VISION & IMAGE PROCESSING

ASSIGNMENT 1

Date: 6 December 2017

Student: Cristian Mitroi

Introduction

I have organized my report around the four assignment exercises, each of them being devoted a section.

In each section there will be one or more images or figures. I will reference them by name throughout the text.

The goal of the assignment was to become familiar with edge detection methods in image processing. I have mostly used the already implemented methods from the `OpenCV` library (`cv2`) in the Python programming language. I decided to do this because, in the short time alloted, I thought it would be more important to focus on understanding how the different parameters for the functions affect the result. Had I more time I would have also been keen on implementing some of the methods on my own.

Definitions

Before we delve into the experiments themselves I want to provide some definitions for the various terms in the report:

Gaussian function Is used to describe the probability of a "normally distributed random variable" [6]. The graph of the function is marked by a distinctive "bell curve" form.

Convolution Is a mathematical operation on two separate functions. It produces a third function. [7]

Kernel Is a small matrix used for particular operations in the image processing field. E.g. blurring, sharpening, edge detection etc. [8] "The height and width can differ, but they both must be positive and odd" [9].

Sigma (σ) Also called "standard deviation". It is used to measure the amount of spread of a value. In our case, it controls the amount of "spread" of a kernel matrix.

Exercise 1: Gaussian filtering

See figure "gaussian"

The Gaussian blur function in the `cv2` library implements the Gaussian blur effect. It is mathematically based on applying the Gaussian function (by a *convolution*) on the image matrix.

The `sigmaX` and `sigmaY` parameters correspond, mathematically, to the *standard deviation*.

Notice that in calling the `GaussianBlur` method I provide a kernel size of `(0,0)`. In this situation the kernel size will be chosen based on the provided σ value.

We can see that using higher sigma values yields more blurred images. On the right side we can see that the difference between the original and the respective blurred image is commensurate to the the sigma values.

Gaussian blur is used to "reduce image noise and reduce detail" [1]. It smoothes the difference in values between the pixels.

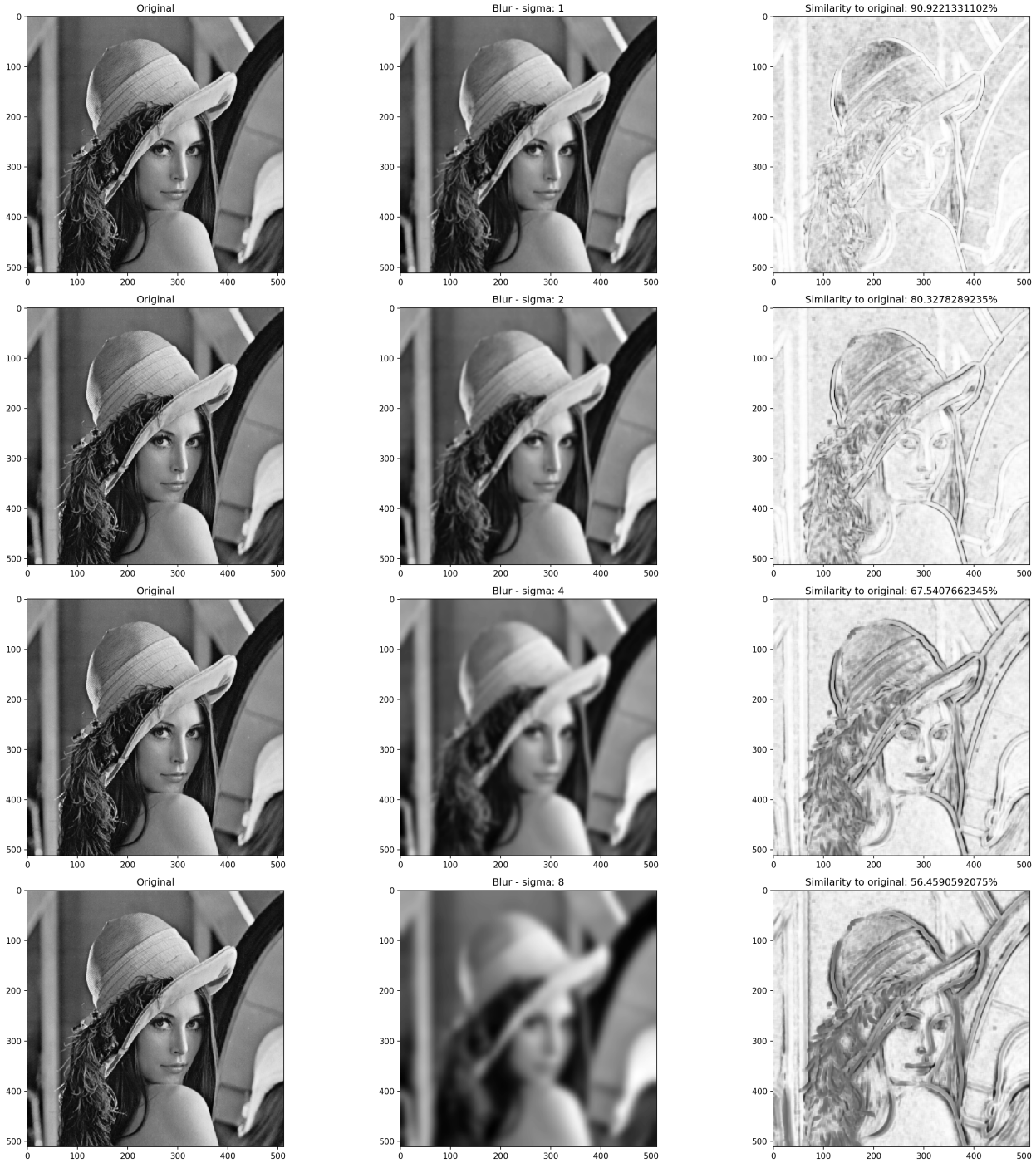

Figure 1: gaussian

Gaussian "removes 'high-frequency' components from the image" [2]. It is a "low pass" filter. Larger σ (standard deviation) remove more details.

Exercise 2: Gradient magnitude

The `Sobel` method in OpenCV is an implementation of gradient magnitude (edge) detection, based on the first order derivative of the Gaussian function. In this case, in mathematical terms, an "edge" will be a "jump in intensity" [3]. According to the OpenCV documentation, it "computes an approximation of the gradient of an image intensity function" [3]. It also "combines Gaussian smoothing and differentiation." [3]

The method itself takes as parameters:

- the image
- depth
- direction (as two parameters). whether it's the `x` or the `y` axis
- kernel size.

Using `-1` as the second parameter tells the `Sobel` method to use the same datatype for the result image as the input image (`float64`).

The Sobel method operates on a given direction (see `direction` param. above). Thus we need to call it twice, once for X and once for Y. Then we need to apply Pythagora's to get an image that includes both the X and Y values.

We also need to apply a Gaussian blur to eliminate some of the noise. In the following we try different σ values for the Gaussian blur. We use the default kernel size by not specifying any specific size to our `get_gradient` function.

See figure "sobel"

We apply a Gaussian blur in order to reduce noise of the image. We do this using different values of σ. We notice, just like in the first section, that higher values yield a more blurred image. We eliminate noise and information at the same time. This also creates thicker edges in the Sobel result.

Exercise 3: Laplacian-Gaussian filtering

The Laplace operator uses the second derivative in order to detect edges. In the first derivative edges are marked by a "maximum" (as used by the Sobel operative), but in the second derivatives they are marked by 0 (zero)

We also need to apply a Gaussian blur to eliminate some of the noise. In the following we try different σ values for the Gaussian blur. We use 5 as the kernel size for the `Laplace` method.

See figure "laplace"

Again, we notice that increasing the σ value of the blur function results in "thicker" edges, just like in results from applying the Sobel method.

We can see that Laplace detects more detailed edges. For example, there are edges detected in Lenna's hair. These are not shown in the Sobel results.

We can also see that the best results are the ones in the second row, where we only apply some light blurring ($\sigma = 1$).

Research has shown that Laplace is more rigurous in detecing edges than Sobel, as it detects "sharp and precise edges" [4].

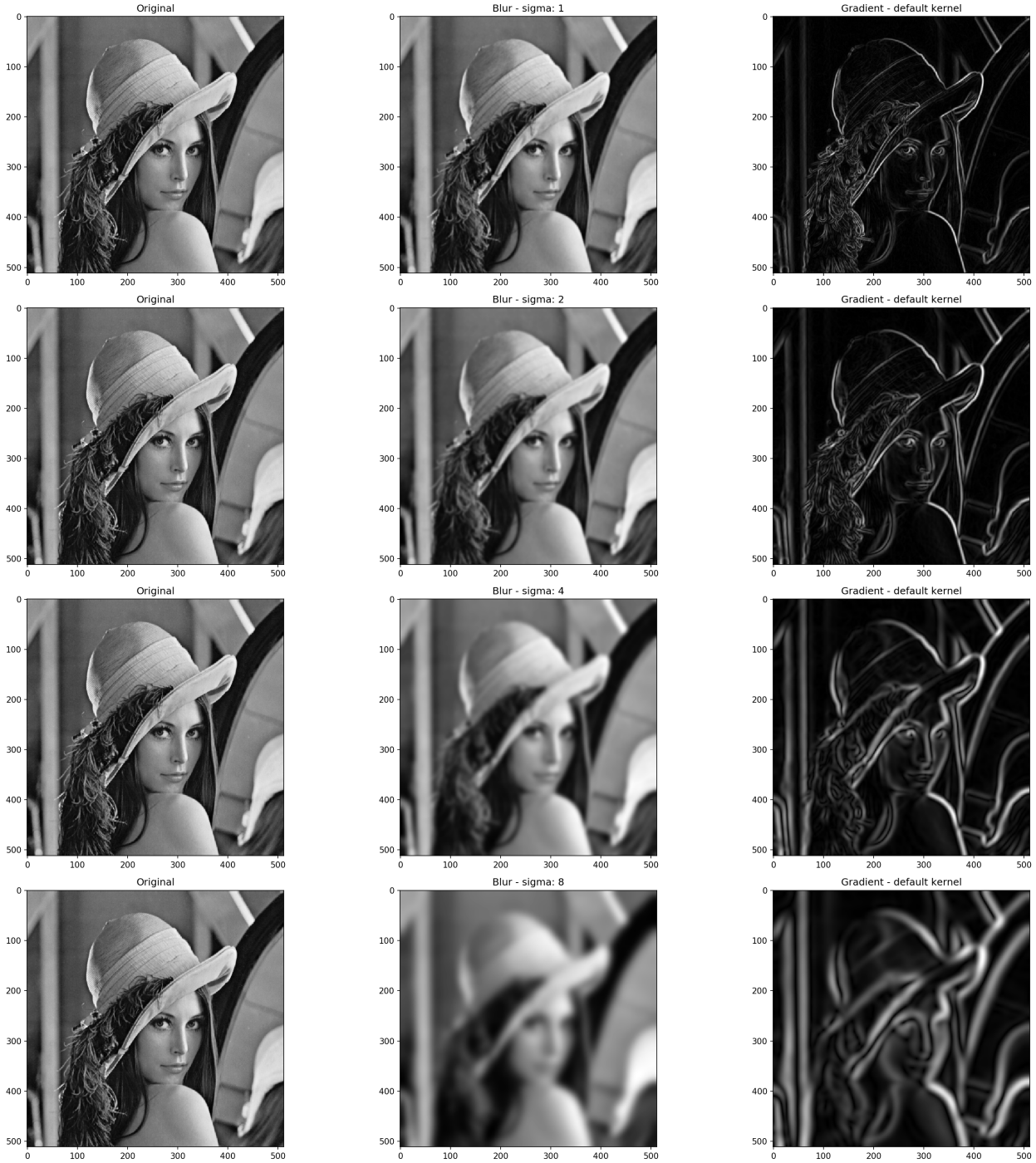

Figure 2: sobel

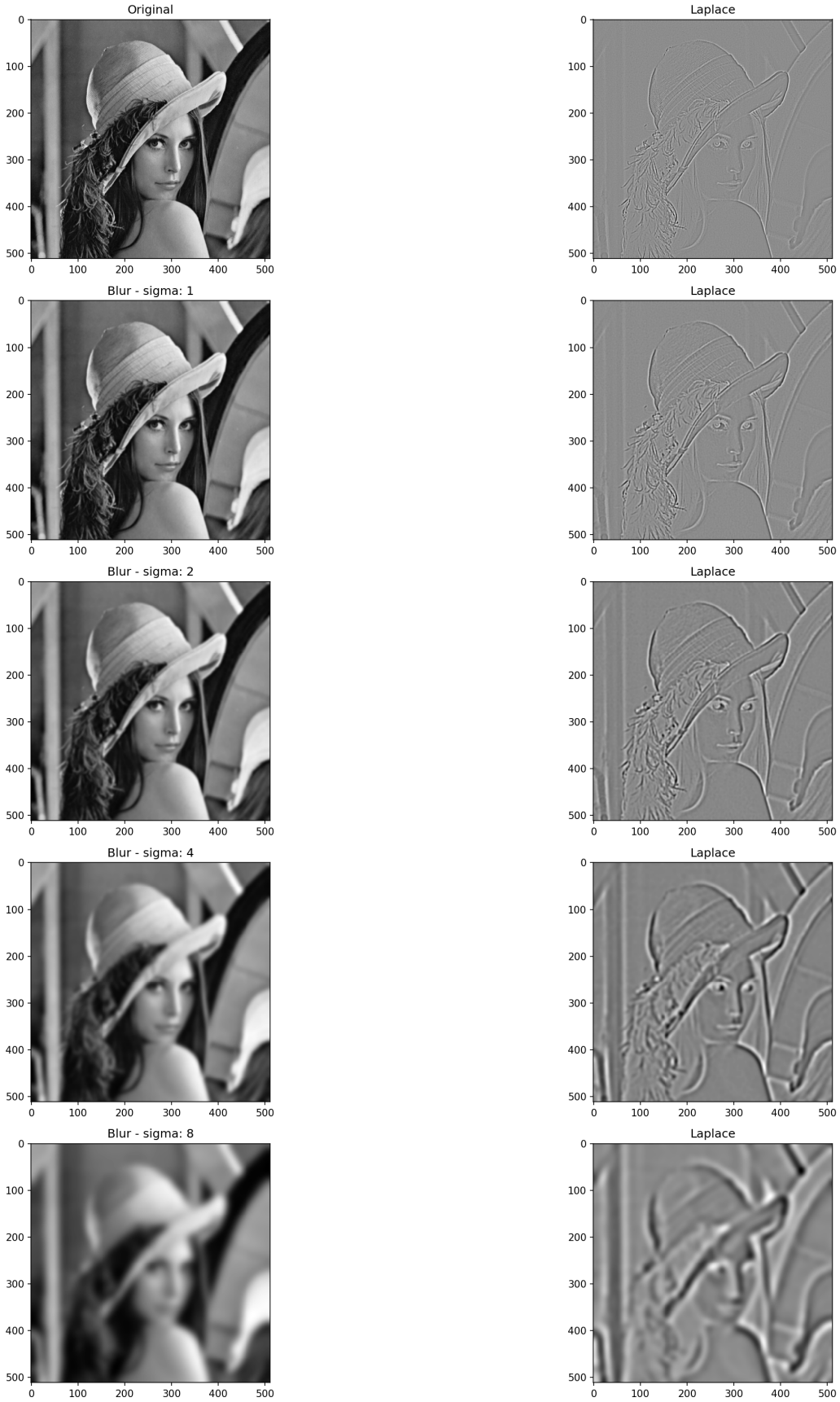

Figure 3: laplace

Exercise 4: Canny edge detection

The Canny algorithm is a multi-stage algorithm that relies on two main parameters: the `minVal` and the `maxVal`. Basically, these decide what counts as an edge and what doesn't.

"Any edges with intensity gradient more than maxVal are sure to be edges and those below minVal are sure to be non-edges, so discarded. Those who lie between these two thresholds are classified edges or non-edges based on their connectivity. If they are connected to"sure-edge" pixels, they are considered to be part of edges. Otherwise, they are also discarded." [1].

In the following section I will experiment with these two parameters.

Thresholds

See figure "canny 1"

Figure 4: canny 1

We can see that by setting the `minVal` too high we lose some of the details. Example: line of the hat is broken. Lenna's left chin line.

See figure "canny 2"

Above we have set the `minVal` to a lower value and we can see that the edges are now fuller (the hat, e.g.).

Let's see what changing the `maxVal` does.

See figure "canny 3"

We can see that the algorithm now becomes more 'pickier' on what counts as an edge and what doesn't. We can notice that the edge along Lenna's shoulder is no longer counted as an edge. This is perhaps due to the colour similarity in the chin area.

The `Canny` method allows for another parameter in calculating the edges, the `appertureSize`. In the following section I will experiment with this:

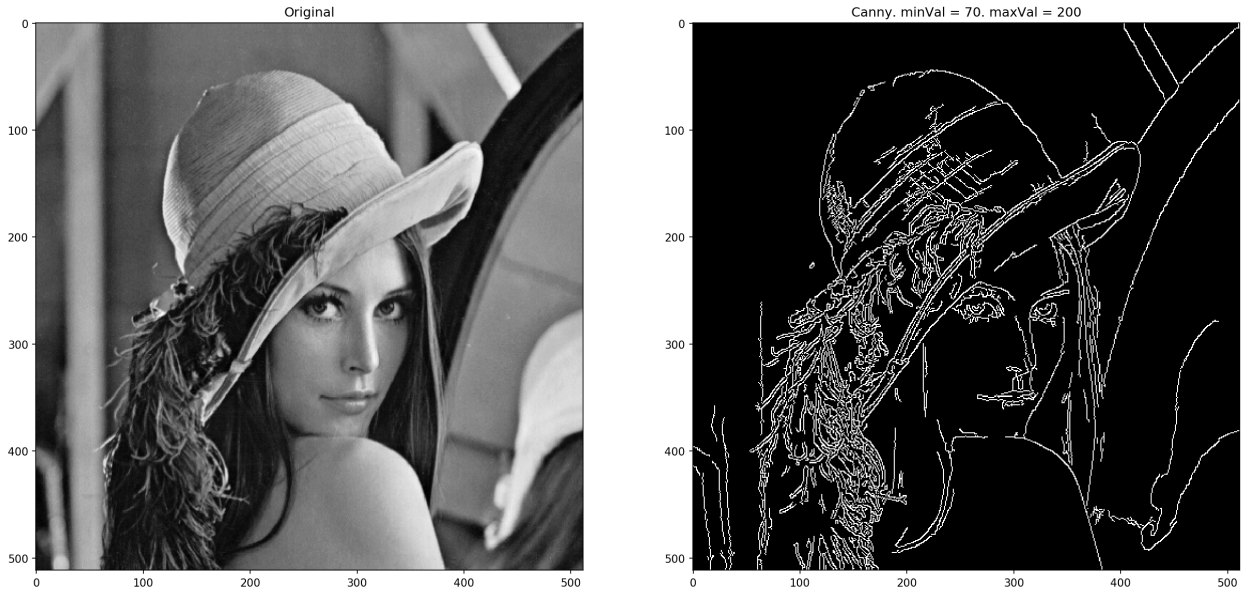

Figure 5: canny 2

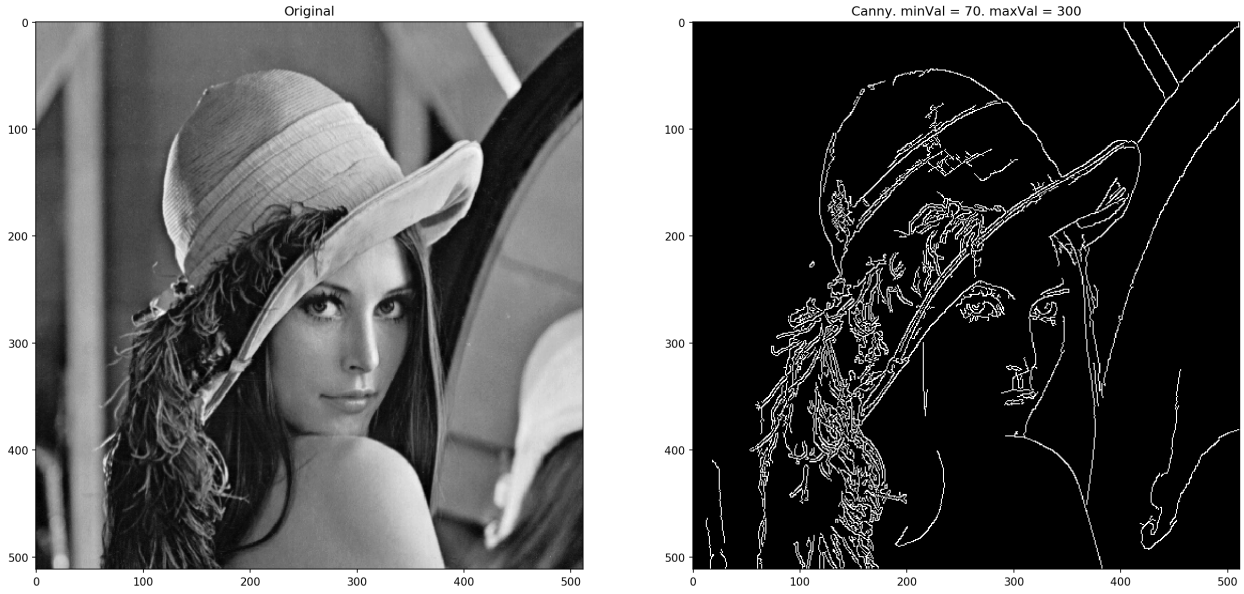

Figure 6: canny 3

Apperture

See figure "canny 4"

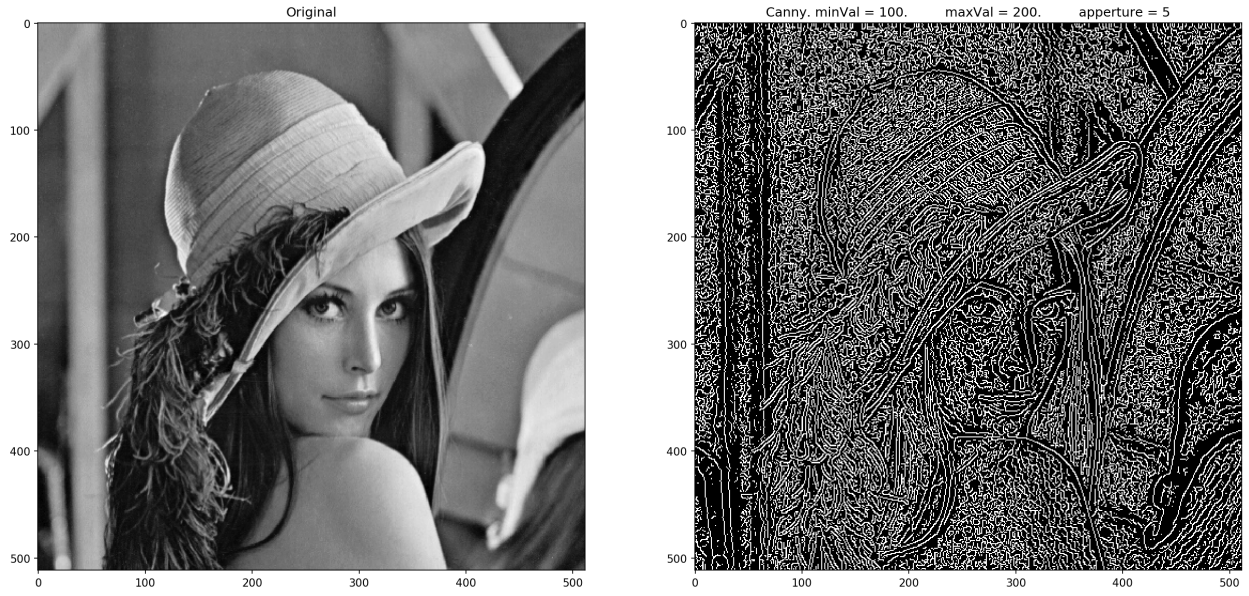

Figure 7: canny 4

In the above figure I experimented with changing the `apertureSize`. This parameter controls the "size of Sobel kernel used for find[ing] image gradients" [1]. We can see that increasing this parameter greatly increases the number of edges found. We then need to adjust the `minVal` and `maxVal` accordingly. See below:

See figure "canny 5"

Figure 8: canny 5

References

[1]: Gaussian blur. (2017, October 11). In Wikipedia, The Free Encyclopedia. Retrieved 14:37, December 6, 2017, from https://en.wikipedia.org/w/index.php?title=Gaussian_blur&oldid=804907373

[2]: "Image Filter & Edge Detection" [PDF slides]. Read on 03-12-2017. http://alumni.media.mit.edu/~maov/classes/vision09

[3]: OpenCv Documentation. "Sobel Derivatives". Online. Referenced on 03-12-2017. https://docs.opencv.org/2.4/doc/tutori

[4]. Abdul-Kader, Sameera. "Comparative Study for Edge Detection of Noisy Image using Sobel and Laplace Operators". Online. Read on 03-12-2017. https://pdfs.semanticscholar.org/8fad/fe64e9c2574c4825b8cb96c86b9e23234b76.pdf

[5]. OpenCV documentation. Online. Read 05-12-2017. https://docs.opencv.org/3.1.0/da/d22/tutorial_py_canny.html

[6]. Gaussian function. (2017, December 3). In Wikipedia, The Free Encyclopedia. Retrieved 14:37, December 6, 2017, from https://en.wikipedia.org/w/index.php?title=Gaussian_function&oldid=813435072

[7]. Convolution. (2017, November 28). In Wikipedia, The Free Encyclopedia. Retrieved 14:52, December 6, 2017, from https://en.wikipedia.org/w/index.php?title=Convolution&oldid=812489678

[8]. Kernel (image processing). (2017, September 2). In Wikipedia, The Free Encyclopedia. Retrieved 14:54, December 6, 2017, from https://en.wikipedia.org/w/index.php?title=Kernel_(image_processing)&oldid=798498421

[9]. "Guassian Blur". OpenCV Documentation. Online. Read 06-12-2017. https://docs.opencv.org/2.4/modules/imgproc/doc