Non linear filters

Overview:

- Image enhancement
  - Mean filter
  - Median filter
  - Mode filter
  - Sigma filter (*)
- Grey-scale morphology
  - Minimum filter
  - Maximum filter
  - Nth filter
Non linear filters

Overview:

- Local extremes
  - LocalMax filter
  - LocalMin filter
- Edge preserving smoothing (*)
  - Kuwahara filter
  - SNN_Mean filter
  - SNN_Median filter
- “Texture” (*)
  - Range filter
  - Variance filter
- Motion detection (*)

Rank operators

The ranking operator initialises a destination image by sliding a mask across a source image.

The pixel values under the mask are used to calculate a new value. Each type of rank operator uses its own algorithm.

This new value is assigned to the destination image at the position of the centre (= origin) of the mask.

Rank operators are non-linear filters.
Image enhancement

- **Mean filter:**
  the new value is the mean value of the selected pixels under the mask.

- **Median filter:**
  the new value is the median value of the selected pixels under the mask.
  The median is the middle value in the sorted order of values.

- **Mode filter:**
  the new value is the mode value of the selected pixels under the mask.
  The mode is the value with the highest frequency of occurrence.

Image enhancement (*)

- **Sigma filter:**
  the new value is the mean value of the selected pixels under the mask if the absolute difference between the mean value and the original pixel is smaller than the specified deviation.
  Otherwise the new value is the value of the origin pixel.

- **Usage:** noise reduction
Demonstration image enhancement (*)

- Open image circles.jl
- Add noise 1 0 50
- Apply the three filters with 3x3 mask to image with noise
- (*) for Sigma use deviation = 10
- Median filter gives best result
- Mode filter gives worse result
- Note there are different kinds of noise, in this case only impulse noise is investigated.
- Note Median with plus 3x3 mask gives even a better result.
Grey-scale morphology

- Minimum filter: the new value is the minimum value of the selected pixels under the mask. Used for grey-scale erosion and grey-scale dilation.
- Maximum filter: the new value is the maximum value of the selected pixels under the mask. Used for grey-scale dilation and grey-scale erosion.
- Nth filter: the new value is the nth value of the ascending sorted selected pixels under the mask. This is a generalisation of Maximum and Minimum filter.

Grey-scale morphology

- Usage:
  - Grey-scale opening
  - Grey-scale closing
  - Noise reduction
Demonstration grey-scale morphology

- Open image circles.jl
- Apply 3 x Maximum filter with full 3x3 mask
  - Result: object border is replaced by background, little 'noise' dots are disappeared
- Subtract 1x Maximised image from original (= 2nd image)
  - Threshold 40 1000 on result gives edge of dark circle
- Apply 2x Minimum filter with full 3x3 mask on circle.jl
  - Result: objects grow
- Apply 3 x Maximum filter with 7x7 mask (no slide)
  - Result object shrink faster in one operation

Apply 3 x Maximum filter with full 3x3 mask
Subtract 1x Maximised image from original (= 2nd image) followed by Threshold 40 1000

Apply 2x Minimum filter with full 3x3 mask
Grey-scale morphology

- Finding background behind small objects
  
  Max and Min used in combination,
  
  $\text{Max}(\text{Min}(\text{image}))$ or $\text{Min}(\text{Max}(\text{image}))$

Max($\text{Min}(\text{image}))$ versus Min($\text{Max}(\text{image}))$
Max(Min(image)) versus Min(Max(image))

- For large edges same result

- Max(Min(image))
  - On bottom side of the small edges
  - Used for generation of dark backgrounds, "removes the bright spots"

- Min(Max(image))
  - On top side of the small edges
  - Used for generation of bright backgrounds, "removes the dark spots"

---

Exercise background generation 1

- Use image stdhand_r.jl.
- Try to find good threshold value in order to separate characters from the background.
  This will be unsuccessful due to uneven lightning conditions
- Remove background from image
- Now try to find good threshold values in order to separate characters from the background.
  - answer: stdhand_r.jls
Exercise background generation 2

- Use image shading_c.jl
- Try to find good threshold value in order to separate the cells from the background. This will be unsuccessful due to uneven lightning conditions
- Remove background from image
- Now try to find good threshold value in order to separate the cells from the background.
  - answer: shading_c_back.jls

Local extremes filters

- LocalMax filter
  - The local maximum filter operator initialises a destination image by sliding a mask across the source image. A new value is calculated for the destination image at the position of the centre (= origin) of the mask. This new value is the value of the origin if the origin value is the local maximum of the pixels under the mask otherwise the new value is set to the background value.
  - If the source pixel value at the origin equals to the background value (=0) the corresponding destination pixel is assigned the background value.

- LocalMin filter
Demonstration LocalMax filter (*)

Find middle of circle in circle.jl:
• Open image circle.jl
• Threshold 0 100
• EuclideanDistanceTransform EDTMask7x7 NoScale EDT 100 1
  note: there will be a special lecture about distance transforms
• LocalMaxFilter 0 EdgeExtend 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1
• Gamma 0.25  // for better displaying
Edge preserving smoothing Filters (*)

Image is smoothed but edges are preserved
- Kuwahara filter
- Symmetric Nearest Neighbour (SNN) filter

Usage:
- Sharpen up vague edges before using edge detection
- Artistic effects

Kuwahara filter (*)

KuwaharaFilter (srcImage, destImage, radius, edge)

The square window with the defined radius around the center pixel is divided in four overlapping regions.
Example for radius = 3:
- window is 5x5 pixels
- 4 regions of 3x3 pixels
- center pixel is in all regions

The output value for the central pixel in the window is the mean value of that region that has the smallest variance.
Demonstration Kuwahara filter (*)

- Open image snowdon.jl
- Apply on image KuwaharaFilter 3 EdgeExtend
- Apply on image KuwaharaFilter 8 EdgeExtend
- Apply Sobel on original image
- Apply Sobel on KuwaharaFilter with radius 8

Original image (*)
Kuwahara with radius 3 (*)

Kuwahara with radius 8 (*)
Sobel on original image (*)

Sobel on Kuwahara with radius 8 (*)
Symmetric Nearest Neighbour (SNN) filter (*)

SNN_MeanFilter (srcImage, destImage, radius, edge)
SNN_MedianFilter (srcImage, destImage, radius, edge)

SNN compares symmetric pairs of pixels within a defined radius with the center pixel. For each pair of pixels the one which is closest in value to the center pixel is calculated.

For the SNN_MeanFilter the new pixel value assigned to the center pixel is the mean of the closest pixels.

For the SNN_MedianFilter the new pixel value assigned to the center pixel is the median of the closest pixels.

Demonstration SNN filter (*)

- Open image snowdon.jl
- Apply on image SNN_MeanFilter 3 EdgeExtend
- Apply on image SNN_MeanFilter 8 EdgeExtend
- Apply on image SNN_MedianFilter 8 EdgeExtend
- Note: not much difference between Mean and Median
Computer Vision: Non linear filters

Original image (*)

SNN_Mean with radius 3 (*)

Jaap van de Loosdrecht, NHL, vdLMV, j.van.de.loosdrecht@nhl.nl
SNN_Mean with radius 8 (*)

SNN_Median with radius 8 (*)
“Texture” (*)

If brightness is interpreted as elevation then a variation in brightness is called a texture. A texture is a measure of surface roughness.

- **Range filter:**
  the new value is the difference between the maximum value and the minimum value of the selected pixels under the mask.
- **Variance filter:**
  the new value is the square root of the sum of the squares of the difference between values of the central pixel and its neighbours.

Both filters give a primitive measurement for texture. Looks similar to edge detection, but gives a lower response on the edges.

Demonstration “texture” (*)

- Open image circles.jl
- Apply both operations with EdgeExtend to both images
Range filter (*)

Variance filter (*)
Motion detection (by Dick Bruin) (*)

Overview:

- Sequences of frames
- Difference of adjacent frames
- Difference with background
- Adaptive mean
- Adaptive median
- Exercise

Sequences of frames (*)

Background

Motion
Difference of adjacent frames (*)

Difference with background (*)
Difference with background (*)

The left background contains the pixel wise median of the frames
The right background contains the pixel wise mean of the frames

These calculations are done off line
These calculations are heavy (especially the median)

Adaptive mean (*)

The left background is the pixel wise mean of the frames
The right background is an adaptive mean

\[ \text{mean'} = \frac{49 \times \text{mean} + \text{frame}}{50} \]

Old frames fade away
This mean adapts to changes of the background (sun light!)
Adaptive median (*)

The left background is the pixel wise median of the frames.

The right background is an adaptive median:

\[
\begin{align*}
m'[x, y] &= m[x, y] - 1, & \text{if } f[x, y] < m[x, y] \\
&= m[x, y] + 1, & \text{if } f[x, y] > m[x, y] \\
&= m[x, y], & \text{if } f[x, y] = m[x, y]
\end{align*}
\]

\(m[x, y]\) is a pixel of the adaptive median.

\(f[x, y]\) is a pixel of the current frame.

Exercise motion detection 1 (*)

Use motion_diff.js to experiment with the difference of adjacent frames method:

```javascript
$cam = XPSCam
Snapshot $cam Int16Image 0 prev
while true do
    Snapshot $cam Int16Image 0 img
    Copy prev diffprev
    Difference diffprev img
    Copy img prev
    Threshold diffprev 60 1000
    display img
display diffprev
endwhile
```

Jaap van de Loosdrecht, NHL,vdLMV, j.van.de.loosdrecht@nhl.nl
Exercise motion detection 2 (*)

Adapt script for the methods:
• difference with adaptive mean
• difference with adaptive median

Answers: motion_adapt_mean.jls and motion_adapt_median.jls

Optional:
using a threshold on the “difference of frames” method and multiplying the sum of the pixels by a factor X an estimate of the amount of movement can be made
find the best X for the sequence of frames