Report on Assignment 2

Matej Bevec
Email: matejbevec98@gmail.com

***Abstract*—When analysing computed tomography (CT) image, edge detection is often the first step in the pipeline. In this assignment, we implement a common edge detection algorithm, the Canny detector, and test it on the task of human organ contour detection. We briefly describe the implemented method and present qualitative results. Based on the observed results, we conclude that the basic Canny detector is effective at the given task, although it exhibits certain tradeoffs.**

I. Introduction

In biomedical image analysis, as in many other computer vision applications, edge detection is an important building block in many pipelines. It is the shared first step in many different CV problems and can be seen as a low-level information reduction/compression procedure that facilitates higher-level downstream tasks, such as object detection or segmentation. An edge detection method must detect real object edges in an image (i.e. rapid changes in intensity), while ignoring artifacts such as noise.

In this exercise, we implement one of the most widely used edge detection algorithms — the Canny edge detector — and apply it to the task of human organ contour detection in computed tomography (CT) images.

II. Method

The algorithm takes a grayscale image as input and produces a binary image, where the detected edges are 8-point-connected one-pixel-wide white lines. Ideally, all such lines should represent real edges in the original image, while no lines should correspond to noise, artifacts or non-edge features.

The detection pipeline consists of the following 4 steps:

1) **Gaussian blur.** First, the image is blurred with a Gaussian kernel to reduce high-frequency noise and other details in the image, since such intensity changes could be detected as edges in the later steps. The size of the Gaussian kernel, and its standard deviation (σ) are two of the user-defined parameters.
2) **Computing gradient.** Next, the first derivative of the image in x direction (G_x) and y direction (G_y) are computed. They represent local intensity changes in their respective direction and are computed by convolving the image with a Sobel filter. Then, the gradient magnitude is computed using the pythagorean theorem ($M = \sqrt{G_x^2 + G_y^2}$), along with the gradient angle ($A = arctan(\frac{G_y}{G_x})$. The magnitude relates to an edges "strength" (i.e. contrast), while the angle represents the direction perpendicular to the edge (i.e. the direction of intensity change).
3) **Non-maxima supression.** Given the gradient, we wish to find exact one-line-wide edges (i.e. the "centers" of wider edges). This is achieved by applying non-maxima supression. For every pixel, its neighbors in the direction of the gradient are checked. The pixel is only kept it is higher in value than both of the neighbors, otherwise it is set to 0.
4) **Thresholding and edge linking.** Next, the image is *thresholded* to remove remaining pixels which likely do not represent edges. Two thresholds are used. The pixels higher in magnitude than the *higher threshold* are considered "strong" edges, meaning they likely correspond with real edges. The pixels higher in magnitude than the *lower threshold* are considered "weak" edges, meaning they may or may not correspond to real edges.
   In the final step, weak and strong edges are combined via edge linking using the hysteresis method. Following the intuition that real weak edges are likely continuations of strong edges, the weak pixels are marked strong only if they are 8-point-connected to an existing strong pixel. The resulting image representing strong edges is the final output.

We implement the algorithm in Python following the described steps. The Gaussian blur and Sobel filters are implemented with the OpenCV Python library, while other components are manually implemented using Numpy and native Python features. Figure 1 depicts the intermediate images at different stages.

III. Results

We test our implementation on 4 CT images. In figures 2, 2 and 4 we explore the performance of the algorithm at different parameters. The Sobel filter size is kept fixed at 3 (pixels) while we vary Gaussian filter size, standard deviation, lower threshold and higher threshold. Note that since we normalize images to an intensity range from 0 to 1.

In Figure 2, low thresholds are used. It can be observed that most real organs' edges are fully detected. However, the detections also include the edges of smaller substructures as well as noise within homogenous regions (present in all images except the third).

In Figure 3, high thresholds are used. In this case, noise detections are mostly not present (the expection is the fourth image), but there are some discontinuations within real edges.

In Figure 3, low thresholds are combined with a larger Gaussian filter. Most noise seems to be eliminated, while keeping real edges fully connected.

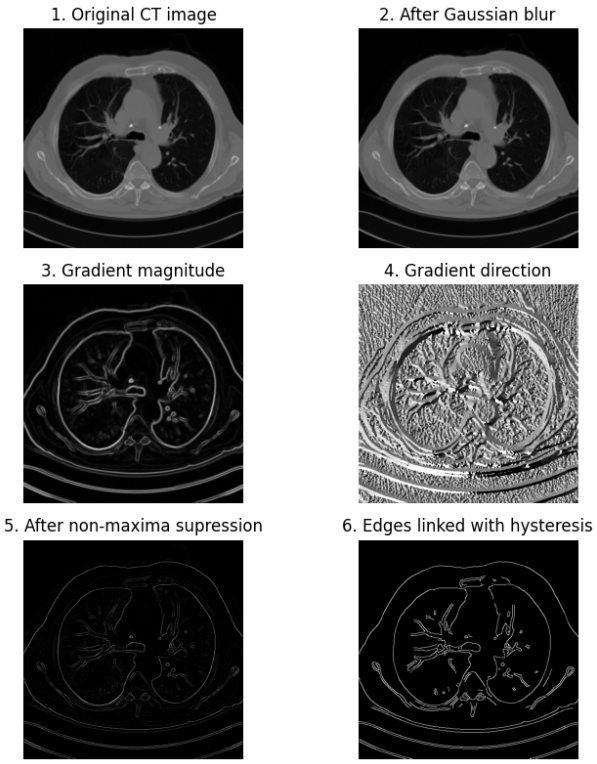

Fig. 1. The intermediate images after the described processing steps.

IV. Discussion

The results expose what appears to be a fundamental tradeoff when configuring the Canny detector. Low thresholds will result in detecting real edges in full, but will include false noise detections. Conversely, high thresholds will avoid false detections, but may result in discontinuities in real edges. Ultimately, this bias needs to be determined in relation to the downstream task.

Figure 4 suggests that using a large Gaussing filter (a "stronger" blur) in combination with low thresholds, might provide the best overall solution, so far as small or fine-detailed structures don't need to be accurately detected.

Additionally, it is evident that even within the used CT database, there is significant variance between images. This means some images (e.g. the third image) may see better results with lower thresholds, while for others (e.g. the fourth image), larger thresholds may be optimal. As a result, using a single fixed set of parameters for the entire dataset may not produce desired results. Dynamically adapted parameters might provide an improvement.

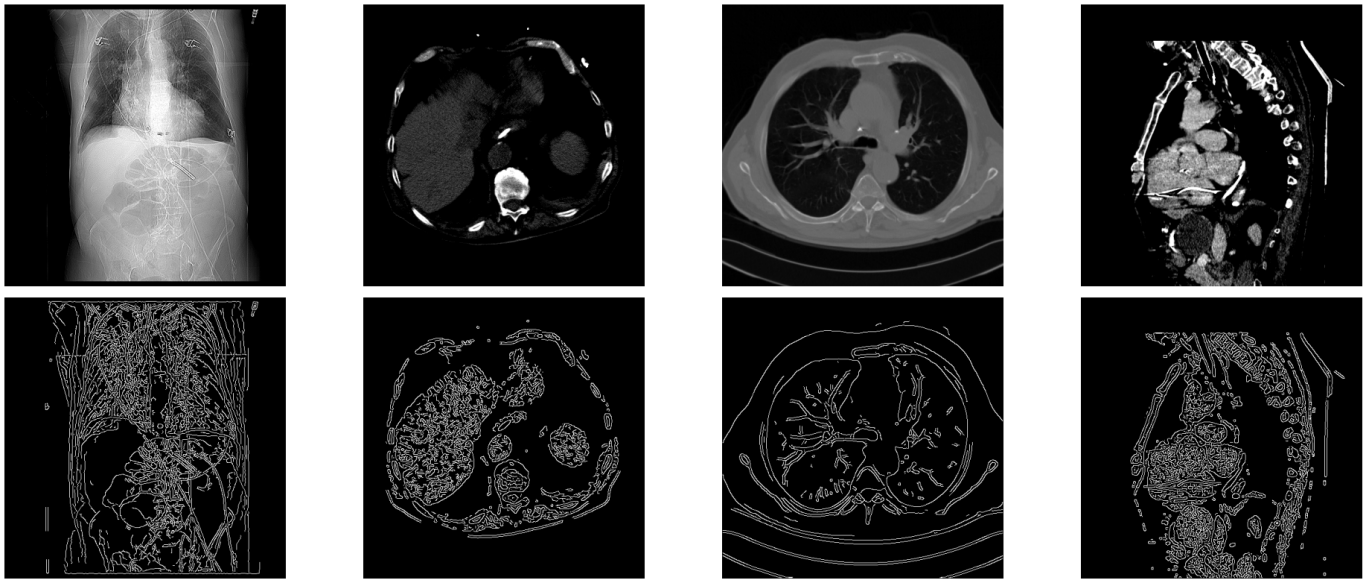

Fig. 2. Detected edges using a 5x5 Gaussian blur window with a standard deviation of 1.5, a lower (weak) threshold of 0.04 and a higher (strong) threshold of 0.12. The top row depicts original CT scan images. The bottom row depicts the Canny detector output, after edge linking.

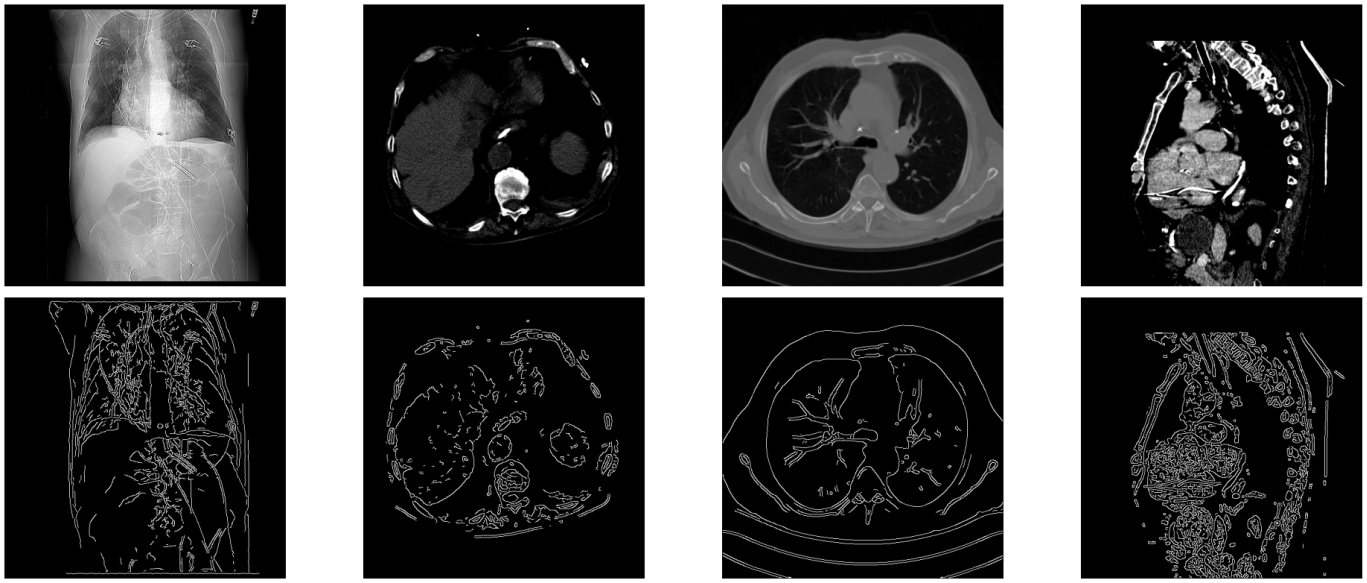

Fig. 3. Detected edges using a 5x5 Gaussian blur window with a standard deviation of 1.5, a lower (weak) threshold of 0.1 and a higher (strong) threshold of 0.2. The top row depicts original CT scan images. The bottom row depicts the Canny detector output, after edge linking.

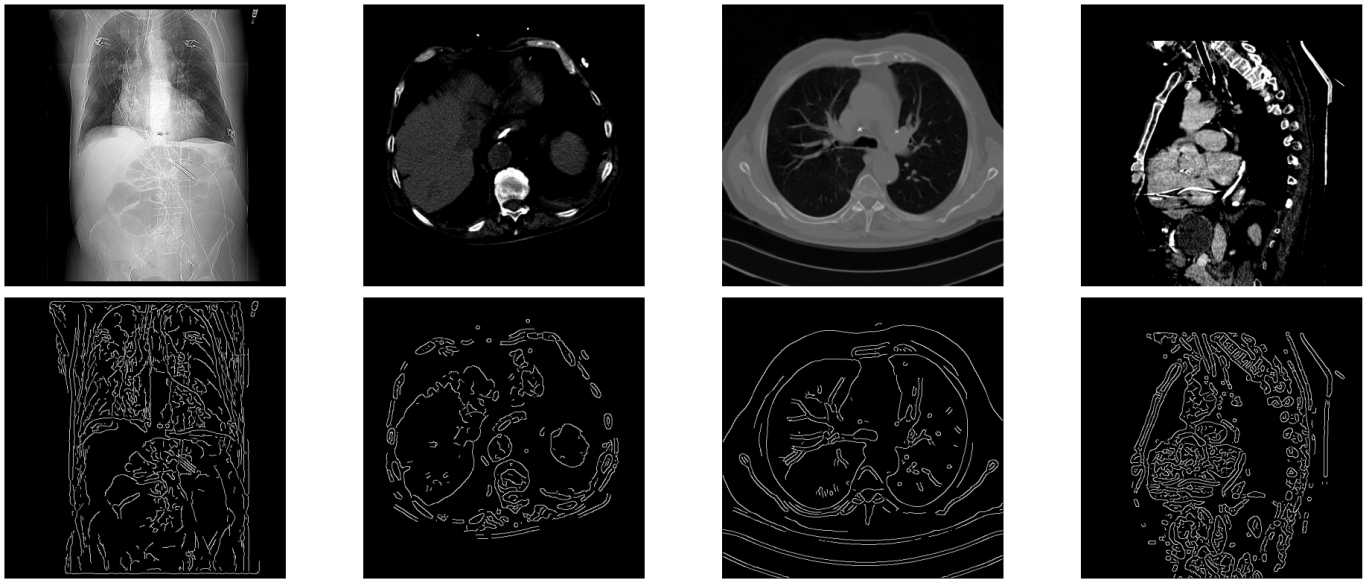

Fig. 4. Detected edges using a 9x9 Gaussian blur window with a standard deviation of 2.5, a lower (weak) threshold of 0.04 and a higher (strong) threshold of 0.08. The top row depicts original CT scan images. The bottom row depicts the Canny detector output, after edge linking.