HA NOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Computer Vision

Ch3.1: Image enhancement – Image filtering

Nguyễn Thị Oanh
oanhnt@soict.hust.edu.vn

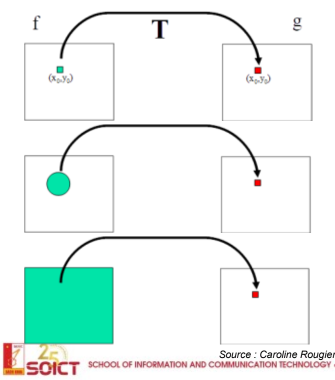
Computer Vision

Ch3.1: Image Enhancement – Image filtering

Content

- Remind: digital image
- Point operators: Contrast enhancement
- Other point operators
- Convolution and spatial filtering (local transformation)
  - Convolution
  - Linear spatial filtering
  - Other filters
- [Binary operator]

Pixel transformation



Isolated: $g(x_0,y_0)=T[f(x_0,y_0)]$

Local: $g(x_0,y_0)=T[f(V)]$
V:neighbors of (x_0,y_0)

Global: $g(x_0,y_0)=T[f(x,y)]$
example: FFT

Source : Caroline Rougier. Traitement d'images (IFT2730). Univ. de Montréal.

Convolution and spatial filtering

- Spatial filtering
  - affects directly to pixels in image
  - **Local transformation** in the spatial domain can be represented as: g(x,y) = T(f [V(x,y)])
    - T may affect the point (x,y) and its neighborhood (K) and output a new value

    (**K**: Filter/Mask/Kernel/Window/Template Processing)
  - **Same function** applied at each position
  - Output and input image are typically **the same size**
- Convolution: Linear filtering, function is a weighted sum of pixel values

I' = I * K

Spatial Convolution

- Convolution: Linear filtering, function is a weighted sum/difference of pixel values

$$I'(n,m) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} I[k,l] \times K[n-k, m-l]$$

Spatial convolution

Original image

*

Mask (kernel)

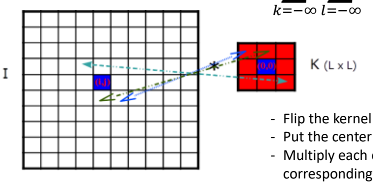
=

Filtered image

Spatial convolution

- Convolution: New value of a pixel(i,j) is a weighted sum of its neigbors

$$I'(n,m) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} I[k,l] \times K[n-k, m-l]$$

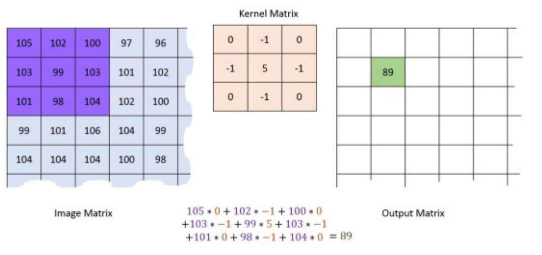


- Flip the kernel both horizontally and vertically.
- Put the center of kernel at each pixel (i,j).
- Multiply each element of the kernel with its corresponding element of the image matrix
- Sum up all product outputs

$$I'(i,j) = \sum_{u=-(L-1)/2}^{(L-1)/2} \sum_{v=-(L-1)/2}^{(L-1)/2} I(i-u, j-v)\, K(u,v)$$

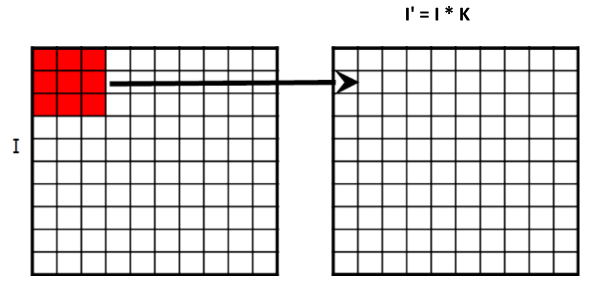
Spatial convolution

- New value of a pixel(i,j) is a weighted sum of its neigbors

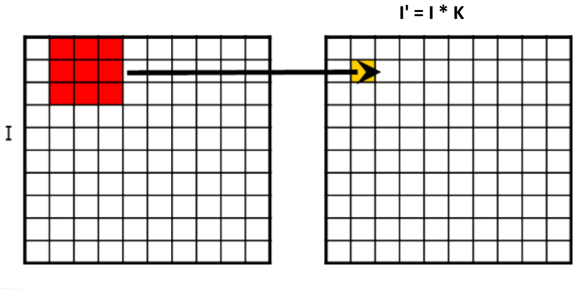


Source: http://machinelearninguru.com

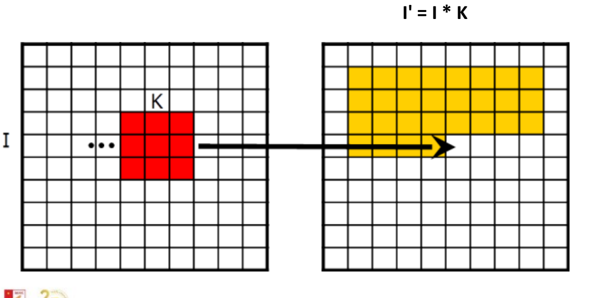
Spatial convolution



Spatial convolution

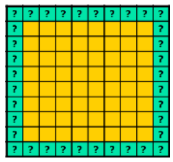


Spatial convolution



Spatial convolution

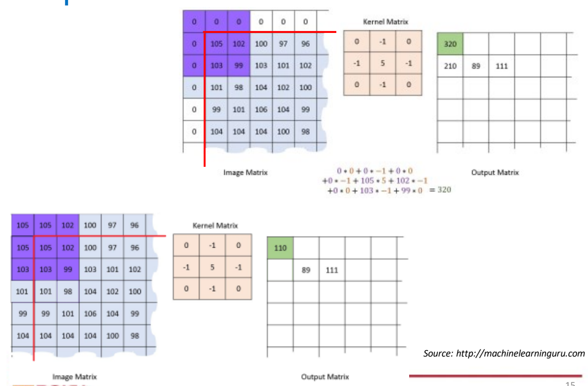
- Border problem?
  - Ignore, set to 0
  - Zero padding on the input matrix
  - reflect across edge:
    - f(-x,y) = f(x,y)
    - f(-x,-y) = f(x,y)



0	0	0	0	0	0
0	105	102	100	97	96
0	103	99	103	101	102
0	101	98	104	102	100
0	99	101	106	104	99
0	104	104	104	100	98

105	105	102	100	97	96
105	105	102	100	97	96
103	103	99	103	101	102
101	101	98	104	102	100
99	99	101	106	104	99
104	104	104	104	100	98

Spatial convolution



Spatial Correlation vs Convolution

Spatial Correlation (☆) and Convolution (★)

$$w(x,y)\bigstar f(x,y)=\sum_{s=-a}^{a}\sum_{t=-b}^{b}w(s,t)f(x-s,y-t)$$

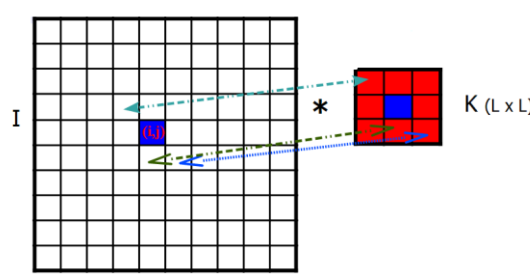
$$w(x,y)\star f(x,y)=\sum_{s=-a}^{a}\sum_{t=-b}^{b}w(s,t)f(x+s,y+t)$$

$$\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}\Rightarrow\begin{bmatrix}9&8&7\\6&5&4\\3&2&1\end{bmatrix}$$

Convolution is correlation with the filter rotated 180 degrees.

Spatial Correlation vs Convolution

- Correlation



Spatial Correlation vs Convolution

- If the mask is symmetric these two operations are identical
- Correlation:
  - Use to find which part in image match with a certain “template”
  - Not associative ➔ if you are doing template matching, i. e. looking for a single template, correlation is sufficient
- Convolution:
  - Use for filtering image (noise removing, enhancement, ..)
  - Associative: allows you to "pre-convolve" the filters ➔ useful when you need to use multiple filters in succession:
    I*h*g = I * (h*g)
- In Matlab: correlation: filter2, convolution: conv2

Examples

Original image

*

0	0	0
0	1	0
0	0	0

→

Filtered image
(no change)

Original image

*

0	0	0
1	0	0
0	0	0

→

Filtered image
(shifted left by 1 pixel)

Source: David Lowe

Some kernels

- 2D spatial convolution
  - is mostly used in image processing for feature extraction
  - And is also the core block of Convolutional Neural Networks (CNNs)
- Each kernel has its own effect and is useful for a specific task such as
  - blurring (noise removing),
  - sharpening,
  - edge detection,
  - …..

Smooth filtering

- "Low-pass" filters :
  - Used for noise removing/blurring,…
  - mean filters, Gaussian filers, …
- Average filtering is the simplest of smooth filtering
- To avoid changing the image brightness, the sum of the coefficients must be equal to 1

Average (mean) filtering

f(x-1,y-1)	*f(x-1,y)*	*f(x-1,y+1)*
f(x,y-1)	*f(x,y)*	*f(x,y+1)*
f(x+1,y-1)	*f(x+1,y)*	*f(x+1,y+1)*

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

Average mask

$$g(x,y) = \frac{1}{9}[f(x-1,y-1) + f(x-1,y) + f(x-1,y+1) + f(x,y-1) + f(x,y) + f(x,y+1) + f(x+1,y-1) + f(x+1,y) + f(x+1,y+1)]$$

Average (mean) filtering

- Mask:
  - All elements of the mask are equal

1/9 x

1	1	1
1	1	1
1	1	1

- Results:
  - Replacing each pixel with an average of its neigborhood
  - Achieve smoothing effect

Original image

Filtered image with box size 5x5

Filtered image with box size 11x11

Weighted Average filtering

- The pixel corresponding to the center of the mask is more important than the other ones.

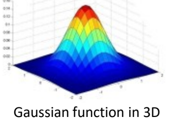
$\frac{1}{16}$

1	2	1
2	4	2
1	2	1

$$g(x,y) = \frac{1}{16}[f(x-1,y-1) + 2f(x-1,y) + f(x-1,y+1) + 2f(x,y-1) + 4f(x,y) + 2f(x,y+1) + f(x+1,y-1) + 2f(x+1,y) + f(x+1,y+1)]$$

Gaussian filtering

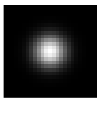
- Gaussian filter



Gaussian function in 3D

0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

Gaussian filter with size 5 x5 , sigma =1



Gaussian image

$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Rule for Gaussian filter:
set **filter width to about 6σ or 8σ [+1]**
(+1 if output is even number)

Gaussian filtering

Original image

Filtered image
with box size 5x5

Filtered image
with box size 11x11

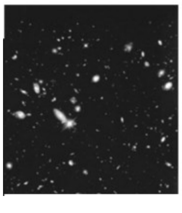
Gaussian filter

- Gaussian filter:
  - **Low-pass filter**: Remove high-frequency component from the image
    - Image becomes more smooth
    - Better than mean filter
  - Convolution with itself is an other Gaussian
    - Repeat the conv. With small-width kernel => get the same result as larger-width kernel would have.
    - Convolving **2 times** with Gaussian kernel of **width σ** is the same as convolving **once with kernel of width $\sigma\sqrt{2}$** : $I*G_\sigma*G_\sigma = I * G_{\sigma\sqrt{2}}$
  - **Separable filter**: The 2D Gaussian can be expressed as the product of 2 functions : one function of x and a function of y:
    - $G_\sigma(x,y) = G_\sigma(x).G_\sigma(y)$

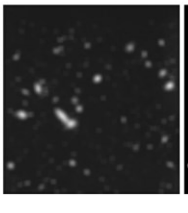
Smooth filtering

- Bluring usage: delete unwanted (small) subjects

Original image



Average filtering: 15x15



Thresholding of blurring image

Sharpening filter

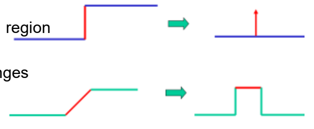
- Highlighs Intensity Transitions
- Base on first and second order derivatives

$$\frac{\partial f}{\partial x} \approx \begin{cases} f(x+1,y)-f(x,y) \\ f(x,y)-f(x-1,y) \\ 0.5\left(f(x+1,y)-f(x-1,y)\right) \end{cases}$$

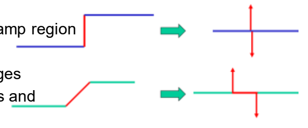
$$\frac{\partial^2 f}{\partial x^2} \approx f(x+1,y)-2f(x,y)+f(x-1,y)$$

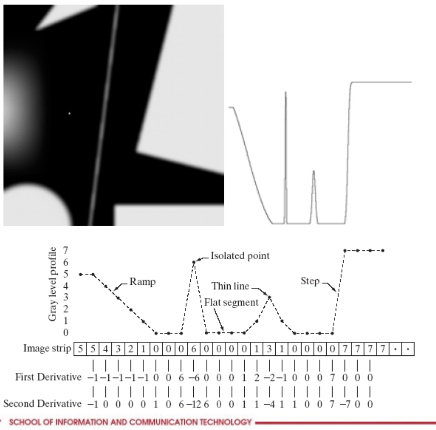
First and second order derivative

- First order derivative:
  - Zero in flat region
  - Non-zero at start of step/ramp region
  - Non-zero along ramp
  - Strong response for step changes



- Second order derivative:
  - Zero in flat region
  - Non-zero at start/end of step/ramp region
  - Zero along ramp
  - Double response at step changes
  - Strong response for fine details and isolated points;





Derivative

- 1st and 2nd Order Derivative Comparison:
  - First Derivative:
    - Thicker Edge;
    - Strong Response for step changes;
  - Second Derivative:
    - Strong response for fine details and isolated points;
    - Double response at step changes

First derivatives

- Filters used to compute the first derivatives of image
  - Robert

1	0
0	-1

0	1
-1	0

  - Prewitt
    - less sensitive to noise
    - Smoothing with mean filter,

then compute1st derivative

-1	-1	-1
0	0	0
1	1	1

-1	0	1
-1	0	1
-1	0	1

  - Sobel:
    - less sensitive to noise
    - Smoothing with gaussian,

then computing1st derivative

-1	-2	-1
0	0	0
1	2	1

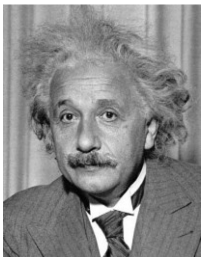
y

-1	0	1
-2	0	2
-1	0	1

x

First derivatives

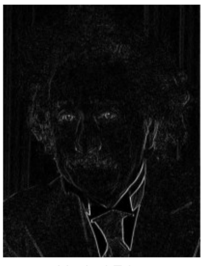
- Sobel



$*$

-1	0	1
-2	0	2
-1	0	1

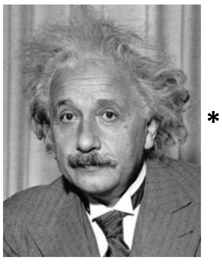
→



Vertical Edge
(absolute value)

First derivatives

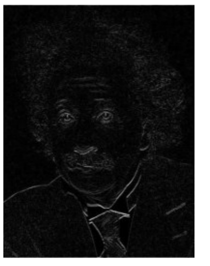
- Sobel



$*$

-1	-2	-1
0	0	0
1	2	1

→



Horizontal Edge
(absolute value)

2nd derivatives - Laplacian filtering

- The Laplacian operator is defined as:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

$$\frac{\partial^2 f}{\partial x^2} = f(x+1,y) + f(x-1,y) - 2f(x,y)$$

$$\frac{\partial^2 f}{\partial y^2} = f(x,y+1) + f(x,y-1) - 2f(x,y)$$

2nd derivatives - Laplacian filtering

- Can be computed as

$$\nabla^2 f = [f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1)] - 4f(x,y)$$

0	1	0
1	-4	1
0	1	0

- Or

$$\nabla^2 f = 4f(x,y) - [f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1)]$$

0	-1	0
-1	4	-1
0	-1	0

- 90^o isotropic filter

2^{nd} derivatives - Laplacian filtering

- Can be computed as

$$\nabla^2 f = [f(x+1,y+1) + f(x+1,y) + f(x+1,y-1) + f(x-1,y+1) + f(x-1,y) + f(x-1,y-1) + f(x,y+1) + f(x,y-1)] - 8f(x,y)$$

1	1	1
1	-8	1
1	1	1

- Or

$$\nabla^2 f = 8f(x,y) - [f(x+1,y+1) + f(x+1,y) + f(x+1,y-1) + f(x-1,y+1) + f(x-1,y) + f(x-1,y-1) + f(x,y+1) + f(x,y-1)]$$

-1	-1	-1
-1	8	-1
-1	-1	-1

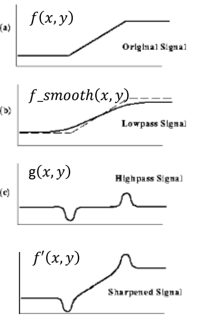
- 45° isotropic filter

Sharpening filter using first derivative

$$g(x,y) = f(x,y) - f_{smooth}(x,y)$$

$$f'(x,y) = f(x,y) + k * g(x,y)$$

(k = 0.2..0.7)



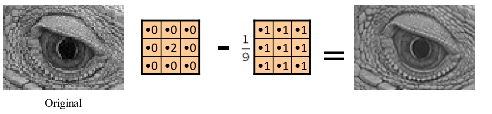
Sharpening filter using first derivative

What does blurring take away?

- Let's add it back:

Juan Carlos Niebles and Ranjay Krishna

Sharpening filter

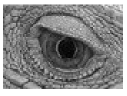


Original

Sharpening filter: Accentuates differences with local average

Juan Carlos Niebles and Ranjay Krishna

Sharpening filter


Original

$$\begin{bmatrix}0&0&0\\0&2&0\\0&0&0\end{bmatrix} - \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix} = ?$$

(Note that filter sums to 1)

$$\begin{bmatrix}0&0&0\\0&1&0\\0&0&0\end{bmatrix} + \underbrace{\begin{bmatrix}0&0&0\\0&1&0\\0&0&0\end{bmatrix} - \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}}_{\text{“details of the image”}}$$

Juan Carlos Niebles and Ranjay Krishna

Sharpening filter using laplacian

Popular method: original image – scaled seconde derivative

$$g(x,y)=\begin{cases} f(x,y)-\nabla^2 f(x,y) & -sign \\ f(x,y)+\nabla^2 f(x,y) & +sign \end{cases}$$

$$\begin{bmatrix}0&-1&0\\-1&+5&-1\\0&-1&0\end{bmatrix}\ 90^\circ\ isotropic \qquad \begin{bmatrix}-1&-1&-1\\-1&+9&-1\\-1&-1&-1\end{bmatrix}\ 45^\circ\ isotropic$$

Note

$$g(x,y)=\begin{cases} f(x,y)-\nabla^2 f(x,y) \\ f(x,y)+\nabla^2 f(x,y) \end{cases}$$

$$\begin{bmatrix}0&-1&0\\-1&5&-1\\0&-1&0\end{bmatrix} = \begin{bmatrix}0&0&0\\0&1&0\\0&0&0\end{bmatrix} + \begin{bmatrix}0&-1&0\\-1&4&-1\\0&-1&0\end{bmatrix}$$

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

Sharpening filter

a) Original image

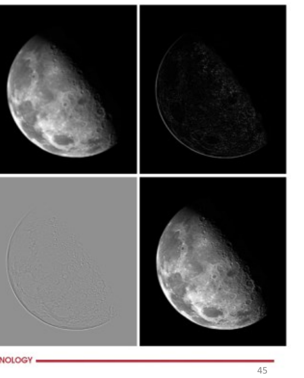
b) Laplacian filtering

c) Laplacian with scaling

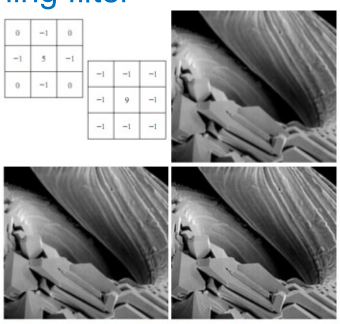
d) Adding original with Laplacian image

a b
c d

FIGURE 3.40 (a) Image of the North Pole of the moon. (b) Laplacian-filtered image. (c) Laplacian image scaled for display purposes. (d) Image enhanced by using Eq. (3.7-5). (Original image courtesy of NASA.)



Sharpening filter



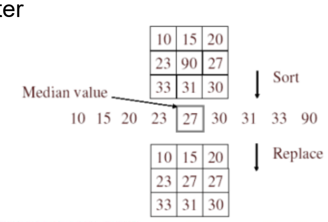
a b c
d e

FIGURE 3.41 (a) Composite Laplacian mask. (b) A second composite mask. (c) Scanning electron microscope image. (d) and (e) Results of filtering with the masks in (a) and (b), respectively. Note how much sharper (e) is than (d). (Original image courtesy of Mr. Michael Shaffer, Department of Geological Sciences, University of Oregon, Eugene.)

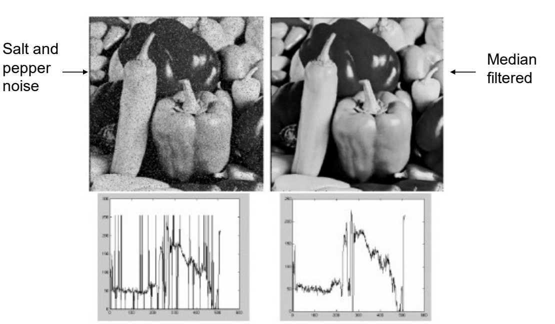
Other filters (non-linear filtering)

Median filter

- No new pixel values introduced
- Removes spikes:
  - good for impulse, salt & pepper noise
- Non-linear filter



Median filter

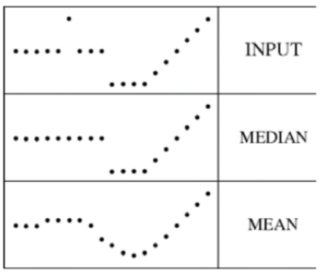


Plots of a row of the image

Source: M. Hebert

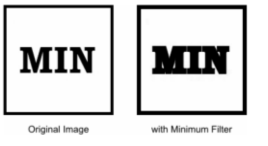
Median filter

- Median filter is edge preserving

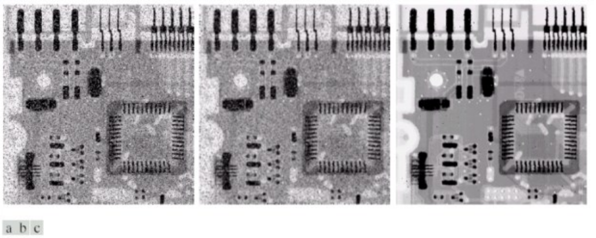


Max/ Min filters

- The maximum and minimum filters are shift-invariant
  - The Minimum Filter: replaces the central pixel with the darkest one in the running window

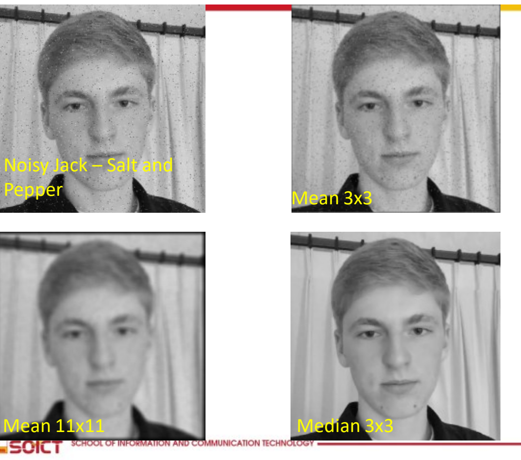


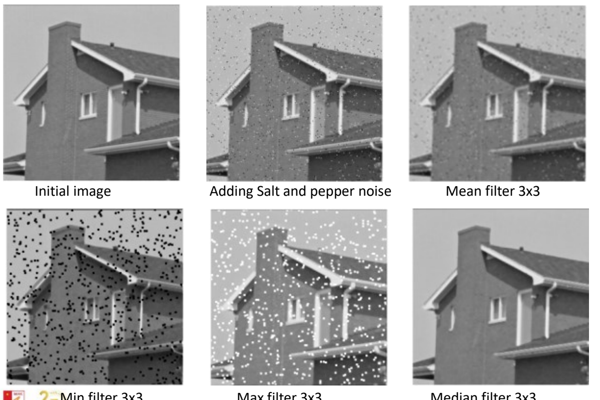
  - The Maximum Filter: replaces the central pixel with the lightest one



a b c

FIGURE 3.37 (a) X-ray image of circuit board corrupted by salt-and-pepper noise. (b) Noise reduction with a 3 × 3 averaging mask. (c) Noise reduction with a 3 × 3 median filter. (Original image courtesy of Mr. Joseph E. Pascente, Lixi, Inc.)





Initial image
Adding Salt and pepper noise
Mean filter 3x3
Min filter 3x3
Max filter 3x3
Median filter 3x3

Median/max/min: problem?

- Size filter !

Exercise

- Given an 8-bit image – 8 x 8

$$\begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}$$

1) Compute and show the histogram: 8 bins, 16 bins, 32 bins

2) Comment about the contrast of the image

3) Equalize the histogram for above image with 8-bins

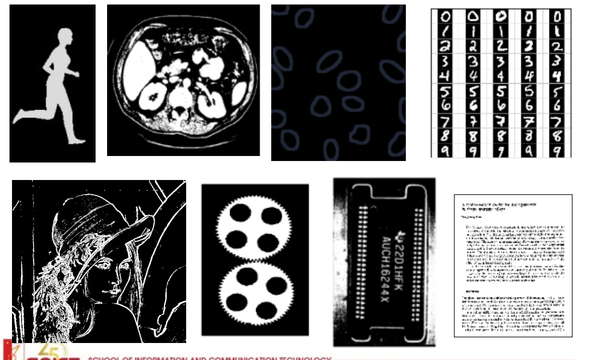
Practice (home work)

- Try to implement/use different filters to apply on images:
  - Mean, gaussian filters
  - First and second image derivative (sobel, prewitt, Laplacian filters)
  - Sharpening filters
  - Median/max/min filters
- Modify the kernel size to see its effet

Content

- Remind: digital image
- Point operators: Contrast enhancement
- Other point operators
- Convolution and spatial filtering (local transformation)
  - Convolution
  - Linear spatial filtering
  - Other filters
- Binary images

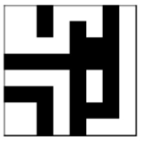
Binary images



Binary images

- Two pixel values: foreground (object, 1) and background (0)
- Be used
  - To mark region(s) of interest
  - As results of thresholding method

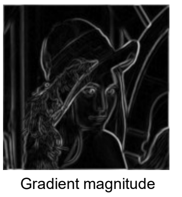
1	1	0	1	1	1	0	1
1	1	0	1	0	1	0	1
1	1	1	1	0	0	0	1
0	0	0	0	0	0	0	1
1	1	1	1	0	1	0	1
0	0	0	1	0	1	0	1
1	1	0	1	0	0	0	1
1	1	0	1	0	1	1	1



Thresholding

- Given a grayscale image or an intermediate matrix → threshold to create a binary output.

Example: edge detection



Gradient magnitude

`fg_pix = find(gradient_mag > t);`

Looking for pixels where gradient is strong.

Thresholding

- Given a grayscale image or an intermediate matrix → threshold to create a binary output.

Example: background subtraction

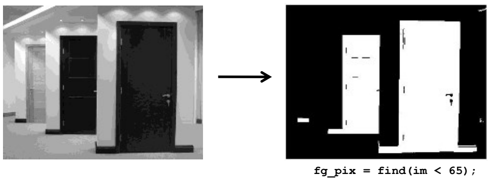
Looking for pixels that differ significantly from the “empty” background.

`fg_pix = find(diff > t);`

Thresholding

- Given a grayscale image or an intermediate matrix → threshold to create a binary output.

Example: intensity-based detection



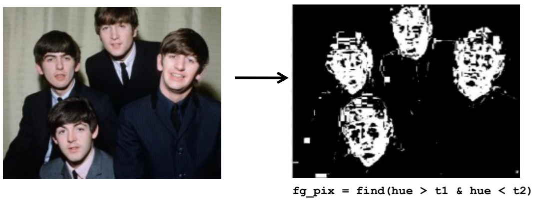
`fg_pix = find(im < 65);`

Looking for dark pixels

Thresholding

- Given a grayscale image or an intermediate matrix → threshold to create a binary output.

Example: color-based detection



`fg_pix = find(hue > t1 & hue < t2);`

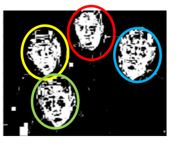
Looking for pixels within a certain hue range.

Slide credit: Kristen Grauman

Issues

- What to do with “noisy” binary outputs?
  - Holes
  - Extra small fragments

- How to demarcate multiple regions of interest?
  - Count objects
  - Compute further features per object



Slide credit: Kristen Grauman

Morphological operators

- Change the shape of the foreground regions via intersection/union operations between a scanning structuring element and binary image.
- Useful to clean up result from thresholding

- Main components
  - Structuring element
  - Operators:
    - Basic operators: Dilation, Erosion
    - Others: Opening, Closing, …

Dilation

- Expands connected components
- Grow features
- Fill holes

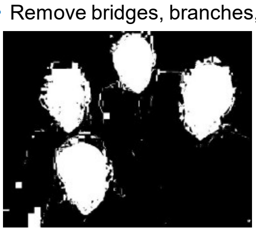
Before dilation

After dilation

Slide credit: Kristen Grauman

Erosion

- Erode connected components
- Shrink features
- Remove bridges, branches, noise



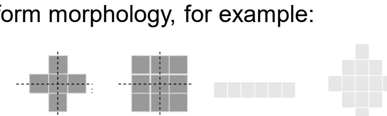
Before erosion

After erosion

Slide credit: Kristen Grauman

Structuring elements

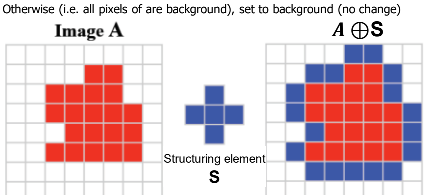
- **Masks** of varying shapes and sizes used to perform morphology, for example:



- Scan mask (structuring element) over the object (foreground) borders (inside and outside) and transform the binary image

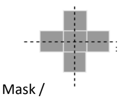
Dilation

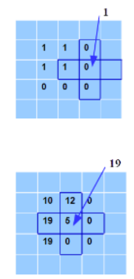
- Moving S on each pixel of A
  - check if the intersection (pixels belonging to object) is not empty
  - If yes, the center of B belongs to the result image
- If a pixel of S is onto object pixels (A), then the central pixel belongs to object
  - Otherwise (i.e. all pixels of are background), set to background (no change)



Dilation

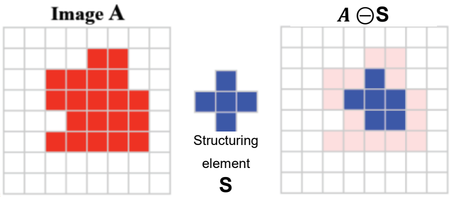
- As max filter
- Can be applied both on
  - binary images
  - or grayscale images





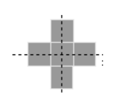
Erosion

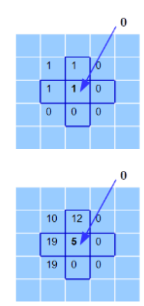
- We put the element S on each pixel x of A
  - like convolution
- If all pixels of S are onto object pixels (A), then the central pixel belongs to object
  - Otherwise (i.e. a mask pixel is background), set to background



Erosion

- As min filter
- Can be applied both on
  - binary images
  - or grayscale images





Opening

- Erode, then dilate
- Remove small objects, keep original shape

Before opening

After opening

Closing

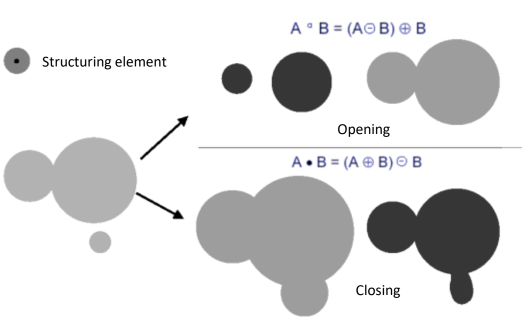
- Dilate, then erode
- Fill holes, but keep original shape

Before closing

After closing

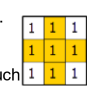
demo: http://bigwww.epfl.ch/demo/jmorpho/start.php

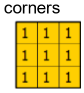
Opening vs Closing

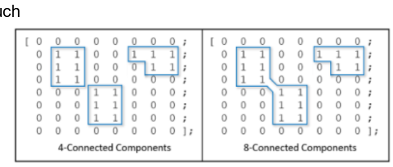


Connected component labeling

- A **connected component,** or an object, in a binary image is a **set of adjacent pixels**. Determining which pixels are adjacent depends on how pixel connectivity is defined.
- There are 2 standard connectivities:
  - 4-connectivity — Pixels are connected if their edges touch
  - 8-connectivity — Pixels are connected if their edges or corners touch



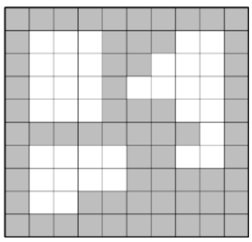




Connected component labeling

- We loop over all the image to give a unique number (label) for each region
- All pixels from the same region must have the same number (label)
- Objectifs:
  - Counting objects
  - Separating objets
  - Creating a mask for each object
  - …

← *Background*
← *Segmented objects*



Connected component labeling

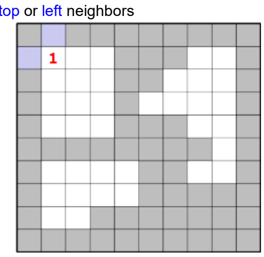
First loop over the image

- For each pixel in a region, we set
  - or the smallest label from its top or left neighbors
  - or a new label

Loop

Neighbors

	X
X	?



Connected component labeling

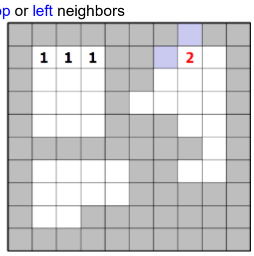
First loop over the image

- For each pixel in a region, we set
  - or the smallest label from its top or left neighbors
  - or a new label

Loop

Neighbors

	X
X	?



Connected component labeling

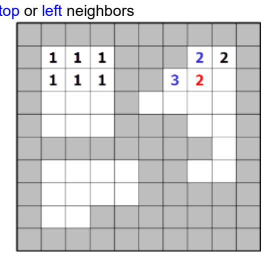
First loop over the image

- For each pixel in a region, we set
  - or the smallest label from its top or left neighbors
  - or a new label

Loop

Neighbors

	X
X	?



Connected component labeling

First loop over the image

- For each pixel in a region, we set
  - or the smallest label from its top or left neighbors
  - or a new label

Loop

Neighbors

X

X ?

	1	1	1				2	2	
	1	1	1			3	2	2	
	1	1	1		4	3	2	2	
	1	1	1				2	2	
								2	
	5	5	5	5			6	2	
	5	5	5	5					
	5	5							

Connected component labeling

Second loop over the image

- For each pixel in a region, we set
  - the smallest from its own label and the labels from its down and right neighbors

Loop

Neighbors

X X

X

	1	1	1				2	2	
	1	1	1			3	2	2	
	1	1	1		4	3	2	2	
	1	1	1				2	2	
								2	
	5	5	5	5			6	2	
	5	5	5	5					
	5	5							

Connected component labeling

Second loop over the image

- For each pixel in a region, we set
  - the smallest from its own label and the labels from its down and right neighbors

Loop

Neighbors

X X

X

	1	1	1				2	2	
	1	1	1			3	2	2	
	1	1	1		4	3	2	2	
	1	1	1				2	2	
								2	
	5	5	5	5			2	2	
	5	5	5	5					
	5	5							

Connected component labeling

Second loop over the image

- For each pixel in a region, we set
  - the smallest from its own label and the labels from its down and right neighbors

Loop

Neighbors

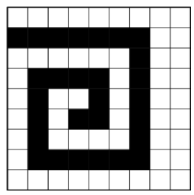
X X

X

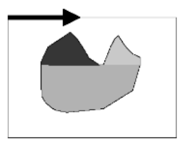
	1	1	1				2	2	
	1	1	1			2	2	2	
	1	1	1		2	2	2	2	
	1	1	1				2	2	
								2	
	5	5	5	5			2	2	
	5	5	5	5					
	5	5							

Connected component labeling

- Two loops are enough?
  - *example: spiral region !*



- Solutions
  - We continue, go and back two ways, until no new change in labels
  - It is possible to do only one loop: manage a table of equivalences when 2 different labels are neighbors

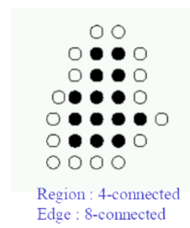


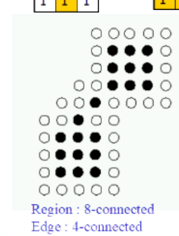
CC labeling: how many neighbors?

- Advice: Use different connectivity for edges and regions
  - *4-Connectivity for regions*
  - *8-Connectivity for edges*

1	1	1
1	1	1
1	1	1

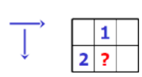
1	1	1
1	1	1
1	1	1



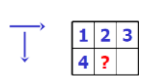


CC labeling: how many neighbors?

- Regions labeling
  - We use 4-connectivity
  - Each loop, we compare 2 neighbors



- Edge labeling
  - 8-connectivity
  - Each loop, we compare 4 neighbors



Next lesson

- Rappel: digital image representation
- Point Processing
- Convolution and Linear filtering
- More neighborhood operators
- Image transforms
  - Frequency domain
    - Frequencies in images
    - Fourrier transform
    - Frequential Processing (frequential filters)
  - PCA (additional reading)

References

- Lecture 2: CS131 - Juan Carlos Niebles and Ranjay Krishna, Stanford Vision and Learning Lab
- Vision par Ordinateur, Alain Boucher, IFI

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Thank you for your attention!

soict.hust.edu.vn/ fb.com/groups/soict