Image Filtering

Ling Shao

2

Linear filtering of discrete images

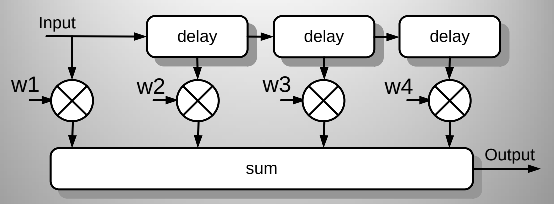
3

What is a linear filter?

- A linear filter is a system with an input and an output that:
  - Contains delays, multipliers and adders
    - Connected such that the output results as a weighted sum of delayed copies of the input signal and the output signal

4

Example linear filter



$$y[k] = \sum_n w_n x[k-n]$$

5

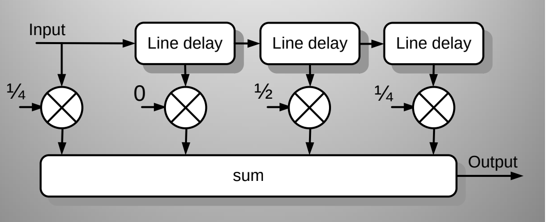
Simple notation

- We shall assume that a (1, 2, 1) filter outputs a weighted sum of horizontally neighbouring pixels in an image
- We shall assume that a $\begin{pmatrix}1\\2\\1\end{pmatrix}$ filter outputs a weighted sum of vertically neighbouring pixels in an image
  - Vertically shifting the image requires a line-delay
    - Orders of magnitude more expensive than a pixel-delay
  - Analogously, temporal filtering requires picture-delays
    - Again orders of magnitude more expensive

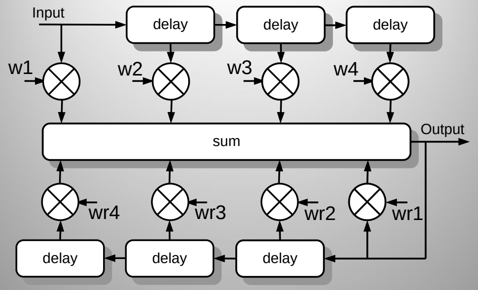
6

From horizontal to vertical filtering

- If we increase the delay from 1 to # pixels/line, the pixels combined in the filter output are vertical neighbours
- A delay is now implemented with a "line-memory"



7 **What if we re-use output samples? Recursive filtering**

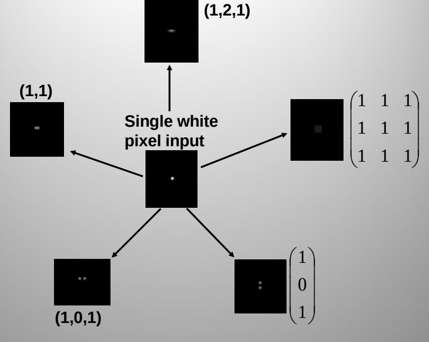


$$y[k] = \sum_n w_n . x[k-n] + \sum_m w_{mr} . y[k-m]$$

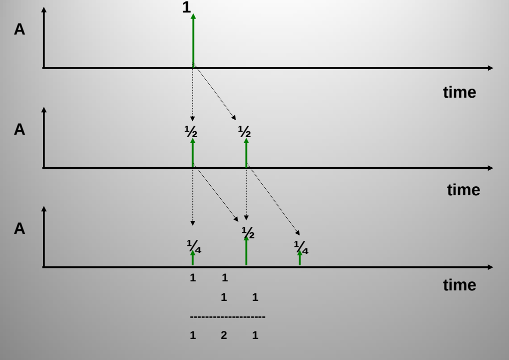
8

Calculating with impulse responses

9 **Some impulse responses**



10 **The cascading of two (1, 1) filters**



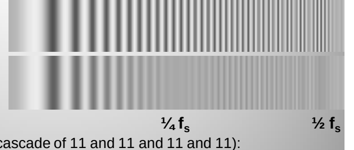
11 **Simple calculation of impulse response of cascade**

- Cascade of (1,1) and (1, 1):
  - 1 1
  - 1 1
  - --------------------
  - 1 2 1
- Cascade of 121 and 11:
  - 1 2 1
  - 1 2 1
  - ---------------------------
  - 1 3 3 1

12 **Simple calculation of impulse response of cascade**

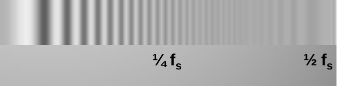
- Cascade of 11 and 101

```
1   1
    0   0
        1   1
----------------------------
1   1   1   1
```



- Cascade of 121 and 121 (cascade of 11 and 11 and 11 and 11):

```
1   2   1
    2   4   2
        1   2   1
--------------------------------
1   4   6   4   1
```



13 **Simple calculation of impulse response of cascade**

- Cascade of (horizontal) 11 and a(vertical) $\begin{pmatrix}1\\1\end{pmatrix}$

1	1
1	1

- Cascade of (horizontal) 121 and a (vertical) $\begin{pmatrix}1\\2\\1\end{pmatrix}$

1	2	1
2	4	2
1	2	1

14

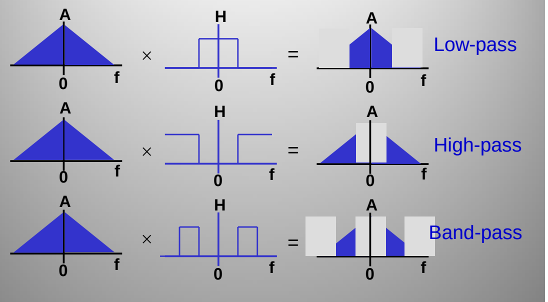
Qualitative analysis of linear filters

15 **Qualitative filter behaviour in the frequency domain**

Transfer of filter

Spectrum input image

Spectrum of output image



16 **Filtering images**

- Move the coefficient matrix (impulse response) over each pixel in the image, multiply the entries by the pixels, and sum together
  - E.g. 3x3 “box”-filter
  - Effect is averaging

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\1 & 1 & 1\\1 & 1 & 1\end{bmatrix}$$

- Formal name for this procedure is convolution of the image data with the filter “mask” or “kernel”

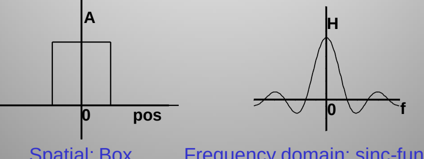
17 **Effect of the box filter on a natural image**

Original (input) | Box-filter output

18 **Box Filter**

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\1 & 1 & 1\\1 & 1 & 1\end{bmatrix}$$

- Box filters smooth by averaging neighbouring pixels
- In the frequency domain, the box-filter leaves low frequencies unaltered and attenuates (reduces) high frequencies
- So, the box filter is clearly a low-pass filter

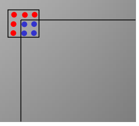


Spatial: Box | Frequency domain: sinc-function

19 What about the image boundaries?

$$F_{output}[x][y] = \sum_{i=-k/2}^{k/2} \sum_{j=-k/2}^{k/2} F_{input}[x+i][y+j]w[i+k/2][j+k/2]$$

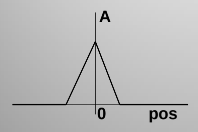
- At (0,0) for instance, you need pixel data for (-1,-1), which doesn't exist
  - Option 1: Make the output image smaller – don't evaluate pixels you don't have all the input for
  - Option 2: Replicate the edge pixels
    - Equivalent to: pos = x + i; if (pos < 0) pos = 0; same for y
  - Option 3: Reflect image about edge
    - Equivalent to: pos = x + i; if (pos < 0) pos = -pos; same for y



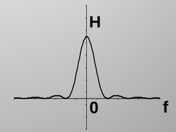
20 Bartlett Filter

$$\frac{1}{81}\begin{bmatrix} 1 & 2 & 3 & 2 & 1 \\ 2 & 4 & 6 & 4 & 2 \\ 3 & 6 & 9 & 6 & 3 \\ 2 & 4 & 6 & 4 & 2 \\ 1 & 2 & 3 & 2 & 1 \end{bmatrix}$$

- Triangle shaped filter in spatial domain
- Cascade of two box filters
- So in frequency domain; attenuates high frequencies more than a box-filter



Spatial: Triangle (Box⊗Box)



Frequency: sinc^2

21 Effect of the Bartlett filter on a natural image

Original (input)

Bartlett-filter output

22 High-pass filters

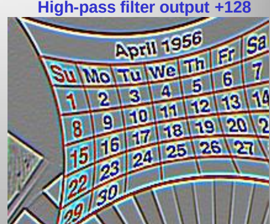
- A high-pass filter can be obtained from a low-pass filter
  - If we subtract the smoothed image from the original, we must be subtracting out the low frequencies
  - What remains must contain only the high frequencies
- High-pass masks come from matrix subtraction:
  - eg: 3x3 Box:

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \frac{1}{9}\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

23 Effect of the high-pass filter on a natural image

Original (input)

High-pass filter output +128



24 Edge Enhancement

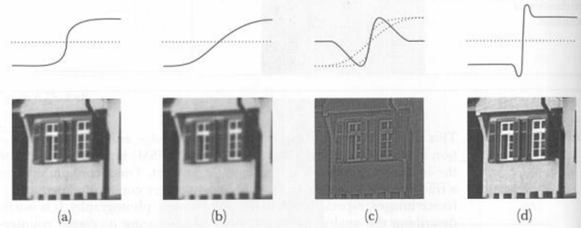
- High-pass filters give high values at edges, low values in constant regions
- Adding high frequencies back into the image enhances edges
- Possible approach:
  - $\text{Image} = \text{Image} + [\underbrace{\text{Image} - \underbrace{\text{smooth(Image)}]}_{\text{Low-pass}}}_{\text{High-pass}}$

25 Effect of the edge-enhance, or peaking, filter

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 12 & -1 \\ -1 & -1 & -1 \end{bmatrix} \times 1/4$$

26 Edge enhancement, peaking, or unsharp masking

Figure 6.32. (a) Original image. (b) Blurred image. (c) Difference between first two. (d) Enhanced image.



(a) (b) (c) (d)

27 Negative filter coefficients may lead to problems!

- The negative values in high-pass filters can lead to negative image values
  - Most image formats don't support this, and if they do, we cannot have image negative light from the display….
- Also, the boosting effect on high frequencies may lead to values higher than peak-white
- Solutions:
  - Clipping: Chop off values below min or above max
  - Offset: Add a constant to move the min value to 0
  - Re-scale: Rescale the image values to fill the range (0,max)

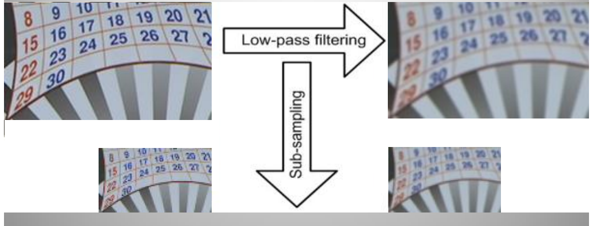
28

The purpose of filters

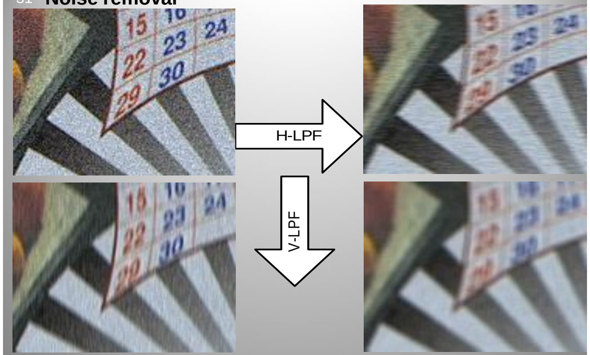
29 Purpose of filters

- Removal of spectral components
  - E.g. for alias prevention, or removal of interference signal
- Enhancement of spectral components
  - Edge/feature enhancement
- Removal of noise
  - Balance between noise suppression and suppression of relevant image components
- Interpolation
  - Image up- and down-scaling, geometrical deformations

30 Alias prevention



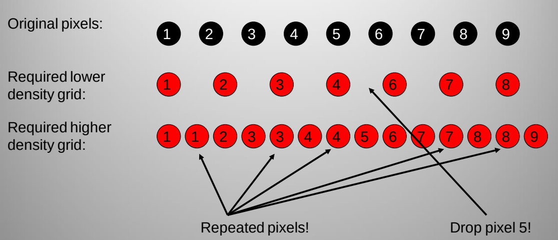
31 **Noise removal**



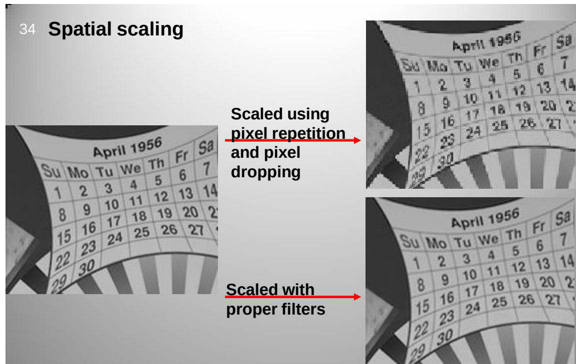
32

Filtering and image re-sizing (1D case)

33 **The simple way. Pixel repetition and dropping.**

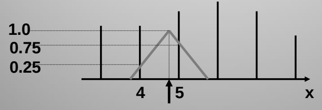


34 **Spatial scaling**



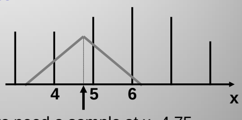
35 **2 tap linear interpolation**

- The filter size is 2, and we need a sample at x=4.75
  - We need the filter output, F(x), from x=4, x=5
  - We need the filter value, H(s), at s=-0.75, and s=0.25
  - We compute: F(4)*H(-0.75)+F(5)*H(0.25)+F(6)*H(1.25)
  - Result is: 0.25F(4) + 0.75 F(5)



36 **Interpolation with the Bartlett filter**

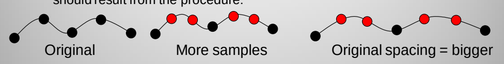
- Place a Bartlett filter at the required position
- Multiply the value of the neighbouring pixel by the corresponding filter value, and accumulate the result
  - Convolution with discrete samples
- The filter size is a parameter



- Assume the filter size is 3, and we need a sample at x=4.75
  - We need the image samples, F(x), from x=4, x=5 and x=6
  - We need the filter value, H(s), at s=-0.75, s=0.25 and s=1.25
  - We compute: F(4)*H(-0.75)+F(5)*H(0.25)+F(6)*H(1.25)

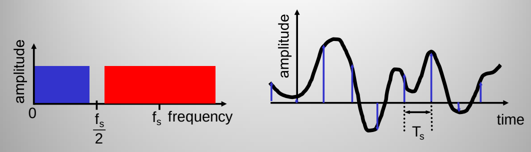
37 Re-sampling

- Making an image larger is like sampling the original signal at a higher density
  - You need more pixels to represent the same thing, so a higher pixel density should result from the procedure:



- Reducing an image in size is like sampling at lower density
- Generating new samples of the "same" function is called *re-sampling*
- In theory, 2 steps:
  - Reconstruction of the continuous signal
  - Followed by sampling with the new sampling frequency
- In practice: sample rate conversion in the discrete domain

38 The sampling theorem

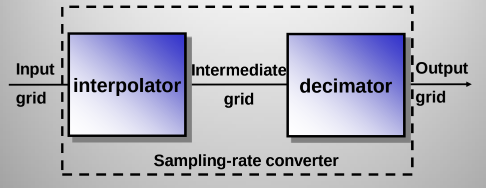


We have a continuous signal

We sample it to obtain a discrete representation

Sampling theorem: we can reconstruct the continuous signal from its discrete representation, provided it contained no frequencies above half the sampling frequency

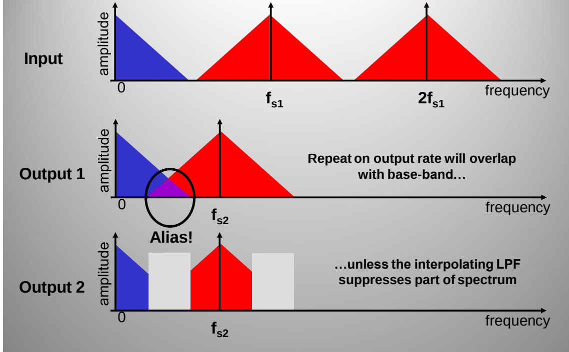
39 The general principle of sample rate conversion



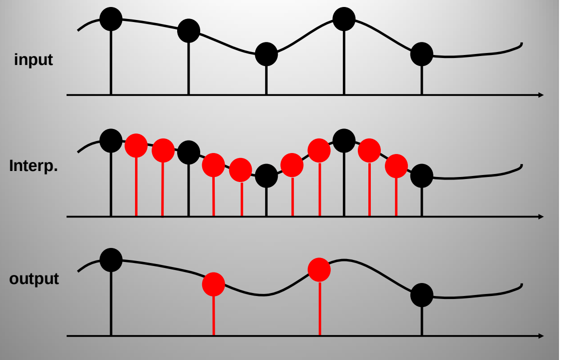
As a consequence of the time-discrete nature of the processing, the input and output sampling frequency have a rational relation.

Output results from an integer up-sampling and an integer sub-sampling

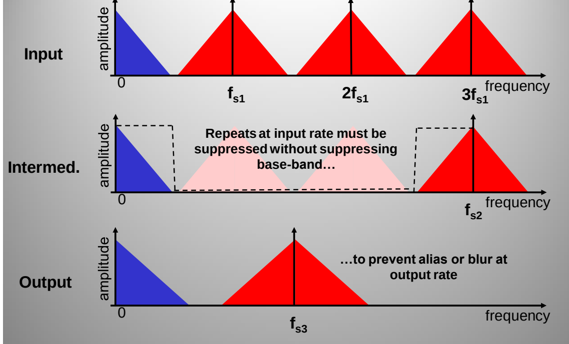
40 Spectra in sampling rate conversion (down-sampling)

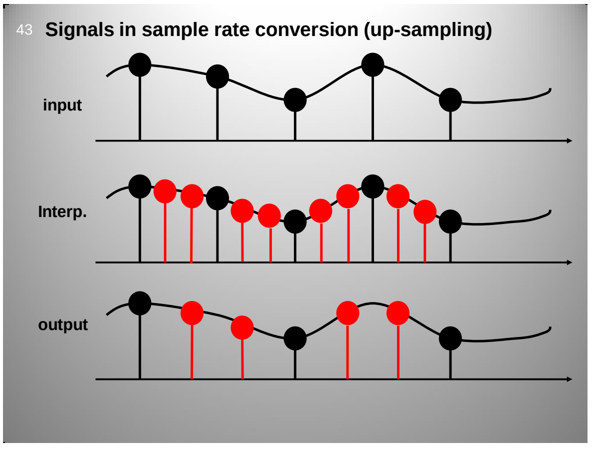


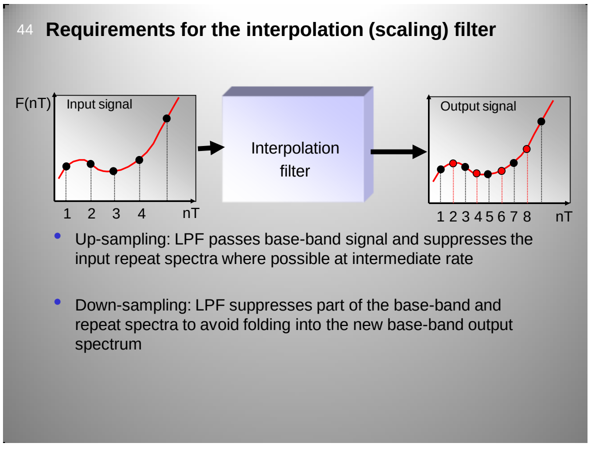
41 Signals in sample rate conversion (down-sampling)

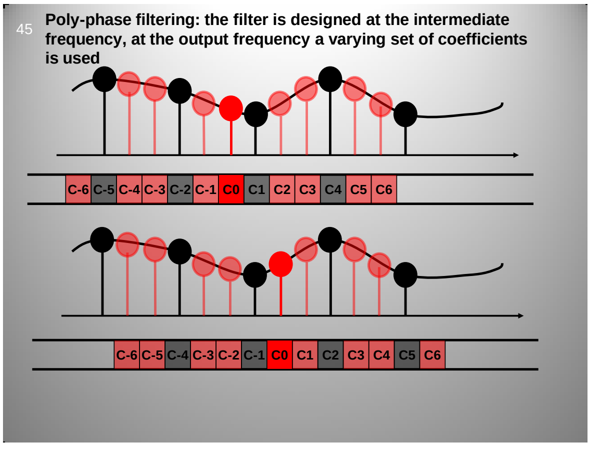


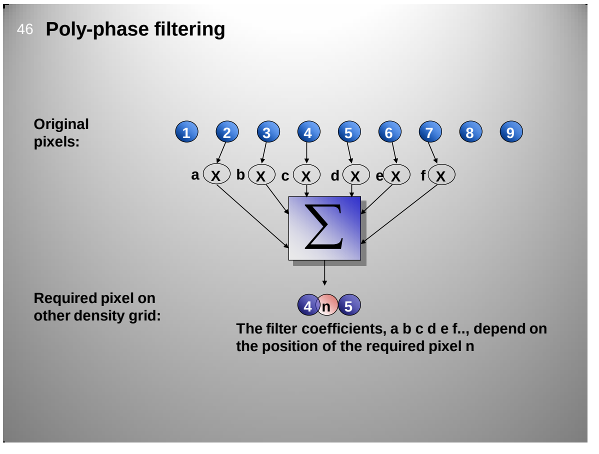
42 Spectra in sample rate conversion (up-sampling)



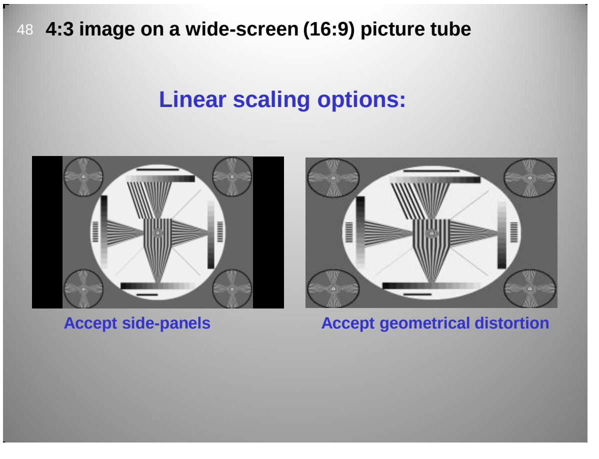

43 Signals in sample rate conversion (up-sampling)
input
Interp.
output


44 Requirements for the interpolation (scaling) filter
F(nT)
Input signal
1 2 3 4 nT
Interpolation filter
Output signal
1 2 3 4 5 6 7 8 nT
• Up-sampling: LPF passes base-band signal and suppresses the input repeat spectra where possible at intermediate rate
• Down-sampling: LPF suppresses part of the base-band and repeat spectra to avoid folding into the new base-band output spectrum


45 Poly-phase filtering: the filter is designed at the intermediate frequency, at the output frequency a varying set of coefficients is used
C-6 C-5 C-4 C-3 C-2 C-1 C0 C1 C2 C3 C4 C5 C6
C-6 C-5 C-4 C-3 C-2 C-1 C0 C1 C2 C3 C4 C5 C6

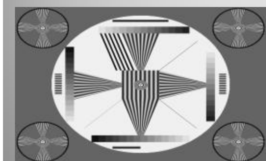

46 Poly-phase filtering
Original pixels:
1 2 3 4 5 6 7 8 9
a X b X c X d X e X f X
Σ
Required pixel on other density grid:
4 n 5
The filter coefficients, a b c d e f.., depend on the position of the required pixel n

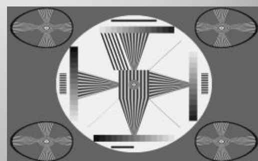

47
Application: Aspect ratio conversion


48 4:3 image on a wide-screen (16:9) picture tube
Linear scaling options:
Accept side-panels
Accept geometrical distortion

49 **4:3 image on a wide-screen (16:9) picture tube**

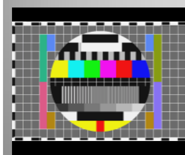
Non-linear scaling options:


Accept geometrical distortion


Panorama view

50 **Wide-screen image on a 4:3 picture tube**

Linear scaling options:


Accept letter-box


Geometrical distortion


Partial image loss

51 **Wide-screen image on a 4:3 picture tube**

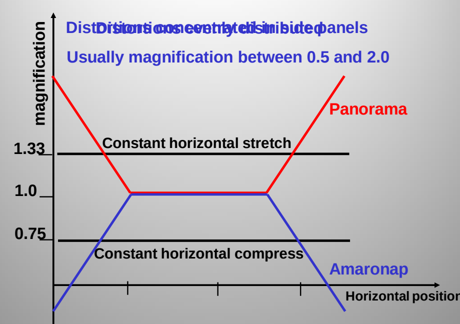
Non-linear scaling options:


Amaronap view

52 **Options in wide-screen conversion**

Distortions concentrated in side panels / Distortions evenly distributed

Usually magnification between 0.5 and 2.0



53

Non-linear filters

54 **Different filter types**

- Linear filters
- Rank-order filters
- Hybrid filters
- Morphological filtering
- Adaptive filters

55 Linear filters not very effective against shot noise

- Linear filters fine in case small noise values are more frequent than high noise values
  - E.g. Gaussian noise
- Shot noise (salt and pepper noise,..) is characterized by relatively few extreme noise values and (almost) no small noise values
  - E.g. impulses from ignition of combustion engines, or film defects
- To know if a pixel is extreme we have to rank adjacent pixel values

56 Linear and rank-order filters – The difference

This is what a linear filter outputs:

$$y = w_1 x_1 + w_2 x_2 + ... + w_n x_n$$

Let the filter support (vector) define the input pixels used:

$$\mathbf{S}(x) = (x_1, x_2, ..., x_n)^{\mathrm{T}}$$

and the coefficient vector defines the weighting:

$$\mathbf{W} = (w_1, w_2, ..., w_n)$$

Now we can briefly define the filtered output pixel as:

$$y(x) = \mathbf{W}.\mathbf{S}(x)$$

57 Linear and rank-order filters – The difference

So, the linear filter is defined as:

$$y(x) = \mathbf{W}.\mathbf{S}(x)$$

If we now define the ordered (ranked) support:

$$\mathbf{S}_r(x) = (x_{(1)}, x_{(2)}, ..., x_{(n)})^{\mathrm{T}}$$

with:

$$x_{(1)} \leq x_{(2)} \leq .... \leq x_{(n)}$$

Then the rank-order filter is defined by:

$$y(x) = \mathbf{W}_r.\mathbf{S}_r(x)$$

58 Linear and rank-order filters – The difference

- The pixel-weights in a linear filter are determined by the spatio-temporal position of the pixel relative to the output position
- The pixel-weights in a rank-order filter are determined by the rank number of the pixel after ordering all values in the support. Examples:
  - minimum,
  - maximum,
  - midpoint = (max + min)/2
  - median,
  - α-trimmed mean

59 Effectiveness against shot noise

Median filter, especially effective for shot noise:

$$\mathbf{W}_{med} = (0,.......1,.......,0)$$

60 Less extreme distributions: The α-trimmed-mean filter

The general rank-order filter is defined by:

$$y(x) = \mathbf{W}_r.\mathbf{S}_r(x)$$

The median filter, especially effective for shot noise:

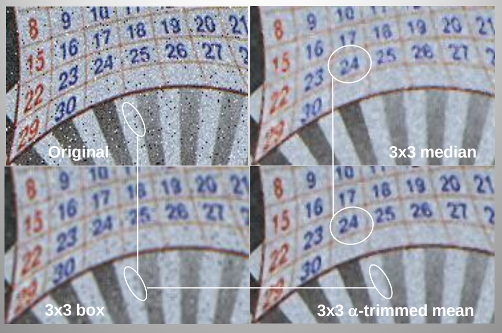
$$\mathbf{W}_{med} = (0,.......1,.......,0)$$

The α-trimmed-mean filter, long-tail distributed noise, but less extreme:

$$\mathbf{W}_{\alpha} = (0,.......0,1,1,1,1,1,0.......,0)$$

α-central pixels are averaged, extremes ignored

61 **Shot noise + Gaussian noise reduction**



62 **Combination of linear and rank-order: The hybrid filter**

If we concatenate the linear and ranked supports:

$$\mathbf{S}_h(x) = (x_1, x_2, ..., x_n, x_{(1)}, x_{(2)}, ..., x_{(n)})^{\mathrm{T}}$$

and also the coefficient vectors defines the weighting:

$$\mathbf{W}_h = (w_1, w_2, ..., w_n, w_{r1}, w_{r2}, ..., w_{rn})$$

Then the hybrid filter is defined by:

$$y(x) = \mathbf{W}_h . \mathbf{S}_h(x)$$

•LMS-optimization is possible to find coefficients

63 **Combination of linear and rank-order: The bilateral filter**

In the linear filter, weights depend on position relative to centre:

$$w_k = f_1(c-k)$$

In the rank-order filter, they depend on the "similarity" with current pixel:

$$w_k = f_2(|x_k - x_c|)$$

In the bilateral filter the weight is defined by:

$$w_k = N.f_1(c-k).f_2(|x_k - x_c|)$$

Where N is selected such that the sum of the coefficients is 1

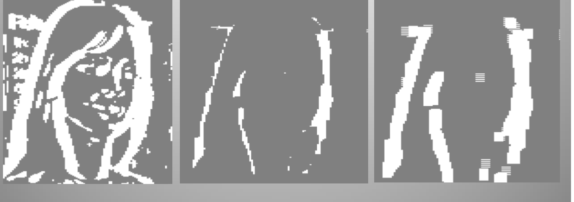
Functions f_1 and f_2 may be e.g. Gaussian or triangular functions

64 **The max and min filter (morphological filtering)**

Original motion detection signal | 5x5 max filter (dilation) | 5x5 max- cascaded with 5x5 min-filter (closing)

65 **The max and min filter (morphological filtering)**

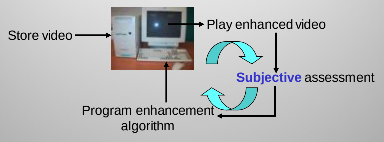
Original motion detection signal | 5x5 min filter (erosion) | 5x5 min- cascaded with 5x5 max-filter (opening)



66

Filter optimization

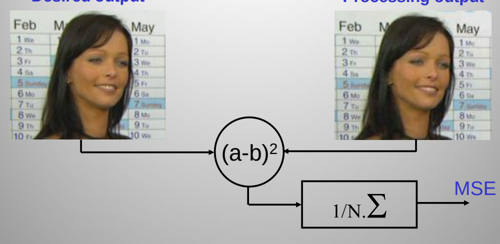
67 Optimization is our problem



Subjective assessment is time consuming

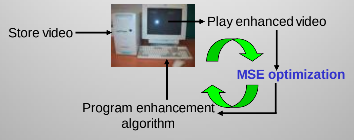
Huge design space requires automatic optimization…

68 It seems straightforward…



Minimize a sum of squared pixel differences by varying all parameters…

69 Bottleneck eliminated?



70 MSE-optimal (non-recursive) linear filtering

•Cross-correlation matrix

•Auto-correlation matrix

- The linear filter is defined by:

$$y = w_1x_1 + w_2x_2 + w_3x_3 + \ldots + w_nx_n$$

- Compare the output with original image to know error:

$$e = y_o - y$$

- For a minimal MSE, the first derivative should be zero:

$$\frac{\partial e^2}{\partial w_i} = 2\left(\frac{\partial e}{\partial w_i}\right)e = 2x_i e = 0$$

- If we now define:

$$X_{ji} = x_i x_j \qquad Y_i = x_i y_o$$

We then get:

$$\begin{bmatrix} X_{11} & X_{12} & \ldots & X_{1n} \\ X_{21} & X_{22} & \ldots & X_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ X_{n1} & X_{n2} & \ldots & X_{nn} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \ldots \\ w_n \end{bmatrix} = \begin{bmatrix} Y_1 \\ Y_2 \\ \ldots \\ Y_n \end{bmatrix}$$

71 MSE-optimal (non-recursive) linear filtering

- We have to solve the following equation:

$$\begin{bmatrix} X_{11} & X_{12} & \ldots & X_{1n} \\ X_{21} & X_{22} & \ldots & X_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ X_{n1} & X_{n2} & \ldots & X_{nn} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \ldots \\ w_n \end{bmatrix} = \begin{bmatrix} Y_1 \\ Y_2 \\ \ldots \\ Y_n \end{bmatrix}$$

- We thereto write:

$$\mathbf{X.W} = \mathbf{Y}$$

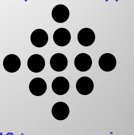
- And conclude upon the following optimal weights:

$$\mathbf{W} = \mathbf{X}^{-1}\mathbf{X.W} = \mathbf{X}^{-1}\mathbf{Y}$$

72 We can use this for noise reduction

Original noisy image

13-taps filter support



13-taps LMSE-filter

13-taps averaging filter

73 **We may also use it for de-blurring noisy images**

Original blurred noisy image

3x3LMSE-filter

3x3 peaking-filter

- Conclusion: LMSE-filter enhances noise less
- Therefore, also reduces the blur less…
- MSE-best compromise

74 **How does this compare with "inverse filtering"?**

- We shall see later that it is also possible to calculate the inverse filter for perfect de-blurring, which turns out to be a recursive filter
- The inverse filter has an infinitely high gain if the blurring filter has a zero response for a given frequency
  - Consequence is that noise will be amplified unlimited…

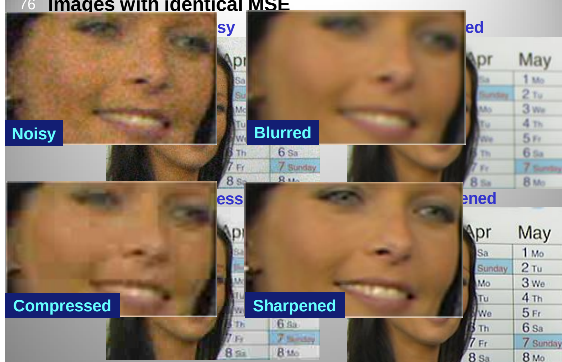
Original blurred noisy image

De-blurring with inverse filter

75

Problem with LMSE-filters

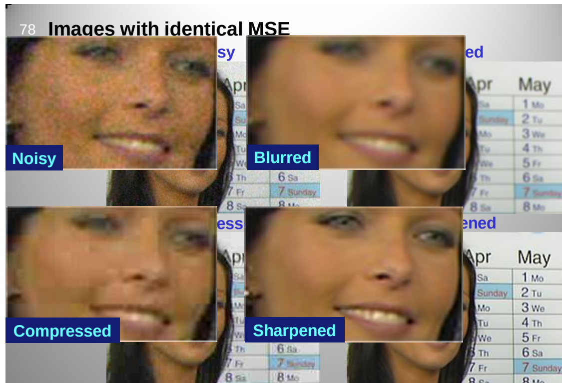
76 **Images with identical MSE**



77

Trained Filters (optimizing adaptive filters)

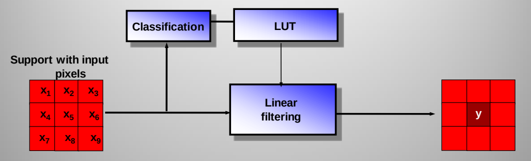
78 **Images with identical MSE**



79 **What's wrong with MSE?**

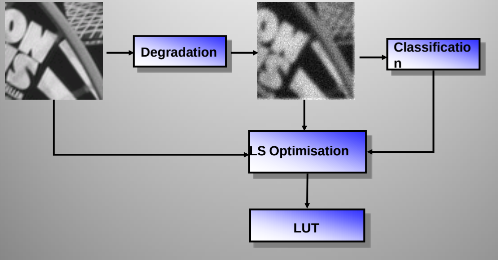
- The mistake is in the averaging!
  - Some picture parts get a lot better with a method that is poor on the average!
- → characterize image parts that can be treated identically…
- …and individually optimize (MSE) parts with the same character!

80 **Classification-based trained filters – filtering process**



$$y = w_{1c}x_1 + w_{2c}x_{2c} + ... + w_{nc}x_n$$

81 **Classification-based trained filters – training process**



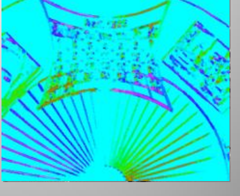
82

Classification examples

83 **Local edge direction classification**

Image

Edge directions (colour)

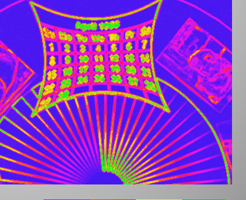


10º 50º 90º 130º 170º

84 **Local contrast classification**

Image

Local contrast (colour)



Low → High

85 **Local sharpness classification**

Image / Local sharpness (colour)

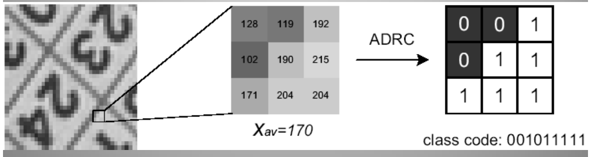
86 **Coding the local structure to classify support data**

Classification code is concatenation of pixels reduced to single bit:

$$\mathrm{ADRC}(x)=\begin{cases}1 & ,(x \ge x_{av})\\ 0 & ,(x < x_{av})\end{cases}$$

with

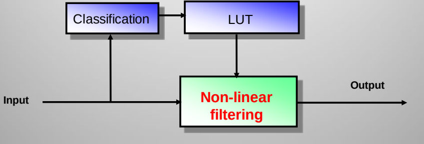
$$x_{av}=\frac{1}{n}\sum_{i=1}^{i=n} x_i$$



87

Not only linear filtering

88 **Architecture of the hybrid filter**

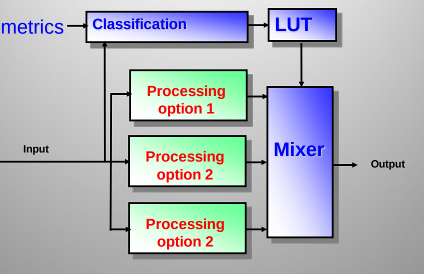


Concept can be extended to trained rank-order and bilateral filters

89

Not only filtering

90 **Trained Mixing of processing options**

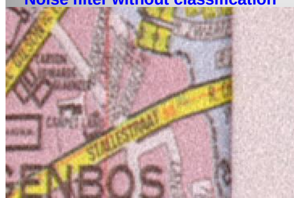


91

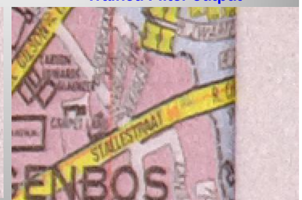
Application examples

92 **Sharpening and de-noising**

Noise filter without classification



Trained Filter output



93 **PixelPlus**

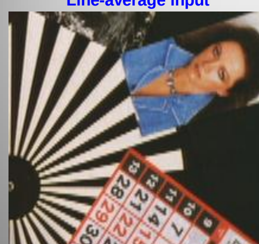
Up-scaling combined with sharpening

Standard up-scaling

Trained Filter output

94 **Interlace artifact removal**

Line-average input



Output Trained Filter

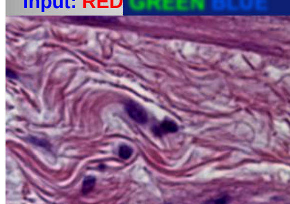
95 **Enhancement of digital video**

Input

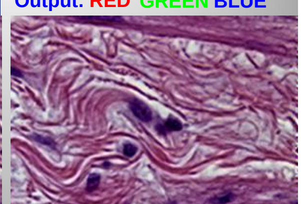
Trained Filter output

96 **Microscopy resolution enhancement**

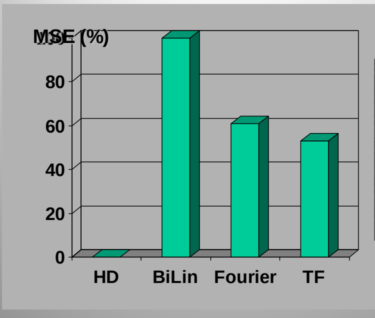
Input: RED GREEN BLUE

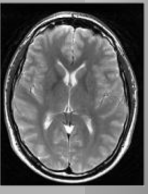


Output: RED GREEN BLUE

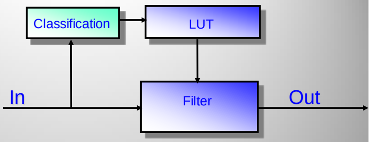


97 **Up-scaling of MRI-images**





98 **Conclusions**



- Similar architecture for many functions
- Design methodology replaces heuristics

99 **The value of trained filtering**

- Automatic optimization
  - Design methodology replaces heuristics for tuning
  - **No thinking → faster**
- Researcher can focus on creatively finding the relevant classes

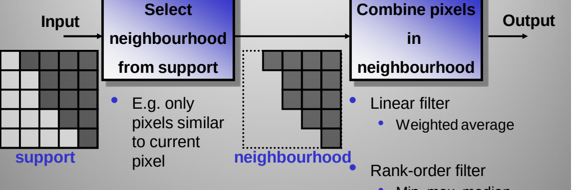
100

Neighbourhood Selection

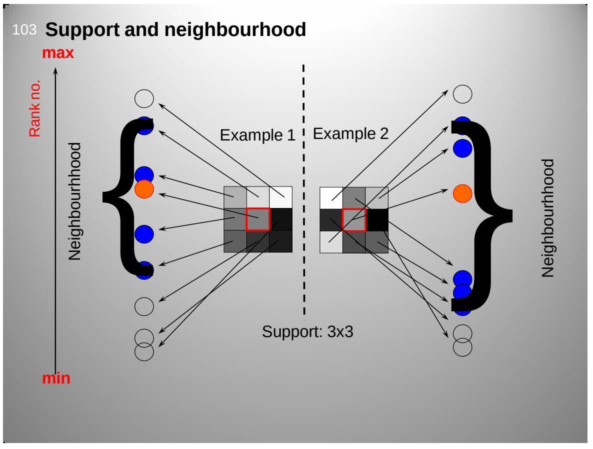
101 **Neighbourhood selection**

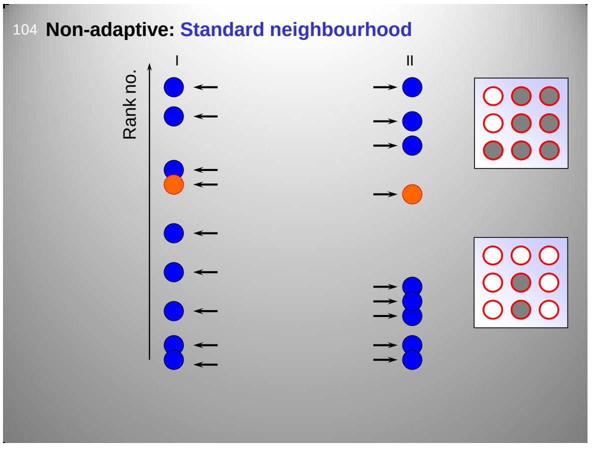
- So far, we assumed that ALL pixels in the support are combined under ALL circumstances
- We may also propose to adapt the set of pixels that are "combined" to expectations about local correlation, using a so-called neighbourhood selection technique
- A neighbourhood shall be defined as a sub-set of the support
  - Various options exist to exclude pixels from the support
    - K-nearest, sigma nearest, symmetrical nearest,…
- Pixels in the neighbourhood are then combined
  - Weighted averaging, rank-order, etc.

102 **Adaptive filtering: Neighbourhood Selection**

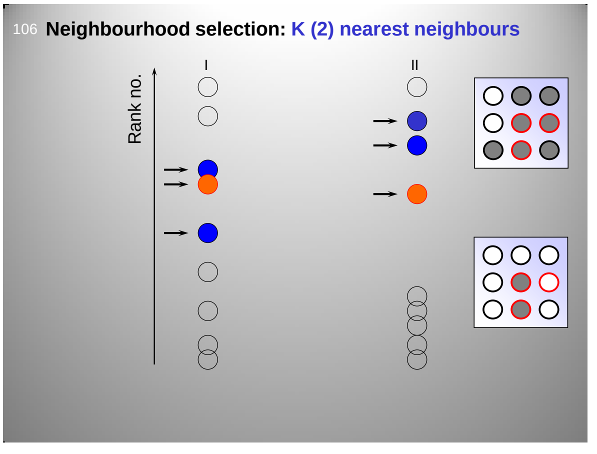


- E.g. only pixels similar to current pixel
- Linear filter
  - Weighted average
- Rank-order filter
  - Min, max, median

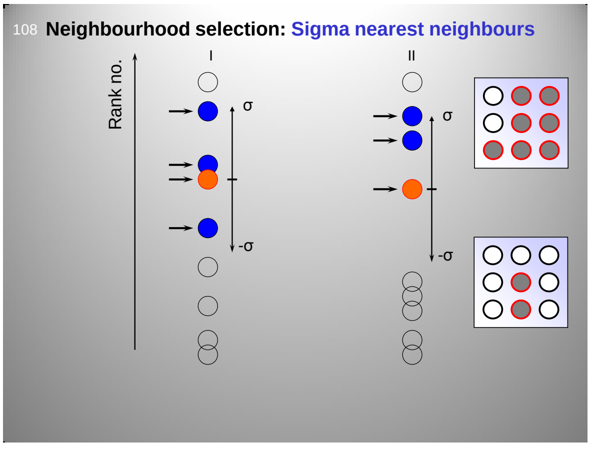

103 Support and neighbourhood
max
Rank no.
Neighbourhood
Example 1
Example 2
Neighbourhood
Support: 3x3
min


104 Non-adaptive: Standard neighbourhood
Rank no.
I
II


105 Effect of the standard neighbourhood selection (5x5)

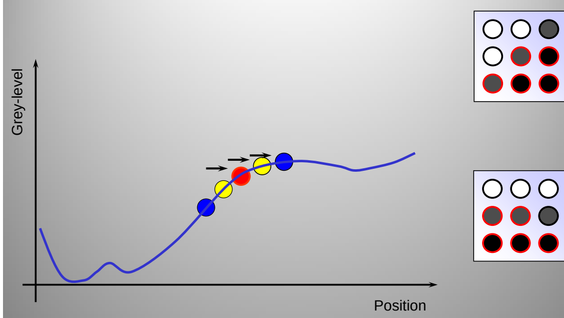

106 Neighbourhood selection: K (2) nearest neighbours
Rank no.
I
II


107 Effect of the k (12) nearest neighbourhood selection (5x5)


108 Neighbourhood selection: Sigma nearest neighbours
Rank no.
I
II
σ
-σ
σ
-σ

109 Effect of the sigma nearest selection (5x5)

110 Neighbourhood selection: Symmetric nearest neighbours



111 The effect of symmetric nearest neighbour filtering (5x5)

112 The effect of symmetric nearest neighbour filtering (5x5)

113 Adaptive filtering: Neighbourhood Selection