Spatial Filtering II

1 Convolution

Definition: Mask

A matrix of coefficients defining a linear filter for processing an image
Also called: filter, kernel, template window

The size of the mask is that of the local neighbourhood of the filter

A linear filter operates on the image linearly, i.e. component-wise multiplication and summation

The spatial linear filtering of a matrix A by the mask M is called the convolution of A by M

Intuitively, the mechanics of convolution work as follows

For each pixel of the input image I_{input}

- Place the centre of the mast M over the pixel $I_{input}(i, j)$
- Do the component-wise multiplication of corresponding elements of the mask and the neighbourhood of $I_{input}(i, j)$

The matrix should be updated only after all responses have been computed

If the mask has odd dimensions it has a well-defined centre. Otherwise, we have to arbitrarily designate an element of the mask as its centre

For a mask of size $(2N + 1) \times (2N + 1)$ convolution can be written:

$$I_{output}(i, j) = \sum_{k=1}^{2N+1} \sum_{l=1}^{N+1} I_{input}(i - N - 1 + k, j - N - 1 + l) m_{kl}$$

Or more conveniently

$$I_{\text{output}}(i, j) = \sum_{k=-N}^{N} \sum_{l=-N}^{N} I_{\text{input}}(i + k, j + l) m_{kl}$$

1.1 Example

Find the convolution of A by M

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad M = \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

2 Gaussian filters

The value of the element p′ of the mask is given by the Gaussian function g of the distance, $d = |p - p'|$ between the centre of the mask and the element p′

We construct the 3×3 Gaussian filter for $\sigma = 0.5$ by dividing the matrix

$$\begin{bmatrix} g_\sigma(\sqrt{2}) & g_\sigma(1) & g_\sigma(\sqrt{2}) \\ g_\sigma(1) & g_\sigma(0) & g_\sigma(1) \\ g_\sigma(\sqrt{2}) & g_\sigma(1) & g_\sigma(\sqrt{2}) \end{bmatrix}$$

by the sum of its elements, getting

$$M_3 = \begin{bmatrix} 0.011 & 0.083 & 0.011 \\ 0.083 & 0.619 & 0.083 \\ 0.011 & 0.083 & 0.011 \end{bmatrix}$$

In image processing we often require the elements of a mask to sum up to 1, in which case convolution does not change the average intensity of the image

The gaussian filter gives more weight to the centre of the mask and, gradually, less weight to the values away from the centre

The Gaussian function is positive everywhere

In principle a Gaussian mask would be infinitely large

We can define the size of the mask, essentially trim the very small weights, so that we do not waste time on computations that will not affect the final result

The elements of the Gaussian masks are coming from the values of a Gaussian function

That allows a description of Gaussian filtering without explicit reference to masks

$$I_p^{\text{output}} = \frac{\sum_{p' \in \Omega} g\left(\left|p - p'\right|\right) I_{p'}}{\sum_{p' \in \Omega} g\left(\left|p - p'\right|\right)}$$

The new intensity I_p^{output} of the pixel p is the weighted sum of the intensities of the pixels p' in a neighbourhood Ω

The denominator of the fraction normalises the expression, making the sum of the weights 1

Gaussian filters are used in practice to remove noise from an image, but often removes the fine features too

Performs blurring/smoothing of the image

Increasing the standard deviation σ makes it more blurred

3 Solution to the blurring

3.1 Non-local means

Main idea:

In a larger neighbourhood, perhaps even the whole image, search for neighbourhoods similar to the one pixel being processed

Replace the current pixel with a weighted mean of pixels with similar neighbourhoods

Weights computed according to similarity between neighbourhoods

The averaging spans pixels that are not local within the image to the target pixel - hence the name

Output pixel = weighted sum of other pixels (all, or a subset of them) weighted a similarity measure of the corresponding neighbourhoods.

$$I_{output}(i,j) = \frac{\sum W_{neighbuburnood}(k,l) I_{input}(k,l)}{\sum W_{neighbournood}(k,l)}$$

A commonly used similarity measure is obtained by applying an exponential function on the squared distance between neighbourhoods (seen as N^2 vectors)

Performance:

- Good on Gaussian noise removal
- Dependent on neighbourhood size $N \times N$
- Slow, but several optimisations exist
- Easily parallelised

4 Laplacian filtering

In continuous mathematics, the Laplacian of a function f(x,y) is defined as a sum of partial derivatives

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

In image processing, thinking of the image as a function $I(x, y)$ from pixel coordinates to intensities, the Laplacian is commonly implemented via either of the discrete convolution filters

$$\begin{matrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{matrix} \quad \text{or} \quad \begin{matrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{matrix}$$

Notice that both mass have some negative elements and they sum up to 0

When we discrete a continuous function, derivatives and the second order derivatives become second order differences (i.e. differences of differences)

Consider a 1-dimensional image with three pixels with intensities a,b,c

$$[a\ b\ c]$$

The intensity differences are

$$[b-a, c-b]$$

The difference of the intensity differences is

$$(c-b) = (b-a) = c - 2b + a$$

Which is the response at the central pixel b in the convolution of the image

$$[a\ b\ c]$$

By the mask

$$[1\ -2\ 1]$$

In 2D images, the differences of the intensity differences in the x and y directions are given my the responses to the masks

$$\begin{matrix} 0 & 0 & 0 \\ 1 & -2 & 1 \\ 0 & 0 & 0 \end{matrix} \quad \text{and} \quad \begin{matrix} 0 & 1 & 0 \\ 0 & -2 & 0 \\ 0 & 1 & 0 \end{matrix}$$

respectively, and by adding them we get the Laplacian filter

$$\begin{matrix} 0 & 0 & 0 \\ 1 & -2 & 1 \\ 0 & 0 & 0 \end{matrix} + \begin{matrix} 0 & 1 & 0 \\ 0 & -2 & 0 \\ 0 & 1 & 0 \end{matrix} = \begin{matrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{matrix}$$

The response to the Laplacian filter is zero at the areas of the image where the intensity changes smoothly

Indeed, when intensities change smoothly, intensity differences are equal, and the second order differences are zero

When the variation of the intensities is not smooth, the responses to the Laplacian filter can be non-zero

The less smooth the variation of the intensities, the higher in absolute value in the response of the Laplacian filter.

Thus, the highest in absolute value responses are at the edges of the image

Laplacian filter operation: highlights areas of rapid intensity change i.e., the edges of the image

Laplacian filter is sensitive to noise. Both edges and noise were amplified

Solution: smooth the image with a Gaussian kernel to remove noise and compute the Laplacian of the smoothed image (known as Laplacian of Gaussian (LoG))

5 General side note

Image boundaries - what do we do here?

- Extrapolate values from inside the image (one of the best options)
- Or, use padding with zeros or any other value (often visually poor results)
- Or, do not perform convolution in these regions (the resulting convolved image will be smaller than the original)