Detecting contours of human organs in CT images using the Marr-Hildreth edge detector

Assignment 2.e

Biomedical Signal and Image Processing 2020/21, Faculty of Computer and Information Science, University of Ljubljana

Urša Zrimšek

***Abstract*—This is a report of the exam assignment for the course Biomedical Signal and Image Processing. It describes a *Marr-Hildreth* edge detection algorithm, methods it uses and graphically shows each step and the final result on examples from CTMRI database.**

I. Introduction

Edge detection is a technique that looks for boundaries inside an image. It is a useful step in many algorithms for segmentation and extraction. It works by finding the places in which the pixel intensity changes the most. In this article we will describe *Marr-Hildreth* edge detection algorithm more thoroughly described in [1], and at the end add a further step called *edge linking.* For testing and for analysis of the intermediate steps we used CT and MRI images of the Computed Tomography-Magnetic Resonance Imaging Database (CTMRI DB).

II. Methodology

This edge detection system is composed of four stages: *image smoothing, detection of edge points, edge localization* and of *edge linking.*

A. Image smoothing

First we need to preprocess our image for noise reduction. We perform image smoothing, by applying *Gaussian blur* - a low pass filter that will reduce the image's high frequency components.

We do that by convolving the image with a Gaussian filter, defined by Gaussian function,

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}.$$

To get the $n \times n$ squared matrix, with which we will convolve the image, we need to center the indexing, $i, j = -\frac{n-1}{2}, -\frac{n-1}{2} + 1, \ldots, \frac{n-1}{2}$ and calculate the matrix elements as $G(i,j)$. The matrix is then normalized so that the sum of the elements in it is equal to 1.

In our case we want the size of the filter and σ to be selected automatically in such a way that it will be appropriate for the size of the input image, and that the size of the filter will be appropriate for the selection of σ - for larger σ (wider Gaussian function) we need to increase the size of the filter. We do that by setting σ to 0.5% of the smaller size of the image and the matrix size n to the ceiling of 6 times σ.

On figure 1 we can see an example of an image smoothed in such a way.

B. Detection of edge points

The next step of detection again uses convolution. Here we use a *Laplacian filter*, that represents a discrete aproximation of Laplacian function of an image I,

$$L(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2},$$

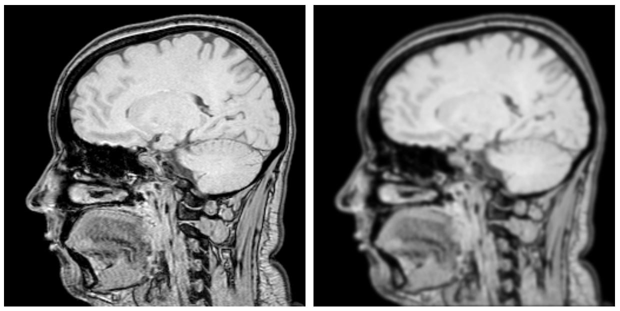

Figure 1. Original image (left) and smoothed image (right), smoothed with Gaussian filter with (automatically selected) $n = 9$ and $\sigma = 1.28$.

that highlights the regions of rapid change in pixel intensity, that could be the candidates for edges.

In our implementation we used a discrete approximation by convolving the smoothed image with filter

$$L = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}.$$

On figure 2 we can see an example of the result of convolution with a Laplacian filter over a smoothed image.

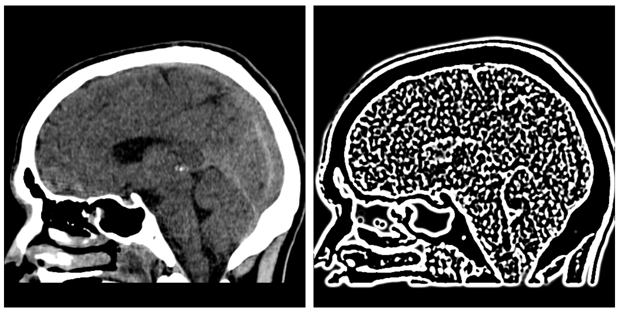

Figure 2. Original image (left) and Laplacian of smoothed image (right).

C. Edge localization

Until this point, we were talking about techniques that a lot of other edge detectors use. *Zero crossings*, the technique for edge localisation that we will describe in this sub section, is unique for Marr-Hildreth edge detector.

It is an approach that takes the output of the previous step and for each pixel decides if it is a valid candidate for an edge point, by looking at it's neighbourhood. We look at a 3×3 neighbourhood centered at the pixel and look at all the opposing neighbours - over both diagonals, up-down and left-right pairs. If at least two of the pairs have different signs, the observed pixel is selected as an edge pixel.

Here we can add an additional condition, that besides having opposing signs, the difference between the neighbouring pixels must exceed a threshold T for them to contribute to the two pairs needed. The threshold is one of the parameters that needs to be given to the detector - if you don't want to use it, set it to 0. You can see a comparison of edge localization with or without threshold on figure 3

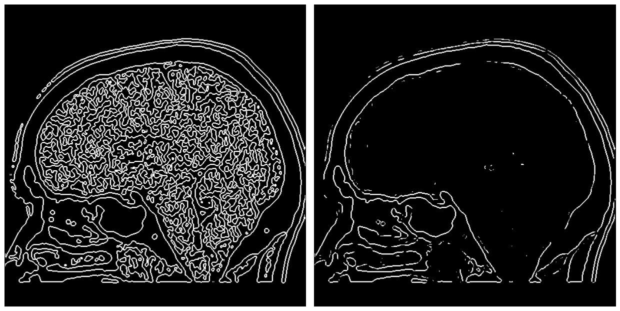
Figure 3. Edge localization on an image seen in figure 2 without the use of threshold (left) and with the use of threshold $T = 0.015$ (right).

D. Edge linking

In edge linking we are editing the binarized image from the output of edge localization, that has ones where edge is located, otherwise zeros. We go over rows and columns (separately) and fill in (set them to 1) all the gaps between edges, that are not longer than K. K is a parameter that also needs to be given to the detector. If you don't want to use it, you should again set it to 0.

At edge linking you should be careful not to set K to high. That will result in a picture as seen on the right side of figure 4.

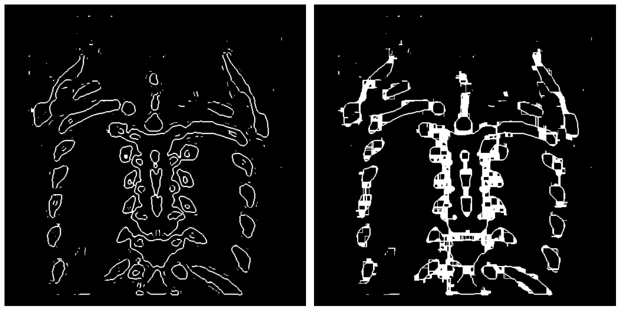
Figure 4. Edge linking with $K = 2$ (left) and with $K = 10$ (right).

III. Results

In this section we will look at how our algorithm performs on some of the examples from CTMRI DB.

On figure 5 we can see that our algorithm can detect all the contours of the organs visible on the original image. But to do so, we need to lower the threshold which results in also detecting some noise contained in the image.

On figure 6 we can see how the choice of threshold affects edge detection. For better visibility we didn't do any edge linking, that would (even with small K) result in unrecognizable edges in top right image that is obtained without thresholding.

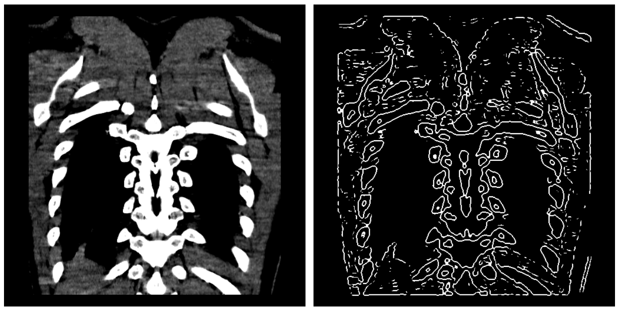
Figure 5. Original image (left) and output of our algorithm using parameters $threshold = 0.007$ and $K = 2$ (right).

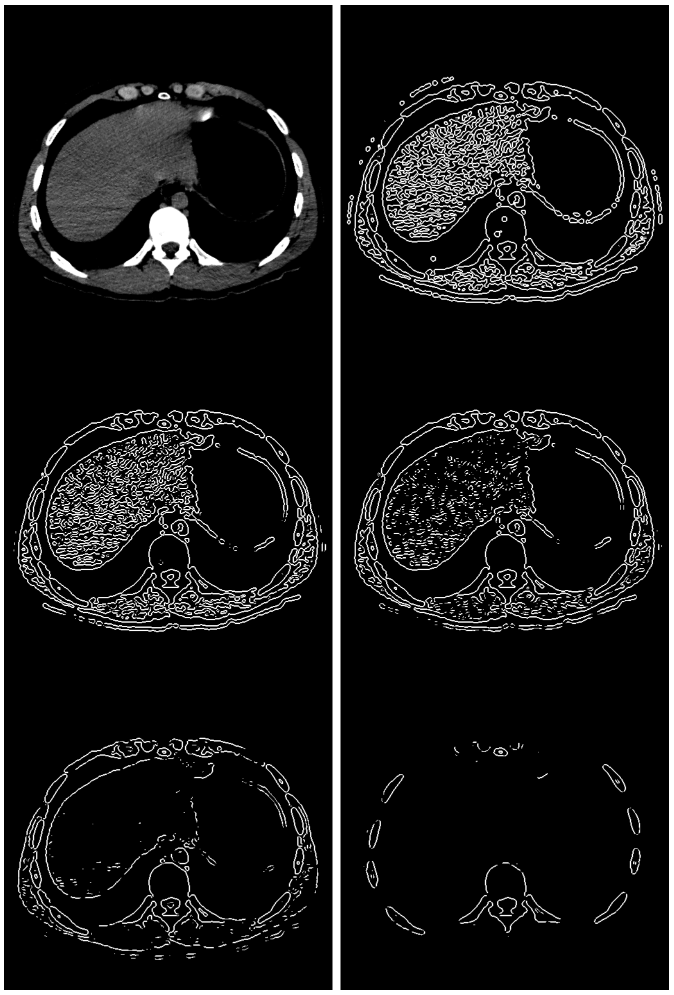
Figure 6. Original image (first) and output of our algorithm using different thresholds $threshold \in \{0, 0.002, 0.004, 0.008, 0.02\}$, without edge linking.

IV. Discussion

When applying the implemented detector to different images we can see, that to get only the important edges, different thresholds need to be used. Sometimes we must try many different values, to be able to obtain the result we want. This is not very useful if we want to implement an edge detector that will serve for further automated diagnoses or disease detection. We propose that the first improvement of this algorithm should be an automated selection of the optimal threshold and also of the edge linking parameter.

We also think that edge linking in such way more often makes the detection worse as it makes it better. To improve it, we could implement a way to link the edges in the direction where they are pointing instead of looking for gaps only in the row/column direction.

References

[1] D. Marr and E. Hildreth, "Theory of edge detection," *Proceedings of the Royal Society of London. Series B. Biological Sciences*, vol. 207, no. 1167, pp. 187–217, 1980.