute window sum from $S[x]$
- Shift in row (NP)
    - update window sum: subtract leaving S, adding entering S
- Next row (NR):
    - update each $S[x]$: subtract leaving pixel, add entering pixel



N_{ops} independent of D_W if image is much larger than window

Outline

1. Image filtering
   - Correlation and convolution
   - Basics of noise filtering
2. Frequently used filters
   - Linear smoothing filters
   - Median filter
   - Laplace filter
   - Unsharp masking
3. Fast and adaptive filters
   - Separable filters
   - Run filtering
   - Adaptive noise filtering

Adaptive neighbourhood selection

- Filters considered up to now are non-adaptive (position-independent):
    - fixed neighbourhood selection procedure
    - fixed function that calculates output value
- Adaptivity means **using local context** to improve performance of noise filters
    - avoid ‘averaging across edges’ by mean filter
    - avoid rounding of corners by median filter
- Main cause of these undesirable effects
    - pixels belonging to different classes (distributions) are mixed by filter
    - when window is on contour, object and background pixels are mixed

Basic idea

- Try to separate
  - object pixels from background pixels
  - relevant greyvalues from noise
- Adaptivity in neighborhood pixel selection: select *relevant* pixels
  - until now, we used all pixels of window
  - now, we will select certain pixels
- Adaptivity in function computing output value: none
  - until now, we used fixed functions: mean, median, etc.
  - this will *not* change

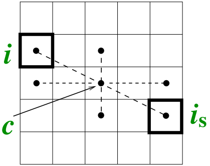
Pixel selection in $n \times n$ window

- **Standard** neighbourhood
  - use all n^2 pixels
- ***k*-nearest** neighbours (k-NN)
  - select k pixels closest in grey value to central pixel c
  - possible choice $k = n \times \left[\frac{n}{2}\right] + (n - 1)$
  - for example: when $n = 3$, $k = 5$
- **Sigma-nearest** neighbours
  - select pixel i if $|I(i) - I(c)| < k \cdot \sigma_{noise}$
  - usually, $k = 2$
  - σ_{noise} is standard deviation of noise

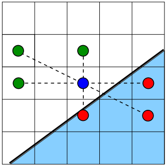
  $\Rightarrow$ estimated in flat (non-variyng) region of image

Symmetric nearest neighbors

- Select pixel i if $|I(i) - I(c)| < |I(i_s) - I(c)|$
  - c is central pixel, $\{i, i_s\}$ pair of central-symmetric pixels
- Local context: intensity and geometry taken into account
- Useful in case of edges
  - selects pixels on same side of edge
  - avoids averaging across edge

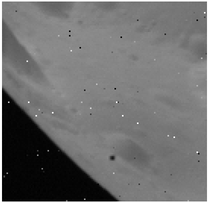


symmetric pixel pair

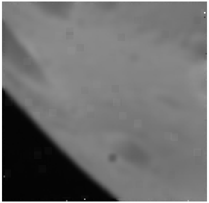


operation on edge

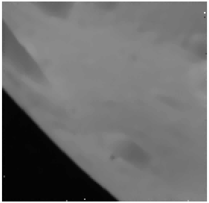
Comparison of standard and adaptive 5 × 5 filters



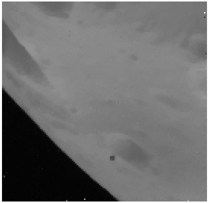
image



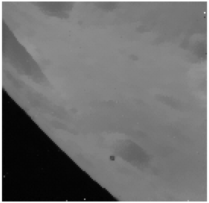
box



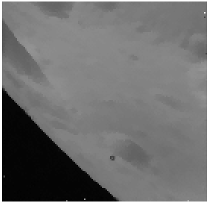
median



k-NN mean

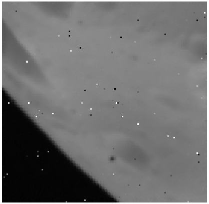


symm. mean

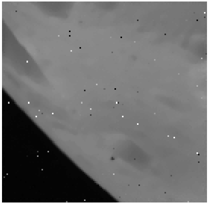


symm. med.

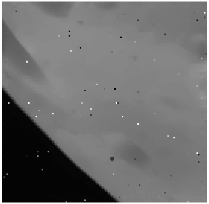
Sigma filter does not remove peak noise



sigma mean 5 × 5



sigma med. 5 × 5



sigma med. 9 × 9

- For peak-noisy pixel $I_{noisy}(x, y)$, interval $I_{noisy} \pm 2\sigma_{noise}$ does not include noise-free neighbours
  - $|I_{noisy} - I_{noisefree}| > 2\sigma_{noise}$
- Peak value I_{noisy} is selected

  $\Rightarrow$ noise is not removed