Chapter 1 – Image Filtering and Gradients

Author: Gianmarco Scarano

gianmarcoscarano@gmail.com

1. Images

An image is a 3D tensor of numbers (for R-G-B images), composed by pixels in a range from [0,255]. We use 8 bits to represent the 256 intensity values, meaning that a specific color has a specific intensity.

So basically, we can think of an image as a function f from $R^2 to\ R$ where $f(x, y)$ gives the intensity at a certain position x and y.

- Quantization = How wide is the distance between two pixels through a delta-value. We say that our samples in the 2D space are *delta* distant.

Of course, for a RGB image, we have a vector of functions, all over the R-G-B channels:

$$f(x, y) = \begin{bmatrix} r(x, y) \\ g(x, y) \\ b(x, y) \end{bmatrix}$$

- Image filtering = Colors of the image are altered without changing pixel positions (Range-pixel change).
- Image warping = Points are mapped to other points without changing the colors (Domain-pixel change).

Image filtering itself, is divided in:

- Point operation = Take a pixel and apply a transformation on top (also called Point Processing)
- Neighborhood operation = We take a patch of pixels and apply a transformation on top (also called Filtering). Filtering is useful when we want to enhance an image (remove noise, sharpen, details) or when we want to extract edges or contours.

2. Linear Filtering

The main operations in Linear Filtering are correlation and convolution. We want to use this sort of “filter” (also called “kernel” or “mask”) in order to have a modified image.

Linear filtering operators involve fixed weighted combinations of pixels in small neighborhoods.

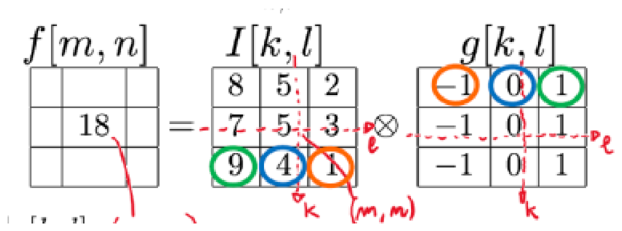
The most famous one is the Box Filter which replaces the pixels with a local average, applying some kind of smoothing (blur) effect. This filter is shift-invariant, meaning that as the pixel shifts, so does the kernel.

$$\mathrm{g(i,j)} = \sum f(i + k, j + l) \cdot h(k, l)$$

When we deal with correlation, instead, we are talking about (for example) 2 signals and compare their differences, analyzing how similar they are. Convolution has also some nicer properties with respect to correlation (such as commutative, associative, distributive, etc).

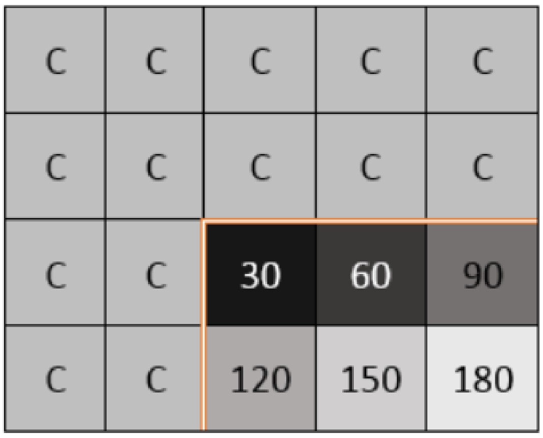
When we are dealing with 2D signals in a 2D convolution, we want to

- Mirror the filter (K and l)
- Swipe it across the image
- Multiply and sum

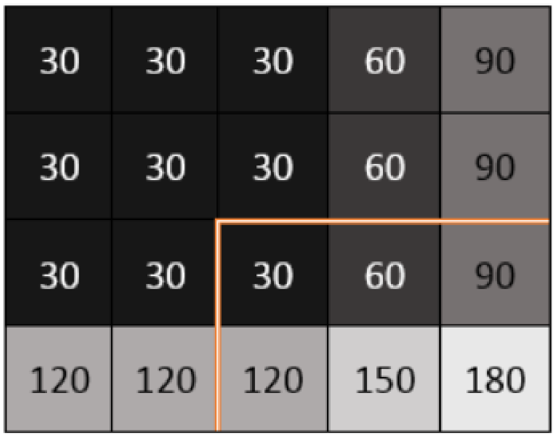


2.1 Padding

So, what happens when we reach the edge? The filter window falls off the edge of the image, that's why we have methods such as padding in order to:

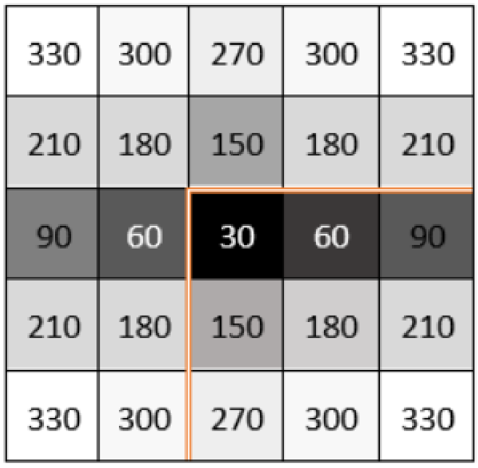
- Zero padding:
    - Set all pixels outside the source image to 0.
- Constant:
    - Set all pixels outside the source image to a specified border value
    - 

C	C	C	C	C
C	C	C	C	C
C	C	30	60	90
C	C	120	150	180

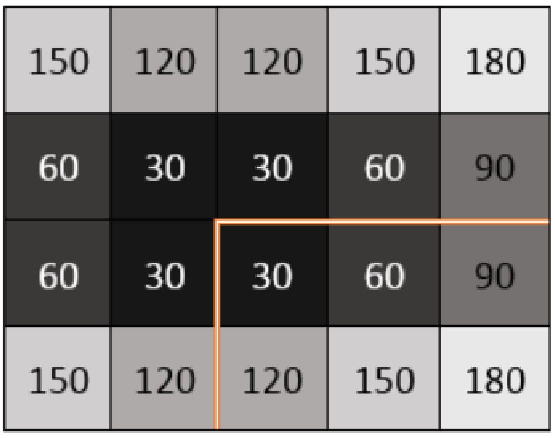
- Clamp / Replicate:
    - The pixel values at the edge of the active frame are repeated to make rows and columns of padding pixels.
    - 

30	30	30	60	90
30	30	30	60	90
30	30	30	60	90
120	120	120	150	180

- Reflection / Mirror:
    - In mirror padding, when padding an image, the pixels at the image border are mirrored or reflected inwards to fill the additional padding cells.



330	300	270	300	330
210	180	150	180	210
90	60	30	60	90
210	180	150	180	210
330	300	270	300	330

- Symmetric:
    - The padding pixels are added such that they mirror the edge of the image.
    - 

150	120	120	150	180
60	30	30	60	90
60	30	30	60	90
150	120	120	150	180

2.2 Separable filters

A 2D filter is separable if it can be written as the product of a "column" and a "row".

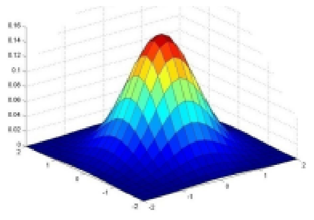
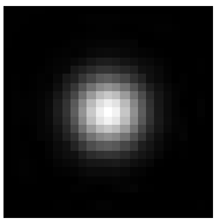
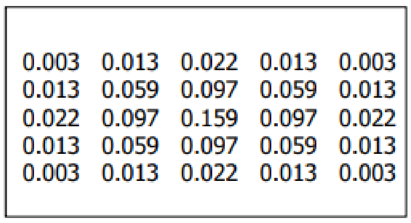
example: box filter

$$\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} = \underset{\text{column}}{\begin{array}{|c|}\hline 1 \\ \hline 1 \\ \hline 1 \\ \hline \end{array}} * \underset{\text{row}}{\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline \end{array}}$$

We separate a kernel because we could significantly speed up the operations by first performing a one-dimensional horizontal convolution followed by a one-dimensional vertical convolution, which requires a total of 2·K operations per pixel instead of K^2.

The most famous filters are:

- Gaussian filter: This filter samples the 2D Gaussian Function and it's theoretically infinite, but in practice truncated to some maximum distance. We can control the blurriness of the filter (width of Gaussian blur) through the sigma (σ) value. It is also a separable filter.

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

5 x 5, $\sigma = 1$

- Smoothing
- Sharpening
- Thresholding: $g(m,n) = \begin{cases} 255 & if\ f(m,n) > A \\ 0 & otherwise \end{cases}$ where A is a pre-defined threshold.
- Median filter: We take the medium of the patch and we replace the 255 value with the average. This fixes the salt and pepper noise problem.

3. Image Gradients

3.1 Edges

Edges are very sharp discontinuities in intensity. In CV, edges are important for:

1. Most semantic and shape information can be deduced from them, so we can perform object recognition and analyze perspectives and geometry of an image
2. They are a more compact representation than pixels.

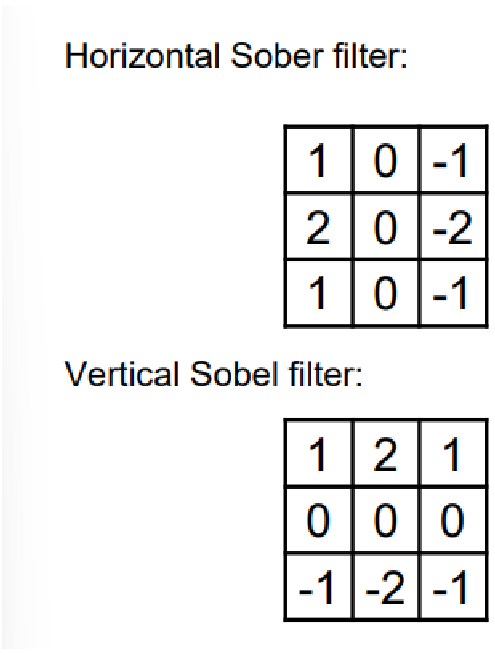
Wherever there is a rapid change in the intensity function, it indicates an edge.

In order to detect edges in an image, we take derivatives, since they provide a large discontinuity.

3.2 Sobel filter

It's basically a filter which defines the edges in an image.

$$\underset{\text{Sobel filter}}{\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}} = \underset{\text{Blurring}}{\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}} * \underset{\text{1D derivative filter}}{\begin{bmatrix} 1 & 0 & -1 \end{bmatrix}}$$

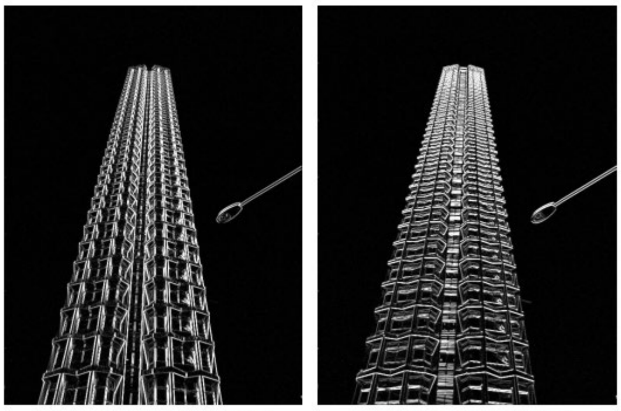


Horizontal Sober filter:

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

Vertical Sobel filter:

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



horizontal Sobel filter vertical Sobel filter

A reasonable approach is to define an edge as a location of rapid intensity or color variation. Think of an image as a height field. On such a surface, edges occur at locations of steep slopes, or equivalently, in regions of closely packed contour lines (on a topographic map). A mathematical way to define the slope and direction of a surface is through its gradient:

$$J(x) = \nabla I(x) = \begin{pmatrix} \frac{\partial I}{\partial x} & \frac{\partial I}{\partial y} \end{pmatrix} \cdot (x)$$

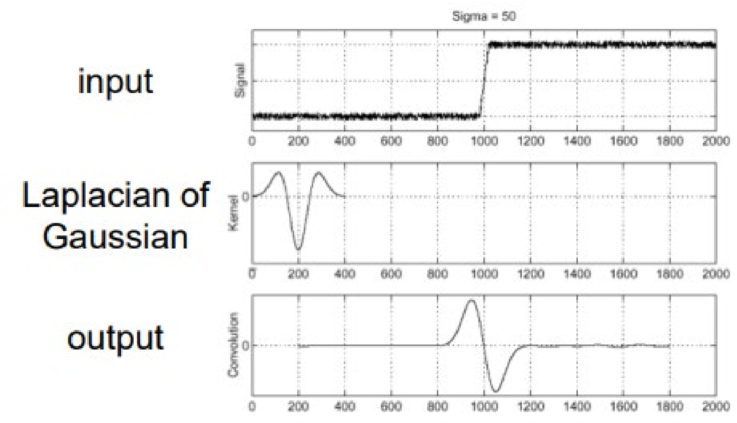
The local gradient vector J points in the direction of steepest ascent in the intensity function.

The pixels with the largest gradient values in the direction of the gradient become edge pixels. However, plotting the pixel intensities of the gradient often results in noise, making it almost impossible to identify where an edge is by only taking the first derivative of the function.

When we work with 2D signals, we might have a noisy input and computing the derivative results in a very noise signal.

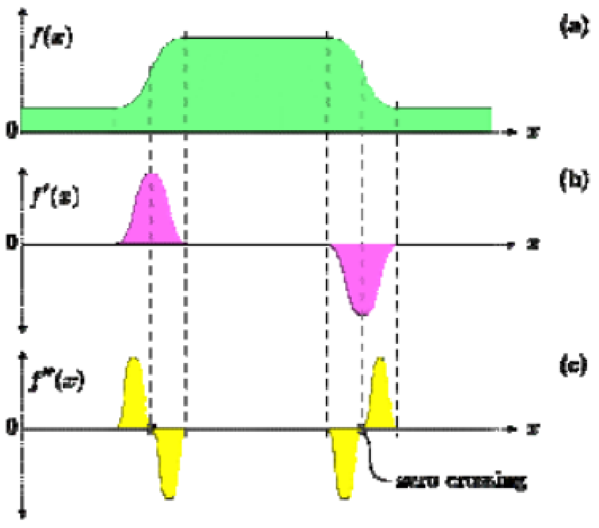
In fact, we first apply a Gaussian filter in order to smooth the signal and then apply the derivative. We could've also applied directly the derivative of a Gaussian on top of our input (nice property).

A Laplace Filter is some sort of second derivative, which can be combined with Gaussian Filtering, producing the zero-crossing phenomenon at edges, as below:



- First derivative at edge is a maximum
- Second derivative at edge is zero

- It is more easy to understand when a function is zero than find a maximum



3.3 Canny edge detector

We are now ready to design an edge detector, which must be accurate, trying to minimize the number of false positives and false negatives as long as having precise localization, pointing at edges where they actually occur and avoid too many responses.

The most common detector is the Canny edge detector, which detects edges by:

1. Applies x and y derivatives of the Gaussian filter in order to improve noise and localization, having one single response as an optimal edge detector would do
2. Finds the orientation of the gradient at each single pixel in the image
3. Performs non-maximum suppression, where we thin the edges down to a single pixel in terms of width. (Simply we check if the pixel is a local maximum along the gradient direction -> Largest gradient magnitude along the gradient direction).
4. Connects the edges by determining the weak and strong ones through a simple threshold (respectively for weak and strong edges).