Advanced Image Processing - Edge Detection

Ing. Viktor Kocur
viktor.kocur@fmph.uniba.sk

DAI FMFI UK

30.10.2019

The Process

Finding edges

In case of a continuous functions the edges are found using derivatives. Some methods use the first derivatives while others use the second ones.

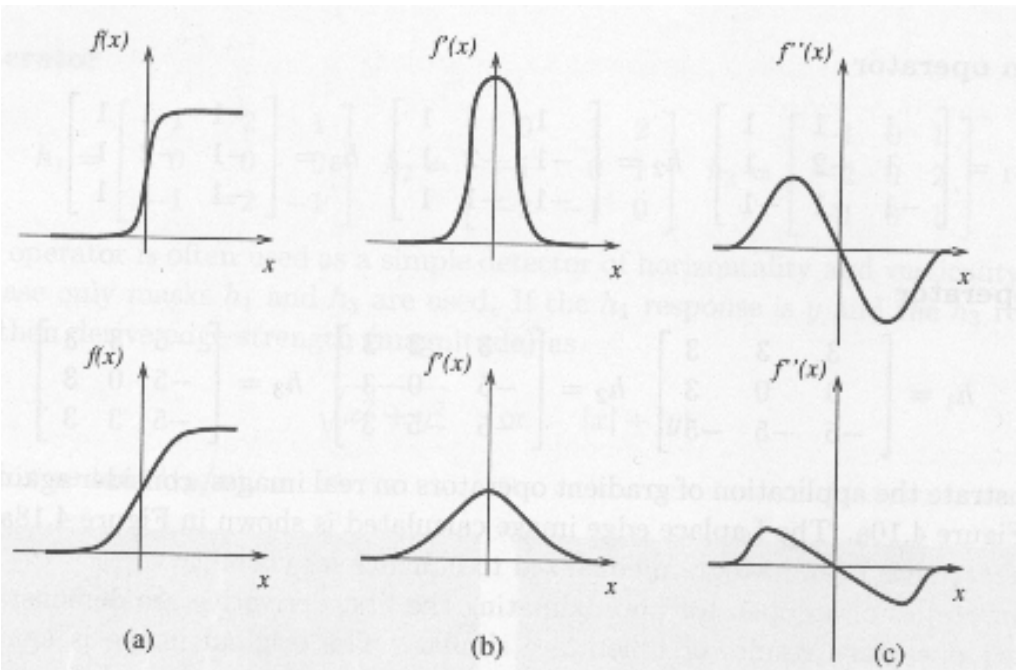


Figure 4.20: *1D edge profile of the zero-crossing.*

2D discrete case

Discrete case

Image is discrete therefore we will utilize differences instead of derivatives. In other words the discrete version of the derivative.

2D

The image is 2-dimensional we therefore use partial differences in different directions.

Using the first derivative

Convolution

We compute the difference with convolution operation in a similar manner to smoothing.

Exercise

Use conv2 and thr Prewitt filters to find the edges in the image zatisie.jpg Do not forget to use rgb2gray. Since we have two filters use $h = \sqrt{h_x^2 + h_y^2}$ to obtain combined edges. After the filtration use thresholding to show the edges.

Prewitt filters

$$\begin{matrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{matrix} \quad a \quad \begin{matrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{matrix}$$

Matlab - edge

edge

edge(I, method) - returns edge image based on the method. First derivative methods are 'Sobel', 'Prewitt', 'Roberts' and 'Canny'. Second derivative methods are 'log' also known as Marr-Hildreth method.

edge

edge(I, method, threshold, direction) - It is also possible to specify threshold to use and the direction of the filter.

Exercise

Exercise

Test the edge detection algotihms based on the first derivative.

Noise

Edge detection can fail with noisy image. Add noise to the image and try to detect edges in the image. Try to perform edge detection after smoothing. Does the result improve?

Canny detector

Gauss smoothing

Canny detector first smooths the image using the Gaussian filter.

Non-maximum suppression

After smoothing a different detector using the first derivative is used. Since simple methods create edges too wide. In each area only the strongest edge is kept. This takes the direction of the edge into consideration.

Weak and strong edge

In the end two thresholds are used to divide the remaining edge pixels to two categories: strong and weak edges. Strong edges are kept. From the weak edges only the ones connected to the strong edges are kept.

Second derivative method

Second derivative

We can find the edge in positions where the second derivative changes sign (zero-crossing).

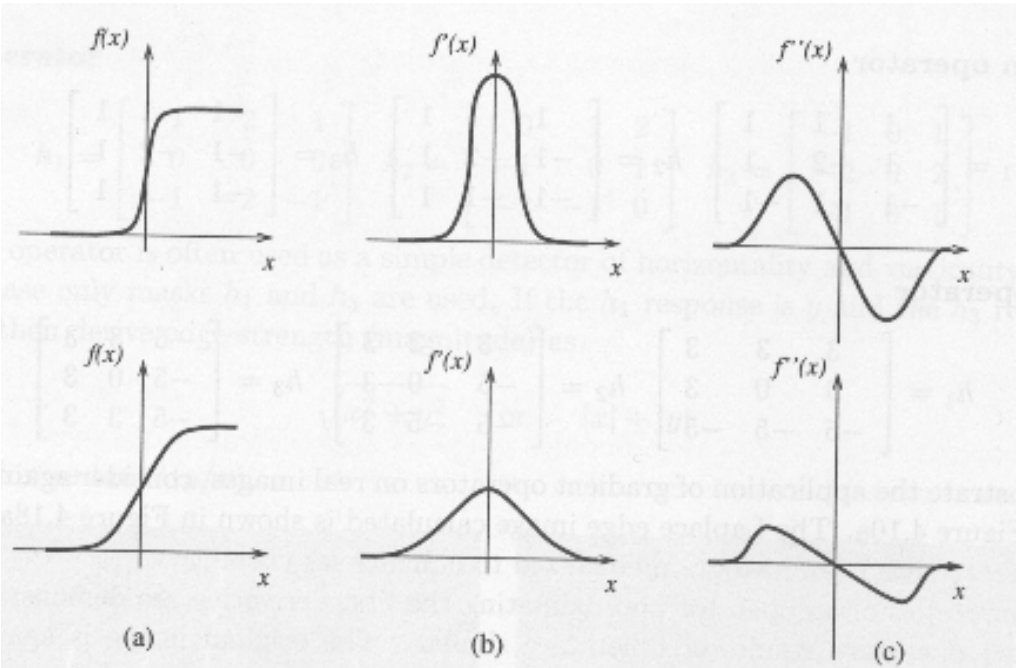


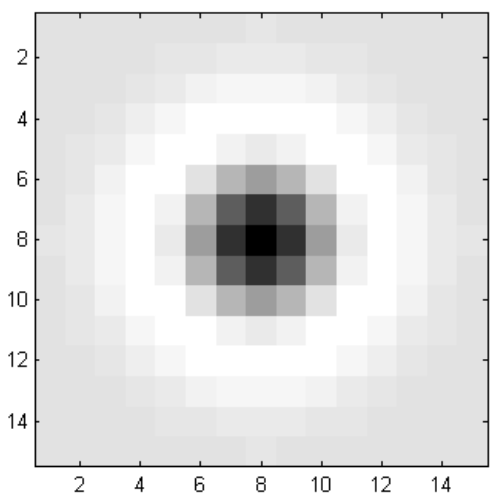
Figure 4.20: *1D edge profile of the zero-crossing.*

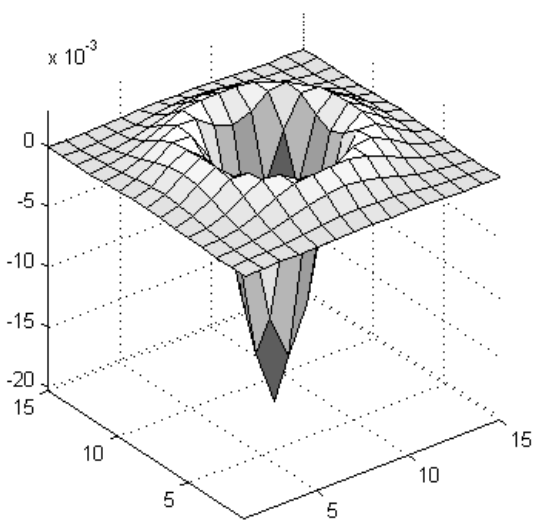
Marr-Hildreth method

LoG

To obtain the second derivative the Laplacian of Gaussian filter is used.

$$LoG = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}}$$





Marr-Hildreth method

Zero-Crossings

After application of the filter the zero-crossings have to be found.

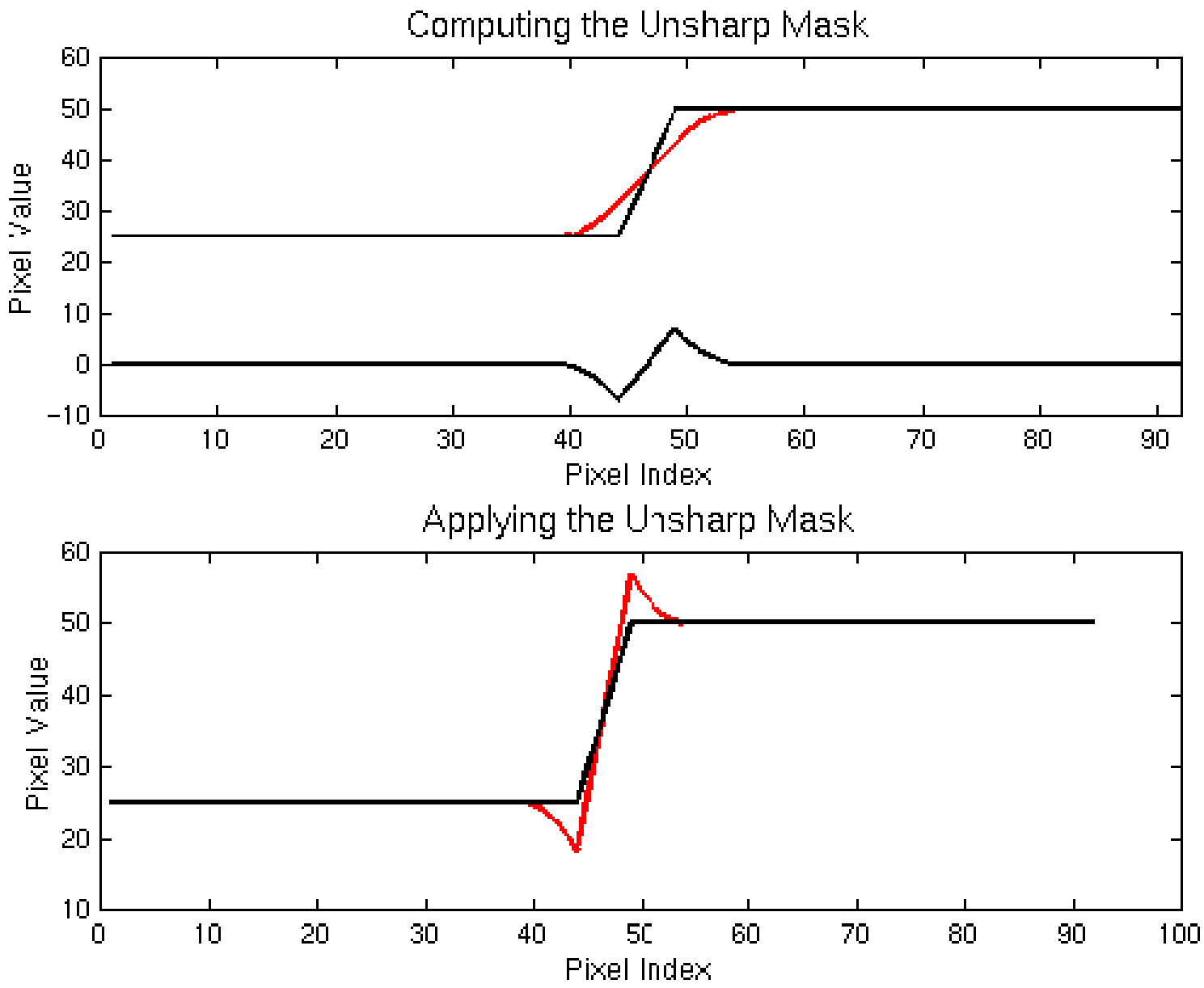
Faster method

It is possible to use difference of Gaussians (DoG) instead of LoG.

Exercise

Use the Marr-Hildreth method denoted as 'log' to detect edges.

Unsharp masking



Unsharp masking

Sharpening

We want to sharpen a blurred image. This task is similar to amplifying the edges.

Unsharp masking - princíp

$$I_{sharp} = I_{original} + p \cdot (I_{original} - I_{smooth})$$

Úloha

Create a function unsharp_mask(I,p,sigma), which applies unsharp masking with the parameter p with the use of addiive Gaussian noise with the parameter sigma. Apply this on the image blurred.pgm

Laplace operator

Laplace operator - definition

$$\Delta f = \nabla \cdot \nabla f = \sum_{i=1}^{n} \frac{\partial^2 f}{\partial x_i^2} \overset{2D}{=} \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Convolution kernels in 2D

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \textit{alebo} \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Laplace operator in matlab

We can generate manually or by using fspecial('laplacian',alpha), where alpha determines how strong is the presence of the diagonal neighbors

Image sharpening using the Laplace operator

Application

$$I_{ostrý} = I_{originál} - p\,(L_{jadro} * I_{originál})$$

Exercise

Load the image blurred.pgm and use Laplace sharpening on it. Use different values of p. Do not forget about data types.