Image filters

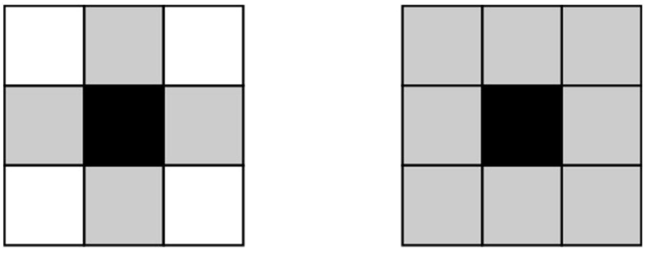
Manuel J. Marín-Jiménez (*Univ. of Córdoba*)

@mjmarinj

1

Pixel connectivity

4 neighbours vs 8 neighbours



2

Image filtering

Operation: Convolution

Ingredients:
- Image
- Kernel (size?)

Border conditions

The basis of CNNs!

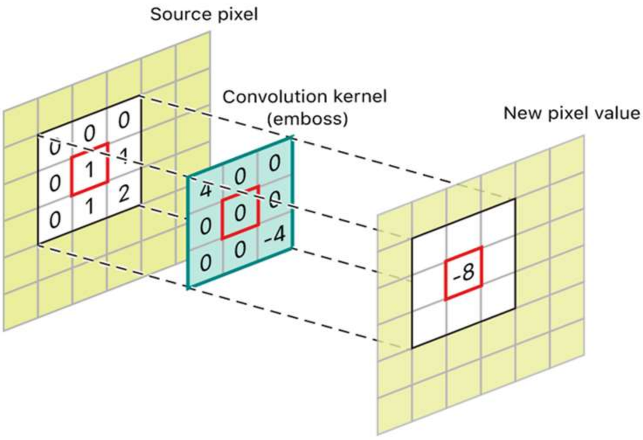


Image filtering

- **High frequencies**: fine details such as borders. Sensitive to noise
- **Low frequencies**: global information, mean. Robust to noise.
- **Low pass** filter: remove high frequencies.
- **High pass** filter: enhance high frequencies by reducing the relevance of low frequencies.

5

Convolution

- **Low-pass linear filters**

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

Box Filter

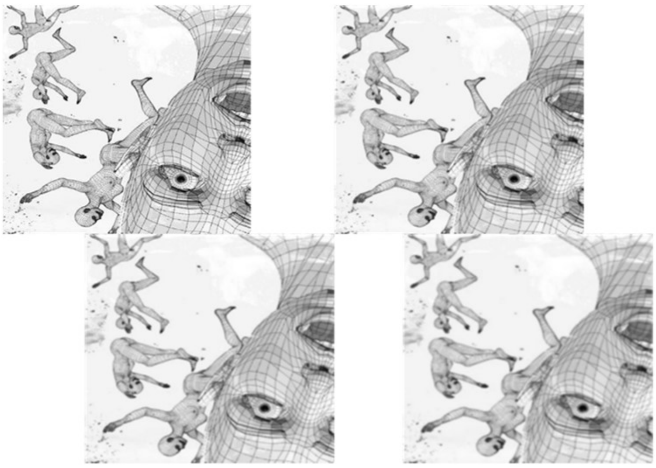
$$\frac{1}{16} \times \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Binomial Filter

6

Convolution

- Changing the kernel size (box filter)



7

Exercise

Apply (manually) a 3x3 box filter to the following image.

0	0	0	0	255	255	255	255
0	0	0	0	255	255	255	255
0	0	0	0	255	255	255	255
0	0	0	0	255	255	255	255
0	0	0	0	255	255	255	255
0	0	0	0	255	255	255	255

9

Exercise

Write your own convolutional operator in C++.

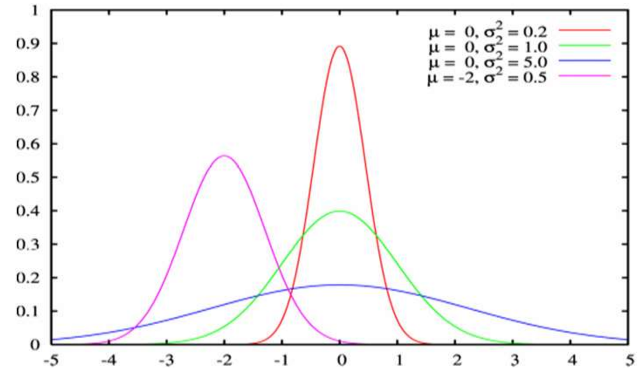
- Take into account the border conditions.
- Test your function with a kernel of your choice to see the result.

```
void fsiv_convolution(const cv::Mat & src, cv::Mat & dst,
     const cv::Mat & kernel, int padding)
{
    // Check if dst image is prepared or ...
    // prepare the output image.

    // Check the padding type

    // Apply the convolution operation
}
```

10

Gaussian-based kernels

- Gaussian distribution



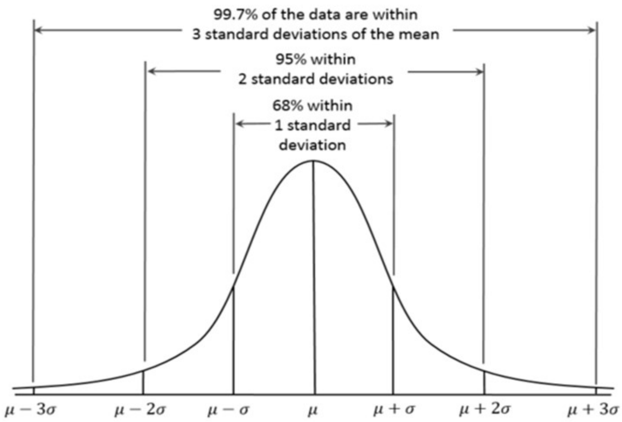
$$\frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

95.44 % of area is within 2 times the standard deviation

11

Gaussian-based kernels

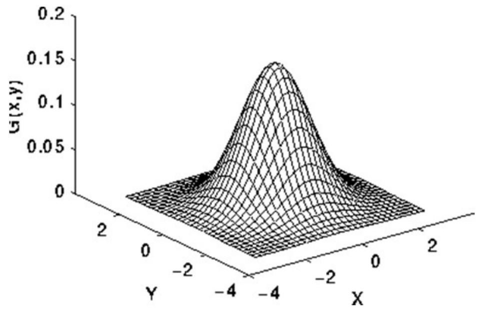
- Gaussian distribution: properties



12

Gaussian-based kernels

- 2D Gaussian distribution



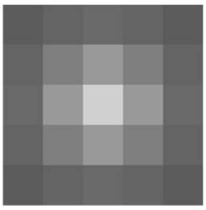
$$G(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} e^{\left(-\frac{1}{2}\left(\frac{(x-\mu_x)^2}{\sigma_x^2}+\frac{(y-\mu_y)^2}{\sigma_y^2}\right)\right)}$$

13

Gaussian-based kernels

- 2D Gaussian kernel
- Example: filter 5x5 with mean (0,0) and var (1,1)

0.00291502	0.0130642	0.0215393	0.0130642	0.00291502
0.0130642	0.0585498	0.0965324	0.0585498	0.0130642
0.0215393	0.0965324	0.159155	0.0965324	0.0215393
0.0130642	0.0585498	0.0965324	0.0585498	0.0130642
0.00291502	0.0130642	0.0215393	0.0130642	0.00291502



Gaussian-based kernels

- 2D Gaussian kernel
- Approximation of the function

$$\frac{1}{273}\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$$

Gaussian-based kernels

- Example

16

Exercise

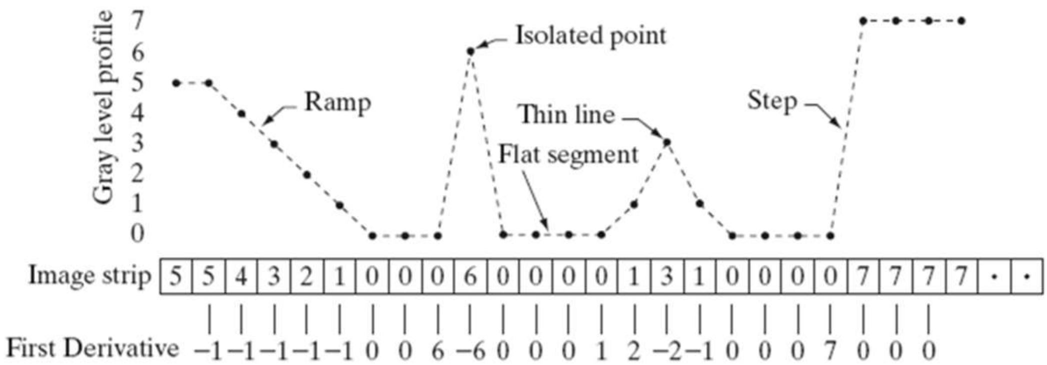
Based on your convolution function, create a **program** to apply Gaussian blurring to an image.

- Use the 5x5 Gaussian kernel previously seen.
- To visually find the differences between the input and output image, create a **function** ‘*fsiv_mat2img(cv::Mat)*’ that converts any matrix into a *uchar* image (i.e compatible with *cv::imshow()*).

17

Image derivatives

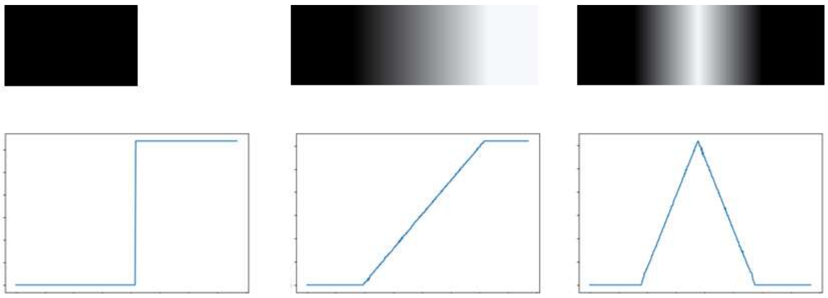
First order derivative
$$f'(x_i) = \lim_{\epsilon \to 0} \frac{f(x_i + \epsilon) - f(x_i)}{\epsilon}$$



18

Image derivatives

First order derivative $$f'(x_i) = \lim_{\epsilon \to 0} \frac{f(x_i + \epsilon) - f(x_i)}{\epsilon}$$

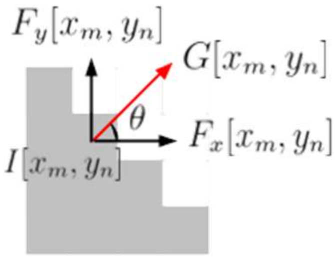


step edge ramp edge roof edge

19

Image derivatives

2D Gradient vector G: (Fx, Fy)



Magnitude: $$|G[x_m, y_n]| = \sqrt{F_x[x_m, y_n]^2 + F_y[x_m, y_n]^2}$$

Orientation: $$\theta = tan^{-1}(\frac{F_y[x_m, y_n]}{F_x[x_m, y_n]})$$

20

Image derivatives

Filter operators for gradient computation

Roberts	$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$		
Prewitt	$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 & -1 \\ 1 & 0 & -1 \\ 1 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$
Sobel	$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & -2 & -1 \\ 1 & 0 & -1 \\ 1 & 2 & 0 \end{bmatrix}$	$\begin{bmatrix} -1 & -2 & 0 \\ -1 & 0 & 1 \\ 0 & 2 & 1 \end{bmatrix}$

FSIV - mjmarin@uco.es

@mjmarinj

20

21

Image derivatives

Exercise: compute the magnitude of the derivative using Sobel's filter

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	255	255	255	255
0	0	0	0	255	255	255	255
0	0	0	0	255	255	255	255
0	0	0	0	255	255	255	255

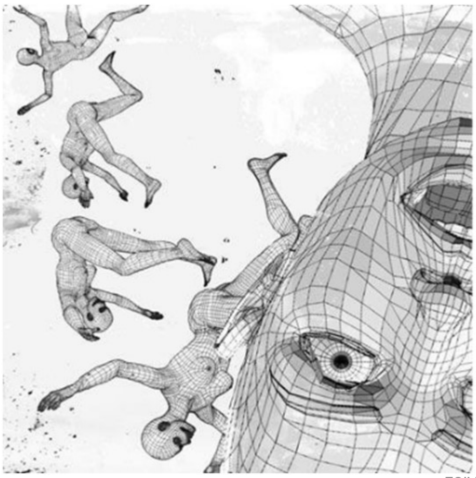
FSIV - mjmarin@uco.es

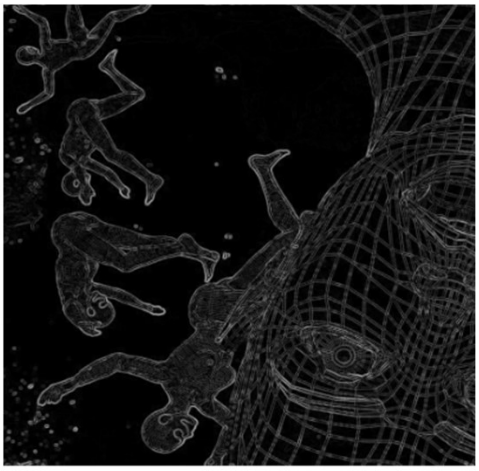
@mjmarinj

21

Image derivatives

Example





Exercise

Using either *cv::Filter2D* or your custom function, apply Roberts' filter to an input **grayscale** image.

- Compute the **magnitude** of the gradient vector and save it as an image.
- You may want to test the other filters (Prewitt and Sobel) as well.

Image derivatives

Second-order derivative: Laplacian of Gaussian

−0,11	−0,11	−0,11
−0,11	0,88	−0,11
−0,11	−0,11	−0,11

Image derivatives

Exercise: apply the previous filter to the given image

0	0	0	0	255	255	255	255
0	0	0	0	255	255	255	255
0	0	0	0	255	255	255	255
0	0	0	0	255	255	255	255
0	0	0	0	255	255	255	255
0	0	0	0	255	255	255	255

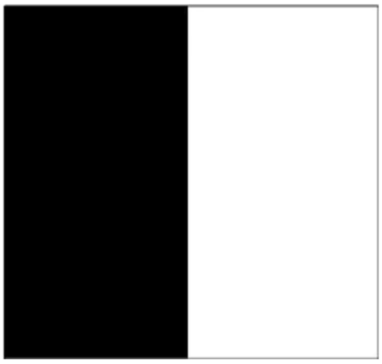
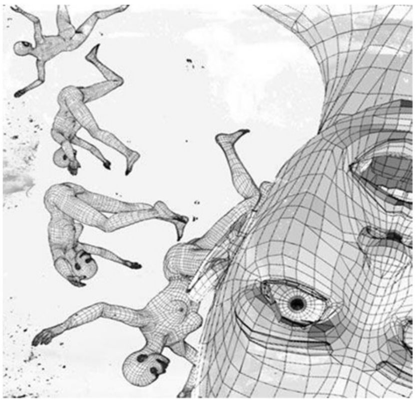
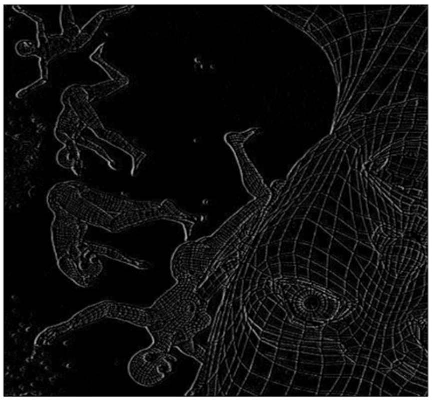


Image derivatives

Example: what frequencies are “passing”?





High-pass filters

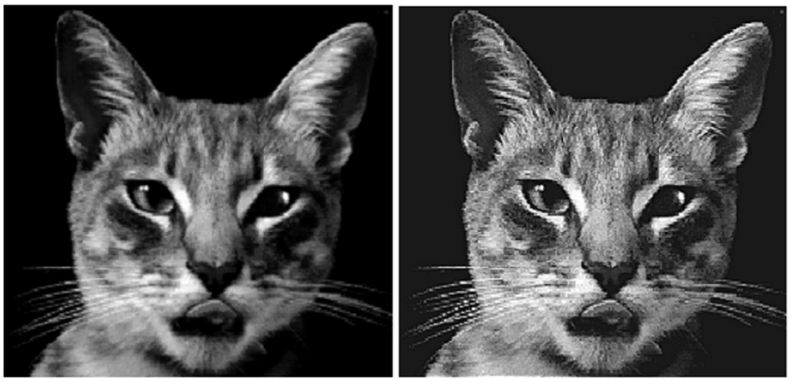
HighBoost filter

HighBoost ← Original Image + c · HighPassImage

$$\begin{bmatrix} -c & -c & -c \\ -c & 8c+1 & -c \\ -c & -c & -c \end{bmatrix}$$

High-pass filter

Example



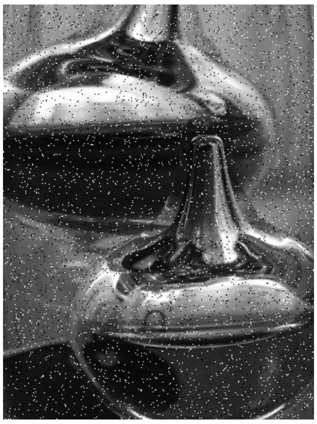
Neighbour-based processing

Non-linear filters

33

Salt-and-pepper noise

Impulse noise → sparse white and black pixels



FSIV - mjmarin@uco.es

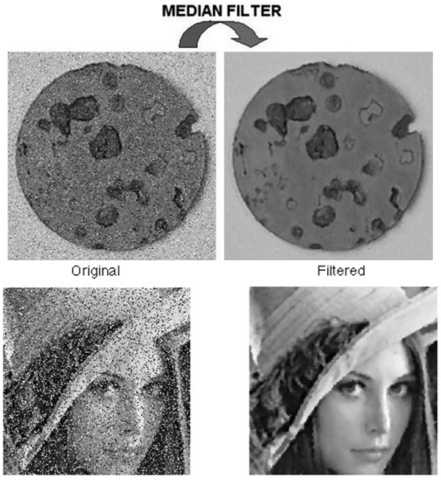
@mjmarinj

33

34

Median filter

Replace each pixel by the median of its neighborhood



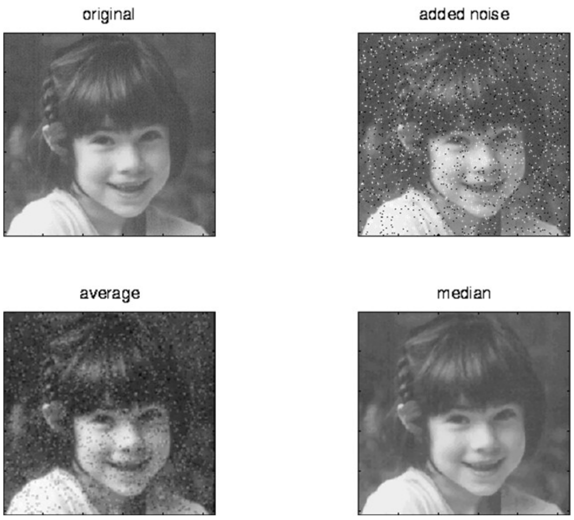
FSIV - mjmarin@uco.es

@mjmarinj

34

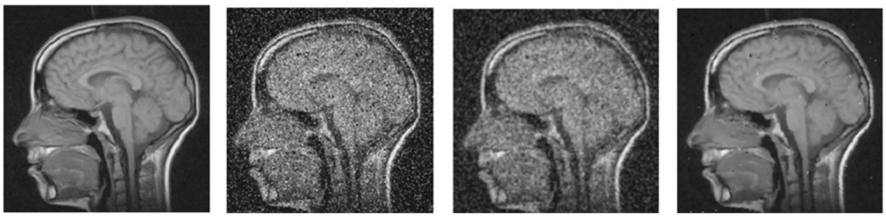
Median filter

Example



Median filter

Example



37

Bilateral filter

Blurring but keeping edges (high freq.)

38

Bilateral filter

Idea: weights proportional to distance and intensity

- *g_s* and *f_r* are Gaussian functions

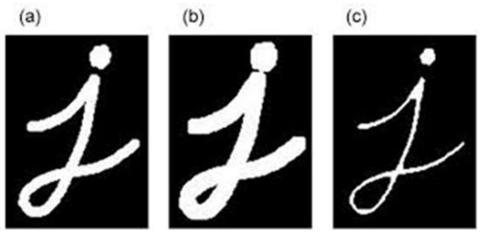
$$I^{\text{filtered}}(x) = \frac{1}{W_p} \sum_{x_i \in \Omega} I(x_i) f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|),$$

$$W_p = \sum_{x_i \in \Omega} f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|)$$

Morphological operations

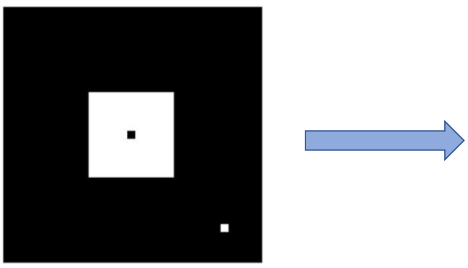
- Dilation: maximum value in the neighborhood. Grows objects (b)
- Erosion: minimum value in the neighborhood. Shrinks objects (c)

Applications: noise filtering, shape simplification, skeletonization ...



Morphological operations

- Opening: Erosion + Dilation
- Closing: Dilation + Erosion



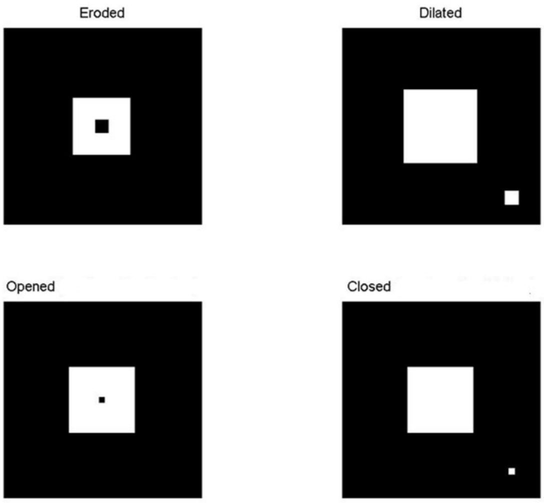


Image filters

Manuel J. Marín-Jiménez (*Univ. of Córdoba*)

@mjmarinj

41