Analysis of computed tomography (CT) images using Canny edge detector

Aljaz Mur Erzen[1]

[1] *ae4664@student.uni-lj.si, 63160011*

Introduction

Automatic analysis of computed tomography (CT) images plays a key role in assistance to medial experts in finding abnormalities and predicting possible diseases. As the first step in the pipeline, edge detection heavily contributes to reliability and performance of the whole analysis system.

This report describes internal mechanisms of Canny edge detection and our implementation for detection on CT images.

Blurring and derivation

An edge can be thought of as a sharp transition from one segment of similar color to another. If our image would be a smooth function of x and y without much noise, we could compute directional derivatives and simply threshold them to find areas considered as an edge.

Directional derivative is related to rate of change of a function in the given direction. Or more geometrically: the angle quotient of a tangent line to our function in given point and direction.

Fortunately, there is no need for derivatives in *all* directions, but only the two that are parallel to the x-axis and y-axis. They are called partial derivatives. One or both of them have high absolute value, we know that there is high rate of change in *some* direction.

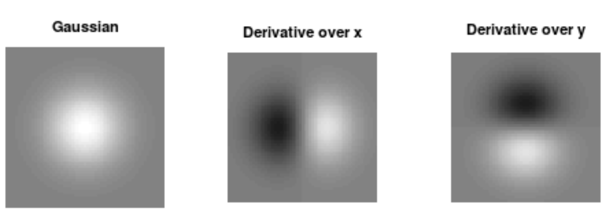

Figure 1. Gaussian and its derivatives over x and y

Unfortunately, input images are not continuous and contain noise which prevents us from easily computing derivatives. Instead we could compute differentials, which are just subtraction between adjacent pixels. This would not solve the problems with noise so instead of subtracting just two adjacent pixels, we apply convolution with a derivative of gaussian (DoG). This takes into account a region whose size is determined by parameter σ.

Note that DoG can be decomposed into convolution of one dimensional gaussian with one dimensional derivative of gaussian. This allows us to apply two smaller convolutions instead of one large, improving computational complexity:

$$\begin{aligned} \frac{\partial I}{\partial x} &\approx I_x = I * \frac{\partial G}{\partial x} = I * (\frac{\partial g}{\partial t}^T * g) = (I * \frac{\partial g}{\partial t}^T) * g \\ \frac{\partial I}{\partial y} &\approx I_y = I * \frac{\partial G}{\partial x} = I * (\frac{\partial g}{\partial t} * g^T) = (I * \frac{\partial g}{\partial t}) * g^T \end{aligned} \tag{1}$$

For the subsequent steps, will need only the magnitude (2-norm of the partial derivatives) and their directions:

$$\begin{aligned} I_{mag}(x,y) &= | \begin{bmatrix} I_x(x,y) \\ I_y(x,y) \end{bmatrix} |_2 = \sqrt{I_x^2(x,y) + I_y^2(x,y)} \\ I_{dir}(x,y) &= \arctan(I_y(x,y)/I_x(x,y)) \end{aligned} \tag{2}$$

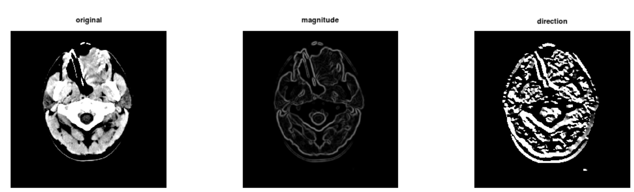

Figure 2. Derivation of CT scan

Non-maximum suppression

Non-maximum suppression is a method of line thinning which takes derivative magnitude image and removes (sets to zero) all pixels that are not local maximums in direction of the gradient (vector of partial derivatives).

For our implementation, we have simplified and discretized the gradient direction into 4 angles as:

$$\alpha = round((I_{dir} + \pi \mod \pi)/\pi * 4) \mod 4 \tag{3}$$

Figure 4 shows the result of such function for every possible gradient direction.

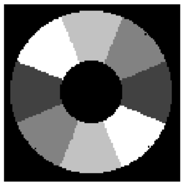

Figure 3. Discretization of direction to 4 classes

Now we can simply compare the magnitude value to the neighboring pixels in one of the 4 discretized angles.

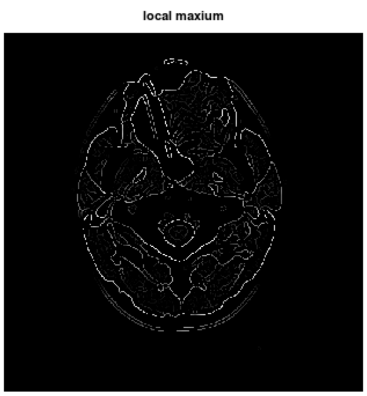

Figure 4. Non-maximum suppression

Hysteresis threshold

Simply applying a threshold to the magnitude image produces an image with may disconnected edges. To prevent this, we apply double threshold. All pixels whose magnitude is higher as the high threshold t_{high} are marked as strong edges. All pixels that are not strong but whose magnitude is higher that the low threshold t_{low} are marked as weak edges. The output contains all strong edges and those weak edges that are connected to some strong edge via 8 connectivity rule.

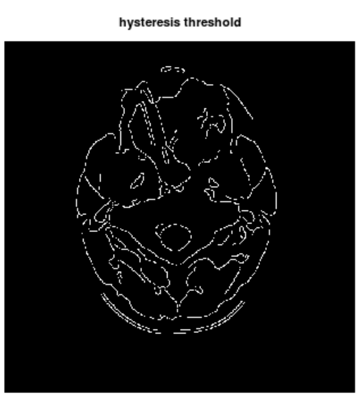

Figure 5. Hysteresis threshold

This is considered the output of the Canny detector. But because the results of our implementation were very segmented, we decided to add another step.

Edge linking

The edges are now very disconnected, which is why we apply two simple morphological operations: dilation and erosion. Both of them take binary image as an input and output.

Dilation is a sliding-widow operator, where result of each window evaluation is computed as or between all input pixels. Effectivity, all positive areas (our edges) in input get expanded or dilated, hence the name.

Erosion is similarly a sliding-widow operator, but the result of each window evaluation is computed as *and* between all input pixels. Effectivity, all positive areas (our edges) in input get shrunk or eroded.

If we apply dilation and subsequent erosion, small gaps between edges may get filled during dilation but would not get eroded, thus connecting the gaps. The downside is that some smaller "holes" will also get filled-in, but for our application, this does represent a problem.

Results