Detecting contours of human organs in CT images using the Canny edge detector

Klemen Škrlj

January 24, 2022

University of Ljubljana
Faculty of Computer and Information Science
e-mail: ks5379@student.uni-lj.si

Abstract

We describe and implement Canny edge detector, which is widely used in medical imaging. We go through the ideas step by step and explain the theory behind it. Instead of traditional manual setting of thresholds for hysteresis, we use Otsu's thresholding algorithm. We compare the results on images from CTMRI DB. In the end we discuss the performance and suggest some further improvements.
Key words: medical images, edge detector, Gauss filter, image derivatives, hysteresis thresholding.

1 Introduction

Computed tomography is commonly referred to as a CT scan. A CT scan is a diagnostic imaging procedure that uses a combination of X-rays and computer technology to produce images of the inside of the body. It shows detailed images of any part of the body, including the bones, muscles, fat, organs and blood vessels. It is noninvasive and very good for diagnostic purposes.

Depending on a use case, we have to normally post process original CT scan to make it more clear and suitable for diagnosis. One of the ways we can do this is by detecting main edges and removing noise. This can be then used to locate tumors or to reconstruct organs in 3D by connecting edges on neighbouring slices. We tackle this problem by implementing Canny edge detector[1].

2 Methods

In general, a good edge detector has to minimize number of false positive detections, detected edges have to be close to their real location and detector has to only produce one point per edge pixel. Edge is by intuition a discontinuity in gray-scale image which can be detected by derivative. To implement full Canny edge detector we follow 4 main steps:

1. Image smoothing
2. Gradient and angle calculation
3. Non-maxima suppression
4. Hysteresis thresholding

We describe each one of them in detail in following sections.

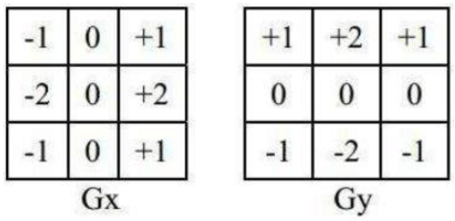
2.1 Image smoothing

Edge detection algorithms are highly susceptible to noise. This noise is normally high frequency so we need a low pass filter. Typically we convolve original image with Gaussian filter seen on Equation 1. Parameter σ determines the extent of smoothing where bigger value means greater effect. Since this is proportional to image size we set it to 0.5% of smaller image size. The whole kernel size is then calculated by formula: $kernelSize = ceil(6 \cdot \sigma)$.

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

2.2 Gradient and angle calculation

We then have to detect sharp changes in the gray-scale image which are candidates for edge pixels. For this we calculate first derivatives in x and y direction. This is done by convolving image with 3x3 Sobel filter, which is defined as seen in Figure 1. This kernel is more robust to noise than a simple 2x1 since it takes more neighbourhood into the account, but still prioritizes immediate neighbours more (where the weight is ± 2).



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

Gx

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

Gy

Figure 1: Sobel operator

Magnitude image is then computer by formula 2 where G_x is derivative image in x direction while G_y is derivative image in y direction. Angle image is computed by equation 3.

$$M(x,y) = \sqrt{G_x^2 + G_y^2} \tag{2}$$

$$A(x,y) = arctan(G_y/G_x) \tag{3}$$

On Figure 2 we can see an example of magnitude and angle images after our preprocessing.

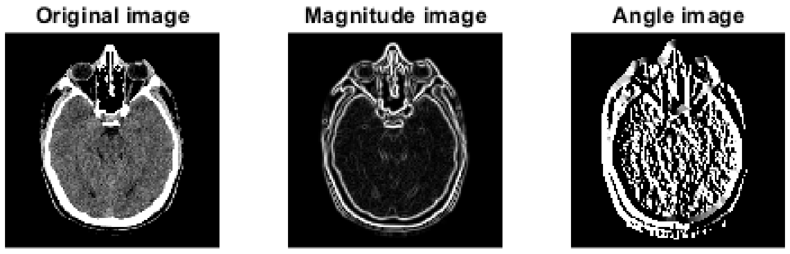


Figure 2: Magnitude and angle image after preprocessing

2.3 Non-maxima suppression

On of the requirements of a good edge detector is producing one pixel wide edges. This is why we perform non-maxima suppression on the magnitude image. The algorithm goes through all potential edges and first classifies them into one of four classes depending on the edge angle as seen on Figure 3.

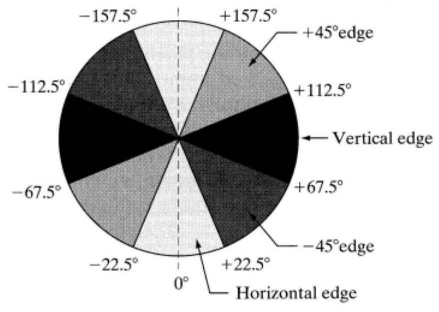


Figure 3: Edge classification depending on angle [2]

Reason for this classification is to look in the direction of current point and evaluate if this is a maximum or not. In other words, we compare magnitude of one point before and one point after the current one and set it to 0, if any of magnitudes is higher then current. Else we keep the value as it is. The output of this algorithm is an image with only one pixel wide lines.

2.4 Hysteresis thresholding

In the last step of Canny edge detector, we have to binarize the image. If we employ a simple, single value threshold, we are left with an image that has either a lot of noise or edges that are not connected well together. To combat this a double thresholding is used. First all pixels that are over high threshold are found. We can be sure that these represent an edge. Then for every one of this points we look in their immediate neighbourhood (neighbouring 8 pixels) and find those that have values higher than low threshold. We classifiy them as an edge pixel as well. We repeat this process on these new edge pixels until no more are detected. Effectively we are following the lines to connect edges together.

A problem that comes from this algorithm is choosing initial lower and higher threshold values. This are hyper-parameters which depend on the current image and static values across different images would not work as well. This is why we use Otsu algorithm[3] to get high threshold. The algorithm works by choosing a threshold which splits the image histogram into two parts: background and foreground. Based on this, weight and variance of each part is calculated. Finally, within class variance is calculated by summing up both variances multiplied by their weights. The algorithm goes through all possible threshold values and returns the one that minimizes within class variance, which is used as a high threshold. Lower threshold is calculated by $T_{low} = T_{high} \cdot 0.5$. This method allows automatic hyper-parameter setup for each image separately.

3 Results

We tested our algorithm on images from CTMRI DB[4]. We first compared results when using static threshold versus when we use Otsu's thresholding as seen on Figure 4.

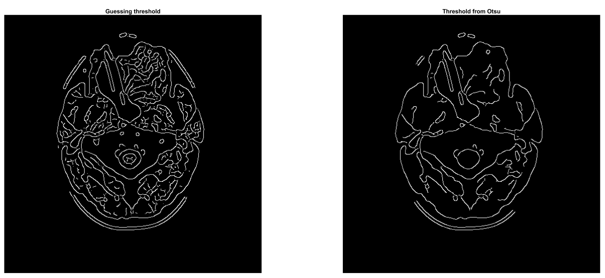


Figure 4: Comparison between guessing threshold and Otsu threshold

Figures 5, 6 and 7 show few examples of how our algorithm with improved thresholding by Otsu works on different images from the database. In first column is an original, in second one is image after non-maxima suppression and in last is final binarized image with edges.

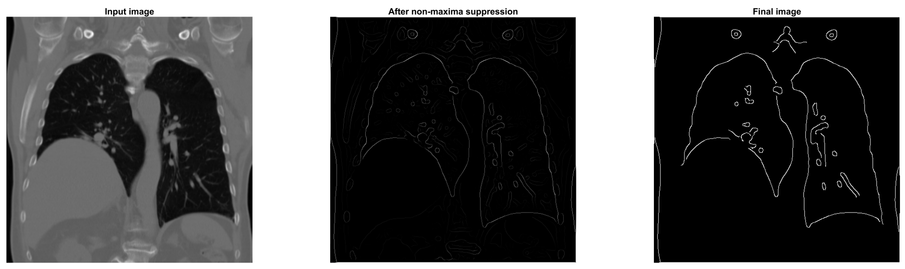


Figure 5: Input image, image after non-maxima suppression and final image

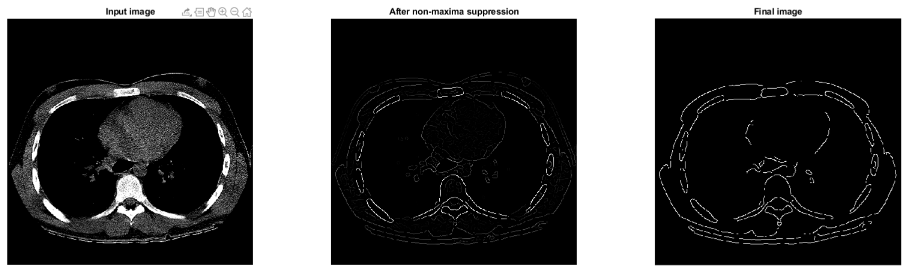


Figure 6: Input image, image after non-maxima suppression and final image

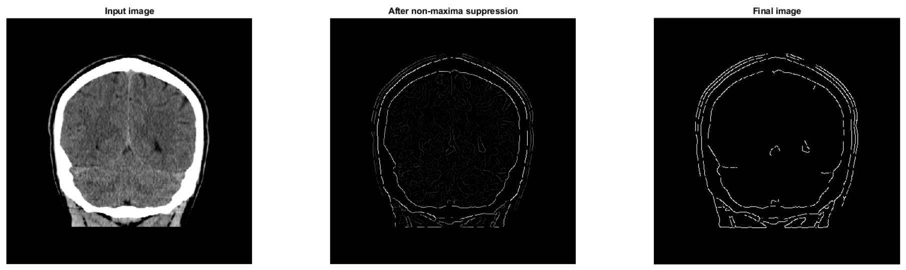


Figure 7: Input image, image after non-maxima suppression and final image

Figure 8 shows original input image with our detected edges in red as an overlay.

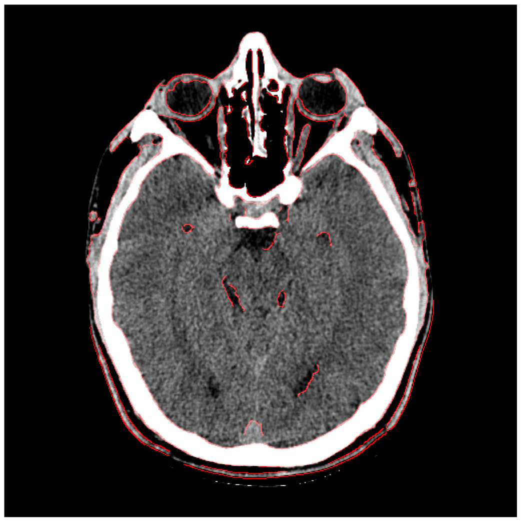


Figure 8: Detected edges overlaid on the original image

4 Discussion

As we can see, our change to thresholding by using Otsu's method for defining high threshold works much better. On the image were we try to guess the threshold we have a lot of noise still present. We would have to do lots of experiments to get the right balance between noise and not missing important edges. Instead Otsu's algorithm does this automatically.

Other images show quite good results. The major edges in all of them are detected well, but the algorithm sometimes struggles with too strict thresholding. This is for example seen on Figure 6 where a gray, round structure in the middle is not seen in final image as good.

Maybe the best way to present the final image is like we did in Figure 8, where overlay with detected edges indicates some important areas, but we still have the full information of an original photo infront of us.

There is still some room for improvements, mainly in the first step where we smooth the image. Here Gaussian filter is prone to smoothing not only noise but some weak edges as well. To combat this we could use an adaptive median filter which is also good for noise removal and leaves edges intact.

5 References

References

[1] John Canny. “A Computational Approach to Edge Detection”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-8.6 (1986), pp. 679–698. DOI: `10.1109/TPAMI.1986.4767851`.

[2] Jager Franc. *EDGE DETECTION AND SEGMENTATION OF IMAGES, I.* URL: `https://ucilnica.fri.uni-lj.si/course/view.php?id=151/`.

[3] Nobuyuki Otsu. “A Threshold Selection Method from Gray-Level Histograms”. In: *IEEE Transactions on Systems, Man, and Cybernetics* 9.1 (1979), pp. 62–66. DOI: `10.1109/TSMC.1979.4310076`.

[4] Alessandro Taddei et al. *CT-MRI database.* URL: `https://lbcsi.fri.uni-lj.si/OBSS/Data/CTMRI/`.