UCLM
UNIVERSIDAD DE CASTILLA-LA MANCHA

Arquitectura de Computadores

Lab 3

Fernando.Rincón@uclm.es
Serafin.Benito@uclm.es

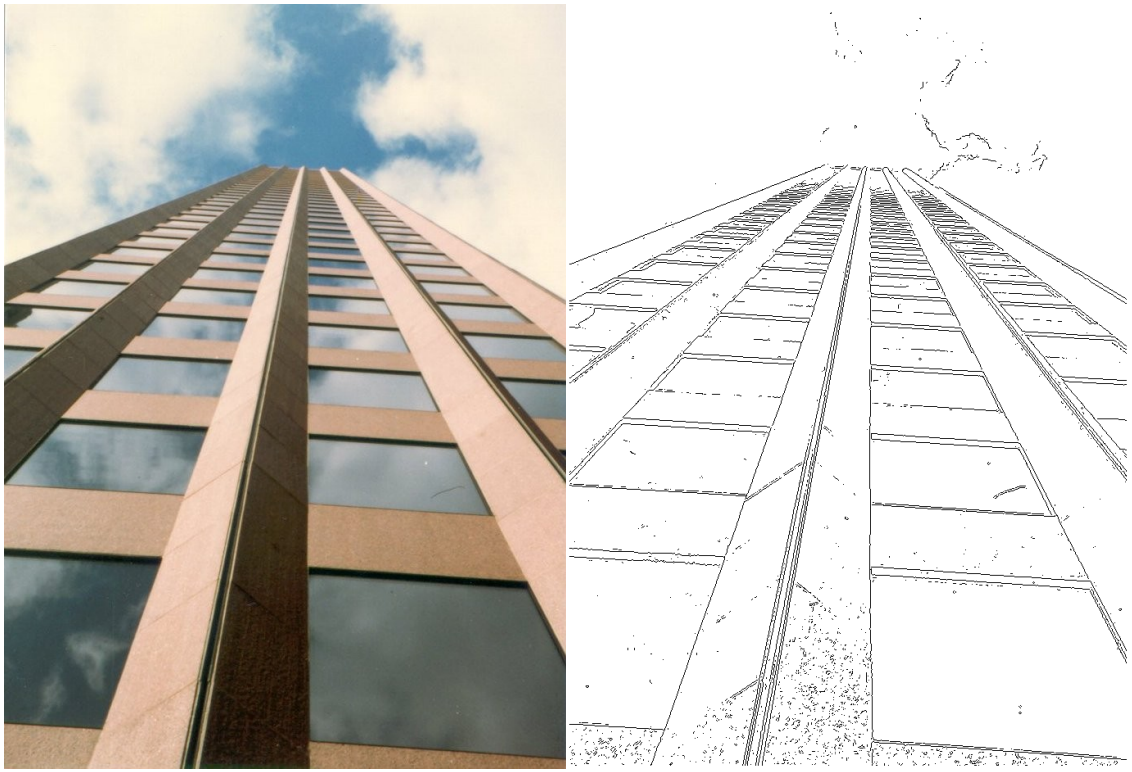
UCLM
UNIVERSIDAD DE CASTILLA-LA MANCHA

Outline

- Edge detection in an image
- Linear filtering $\rightarrow$ convolution product
- Sobel filter

Edge detection in images

- Fundamental algorithm in computer vision
- Based on the detection of sudden changes in the intensity values of the pixels



Linear filtering of images

- Most used method due to its simplicity
- Based on the convolution product
- Eachi pixel of the image is replaced by a value computed from the ones of its neighbours

10	5	3
4	5	1
1	1	7

$\otimes$

0	0	0
0	0.5	0
0	1.0	0.5

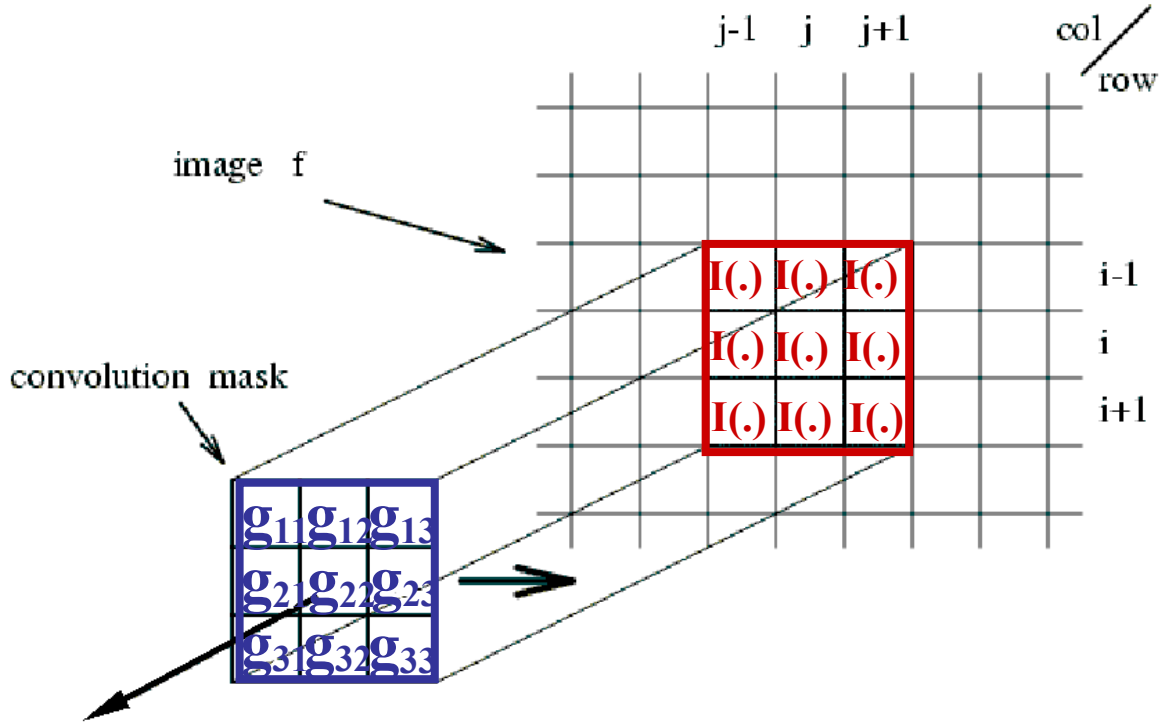
kernel

$=$

	7	

Convolución

- The convolution kernel is applied to the whole image



$$f(i,j) = g_{11}\, I(i-1,j-1) + g_{12}\, I(i-1,j) + g_{13}\, I(i-1,j+1) + g_{21}\, I(i,j-1) + g_{22}\, I(i,j) + g_{23}\, I(i,j+1) + g_{31}\, I(i+1,j-1) + g_{32}\, I(i+1,j) + g_{33}\, I(i+1,j+1)$$

Sobel

- For the sobel algorithm two kernels are applied

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

x filter

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

y filter

- The gradient (value) obtained after the convolution for every component is combined using:

$$|G| = \sqrt{Gx^2 + Gy^2}$$

Task 1: sobel1.cpp

- Write a new function `SobelParallel` resulting from the parallelization of `SobelBasico`
- Add the appropriate function call to `SobelParallel` from the main program:
  - Also measure the time spent
  - And check the correctness of the result
- Explore the schedule option of the for
  - Try the following options and choose the most efficient:
    - `schedule(static,tt)` and `schedule(dynamic,tt)`
  - For (tt) slices, test values 6 and *image_height / num_of_cores*
  - Leave in a comment the times for the 4 options while the code will belong to the best one

Task 2: sobel2.cpp

- Add to `sobel1.cpp` the following functions:
  - `SobelLocal`: optimization of the sequential version based on data locality of the algorithm:
    - At every shift of the kernel 6 pixels are reused, while only 3 need to be requested from memory
  - `SobelLocalParallel`: parallelization of `SobelLocal`
  - `SobelCompleto`: that, per each pixel, it uses **both kernels to compute the gradient** and, later, the module of such gradient
    - While optional, if locality is considered, it will be taken into account for the evaluation
  - `SobelCompletoParallel`: parallelization of `SobelCompleto`

Task 2: sobel2.cpp

- The main program must:
  - Call functions `SobelBasico`, `SobelParallel`, `SobelLocal` and `SobelLocalParallel`, check they are correct, and show the result
  - Call functions `SobelCompleto` and `SobelCompletoParallel`, check they match and show the result
  - Print the running times of each function
  - Also draw your conclusions for the time elapse in each case, and **include them as comments in the beginning** of the program.

Uploads

- Task 1: during the lab session
- Task 2: in two week time
  - After the lab session
- We remind you that:
  - You can work in pairs
  - You just need to upload the **.cpp** source code, including as a comment in the first line the full name of the author/s