Implementation of Image Filters

Prepared by
Aimee Valladares

Computer Vision
CAP 4410

27 September 2020

Table of Contents

List of Figures **2**

1. Image Filter Explanation **10**

- **1.1 Introduction** **10**
- **1.2 Project Description** **10**
- **1.3 Program Design** **11**

2. OpenCV Project **14**

- **2.1 Program Code** **14**
- **2.2 Test Plan** **23**
- **2.3 Test Results** **24**
- **2.4 Conclusion** **25**

References **26**

List of Figures

Figure 1 Gaussian Function …..………………………………………………………….................. 10

$$g(x,y) = e^{\frac{-(x^2+y^2}{2o^2}}$$

Figure 2 Original image……….. 24

Figure 3 Box Filter without function with 3x3 Sliding window 24

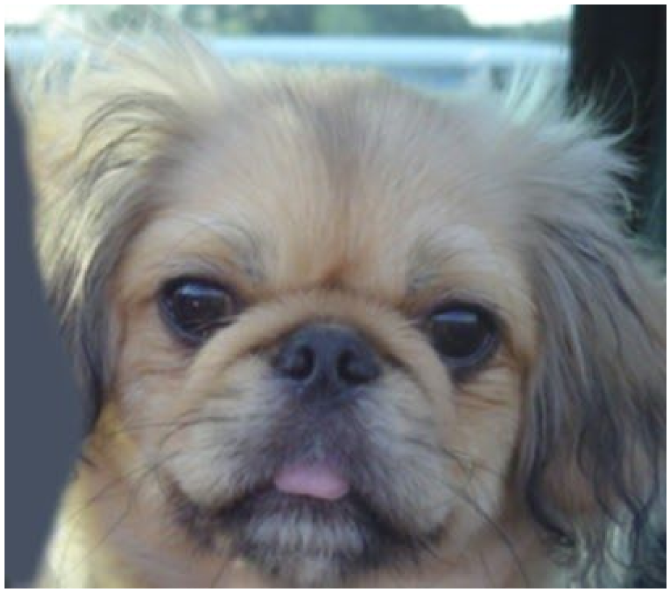


Figure 4 Box Filter without function with 7x7 Sliding window 24

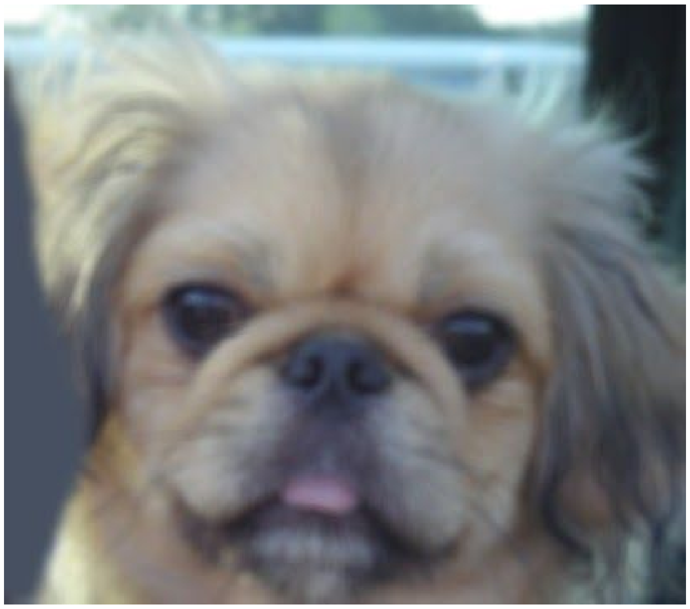


Figure 5 Box Filter with 3x3 Sliding window 24

Figure 6 Box Filter with 7x7 Sliding window 24

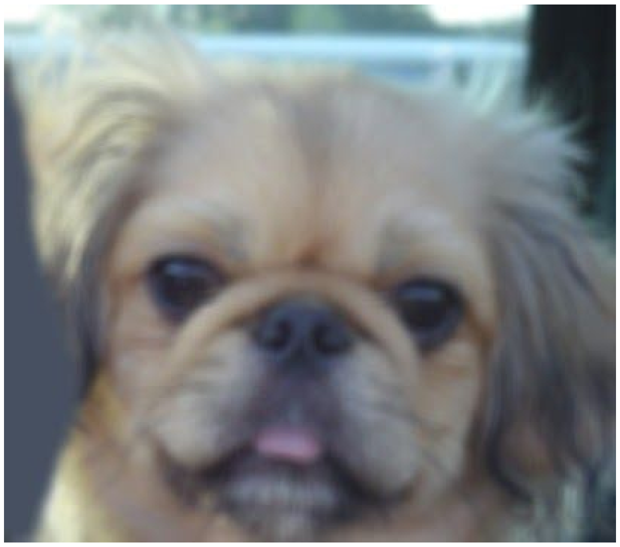


Figure 7 Sobel Edge Filter (x-axis) without function 3x3 Sliding Window 24

Figure 8 Sobel Edge Filter (x-axis) without function 7x7 Sliding Window 24

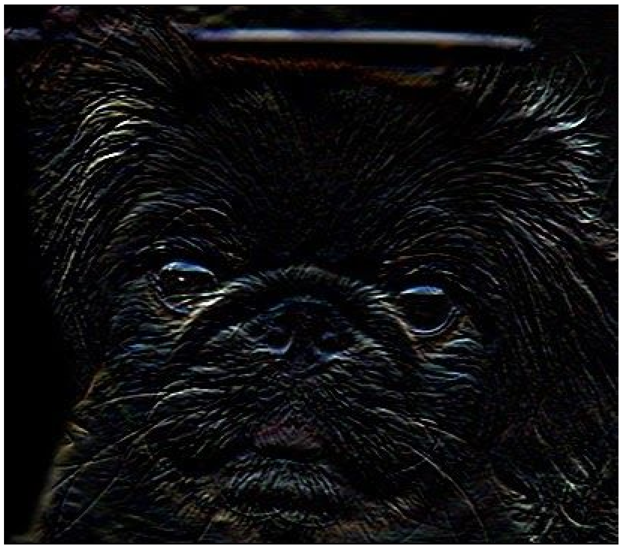


Figure 9 Sobel Edge Filter (y-axis) without function 3x3 Sliding Window 24

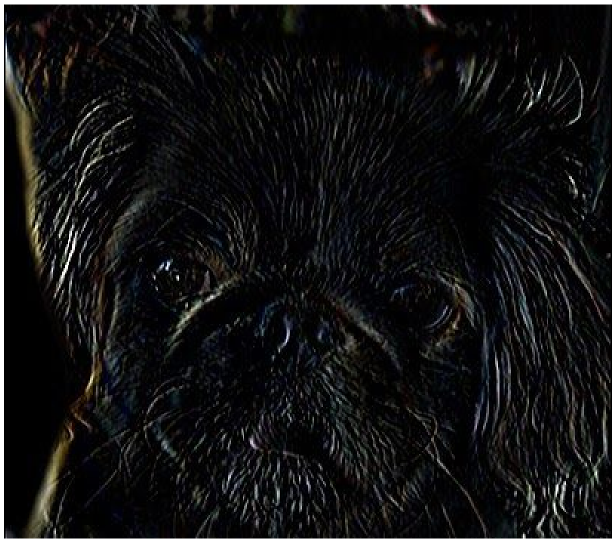


Figure 10 Sobel Edge Filter (y-axis) without function 7x7 Sliding Window …..................................... 24

Figure 11 Sobel Edge Filter without function 3x3 Sliding Window 24

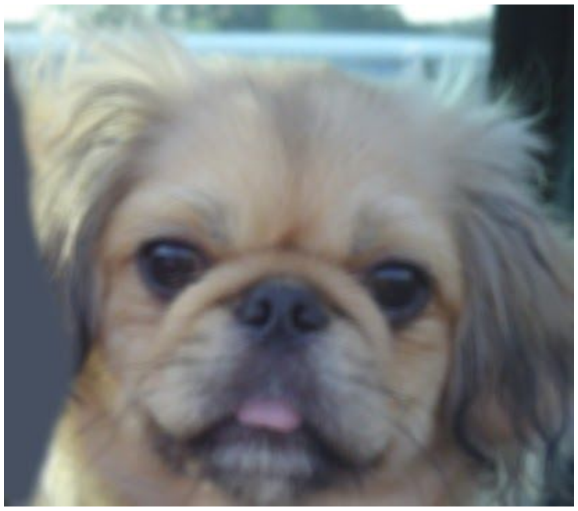
Figure 12 Sobel Edge Filter without function 7x7 Sliding Window 24

Figure 13 Sobel Edge Filter with 3x3 Sliding Window ….. 24

Figure 14 Sobel Edge Filter with 7x7 Sliding Window 24

Figure 15 Gaussian Filter with 3x3 Sliding Window …... 25

Figure 16 Gaussian Filter with 7x7 Sliding Window 25



1. Image Filter Explanation

1.1 Introduction

Image processing is the process of mapping an image into another image for analysis and enhancement. There are two main types of image processing: image filtering (which will be the focus of this report) and image warping. Image filtering is changing the pixel values or range of the image without changing the pixels positions by altering the colors of the image. Image filtering is primarily used to enhance or adjust an image's properties as well as extract information like the edges and corners [1]. These processes are done by *smoothing* an image which eliminates the "outliers" in the image's values considered to be unwanted data (i.e. noise) [2].

There are different types of image filters that smooth an image through different means. One filter may use a point, local, or global operator (i.e. the local mean operator, the local median operator, etc.) to modify the image while a different image filter will use a derivative operator that finds the edges in an image.

1.2 Project Description

In this project we are focusing on the effect different image filters have on an image. The image filters that will be utilized are the box filter, the Sobel Edge Detector filter, and the Gaussian filter. The box filter is a simple option for smoothing an image as it calculates the local mean in a sliding window by multiplying the image sample with the filter kernel and getting the result [2].

Another image filter used to modify an image is the Sobel Edge Detector filter. Unlike the box filter, the Sobel Edge filter primarily uses edge detection as it is a derivative mask that can detect two kinds of edges in an image: horizontal edges and vertical edges. This is done by first calculating the average smoothing with respect to *x* and then applying a derivative filter in *x* resulting with the edges with respect to *x.* Then a derivative filter is applied with respect to *y* and the average smoothing is calculated with respect to *y* resulting in the edges with respect to y. The results are then combined and compared to the threshold resulting in the edges for the image [3].

Finally, another filter that modifies the image through smoothing is the Gaussian filter. The Gaussian filter is similar to the box filter though the Gaussian filter convolves the image with the Gaussian function, as seen in Figure 1, for calculating the transformation to each pixel. Furthermore, the Gaussian filter is a low pass filter as it reduces the high-frequency image data [4].

The main objective of this project is to utilize several types of image filters and how the image changes when applying different sliding window sizes. This implementation will be done by creating an OpenCV project that will apply an image filter with different sliding window sizes. These programs will illustrate the effects of applying an image filter and how the results will change when using varying sliding window sizes.

1.3 Program Design

The OpenCV project in Visual Studio will comprise several C++ files that will implement a specific image filter through different means. The OpenCV code will comprise of seven C++ files with each file using two sliding window sizes, 3x3 and 7x7. The first file will apply the box filter *without* the use of the OpenCV built-in function and apply the two specified sliding window sizes stated earlier. The file will first read and load the image into the program through its file path as well as initialize variables meant to simulate the sliding window. The file will then copy the original file into two different variables where the image will be modified. Afterwards, an if conditional statement is used to help any issues that might arise if the program cannot load the image. Then the program will simulate the box filter function with for loops. Specifically, the for loops will count the number of rows and columns in the image and calculate the local average of the sliding window. Essentially, the program will act like a convolution filter that is multiplying the image sample with the filter kernel. The program will apply the for loops mentioned earlier to the 3x3 sliding window and the 7x7 sliding window. Finally, the program will create and open the windows and display the resulting images in the created windows [5].

The second file will apply the box filter *with* the use of the OpenCV built-in function and apply the two specified window sliding window sizes. This file will operate very similar to the previous file but utilize the boxFilter function. First the file will read and load the image through its file path. Then a conditional if statement will be used to stop the program if there is an issue in reading or loading the image into the program. The file will then initialize the output variables to hold the modified image data information. Then the program calls upon the boxFilter function. The boxFilter function allows the code to read the image, apply the sliding window to the image, and calculate the filtered results by finding the local average. Finally, the program will create and open the windows and display the resulting images in the created windows.

The third file will apply the Sobel filter primarily focusing on the x-axis edges or the horizontal edges *without* the use of the OpenCV built-in function. This file will also illustrate the filter effects when applying it to the two specified sliding windows. The file declares two functions: one that calculates the gradient vector and another that functions as the main. The gradient vector function essentially filters the image by multiplying it with the derivative with respect to x. Meanwhile, the main function reads the data image, calls upon the gradient function, and displays the program results. First the file will read and load the image through its file path. The file will also include a conditional if statement that will cause the program to stop running if the program cannot read or load the images. The program will then call upon the GaussianBlur function that will smooth the image with respect to x as well as apply the sliding window. Then for loops are used to call upon the gradient function and calculate the first order derivatives as well as change the image to black and white to demonstrate the edges of the image. Finally, the program will create and open the windows and display the resulting images in the created windows [6].

The fourth file will apply the Sobel filter that will primarily focus on the y-axis edges or the vertical edges *without* the use of the OpenCV built-in function. This file will also demonstrate the filter effects when applying the two specified sliding windows. The file declares two functions: one that calculates the gradient vector and another that functions as the main. The gradient vector function essentially filters the image by multiplying it with the derivative with respect to y. Meanwhile, the main function reads the data image, calls upon the gradient function, and displays the program results. First the file will read and load the image through its file path. The file will also include a conditional if statement that will cause the program to stop running if the program cannot read or load the images. The program will then call upon the GaussianBlur function that will smooth the image with respect to y as well as apply the sliding window. Note that the order of smoothing the image and then calculating the derivative with respect to y in the program is not indicative of how the Sobel Edge Detection blurs and filters the image. Then for loops are used to call upon the gradient function and calculate the first order derivatives as well as change the image to black and white to demonstrate the edges of the image. Finally, the program will create and open the windows and display the resulting images in the created windows [6].

The fifth file will apply the Sobel filter involving the x-axis and the y-axis edges *without* the use of the OpenCV built-in function. This file will essentially combine the third and fourth file effectively displaying the edges of the image. This file will also illustrate the filter effects when applying the two specified sliding windows. The file declares three functions: one that calculates the gradient vector with respect to x, another that calculates the gradient vector with respect to y, and a function that serves as the main. The gradient vector functions essentially filters the image by multiplying it with the derivative with respect to x and y. Meanwhile, the main function reads the data image, calls upon the gradient function, and displays the program results. First the file will read and load the image through its file path. The file will also include a conditional if statement that will cause the program to stop running if the program cannot read or load the images. The program will then call upon the GaussianBlur function that will smooth the image with respect to x and y as well as apply the sliding window. Then for loops are used to call upon the gradient functions and calculate the first order derivatives as well as change the image to black and white to demonstrate the edges of the image. Finally, the program will create and open the windows and display the resulting images in the created windows [6].

The sixth file will apply the Sobel filter with the x-axis and the y-axis edges *with* the use of the OpenCV built-in function. This file will also demonstrate the filter effects when applying the two specified sliding windows. First the file will read and load the image through its file path. The file will also include a conditional statement that will cause the program to stop running if the program cannot read or load the images. The program will then call upon a GaussianBlur function that will essentially blur the image and remove unwanted data or noise with respect to x and y. Afterwards the program will call upon the Sobel function that takes into account the image depth, the x-derivative, and the y-derivative. Finally, the file will create and open the windows and display the resulting images in the created windows.

Finally, the seventh file will apply the Gaussian filter *with* the use of the OpenCV built-in function. This file will also demonstrate the filter effects when applying the two specified sliding windows. First the file will read and load the image through its file path. The file will then use a conditional if statement that will cause the program to stop running in case there is an issue reading or loading the image. Afterwards the file will call upon the GaussianBlur function that calculates the Gaussian function, as shown in Figure 1 as well as apply the sliding window. Finally, the file will create and open the windows and display the resulting images in the created windows.

The OpenCV project files can be seen in the section labelled “2.1 Program Code” under the heading “2. OpenCV Project”.

For testing purposes, the results for each of the filters will be compared by the results from the different sliding window sizes. This means that the result from the 3x3 sliding window will be compared to the result from the 7x7 sliding window for each filter. Additionally, the effects from each filter will be compared to another filter. In other words, the behavior of the change displayed by the box filter will be compared to the changes by the Sobel filter and the Gaussian filter.

2. OpenCV Project

2.1 Program Code

The OpenCV Project was written in C++ with Visual Studio with the program consisting of seven files. The program source code is included in the ZIP folder provided. The files are well-commented, with extended comments used to help distinguish between the seven files and help explain the logic of the code [7] [8] [9] [10][11].

2.2 Test Plan

To properly demonstrate the performance of the OpenCV project code by running each program and analyzing the outputs. To analyze the accuracy of the file applying the image filters we first compare the sliding windows used in each file. This means that for each program we will compare the modified image with the 3x3 sliding window to the modified image with the 7x7 sliding window. Moreover, we will compare the results from each applied image filter and interpret each image filter's behavior. Specifically, we will interpret how the image filter affects and modifies the original image.

Each of the files will generate 3 different images in separate windows. The first image will illustrate the original unedited image. The second image will illustrate the original image being modified by the applied image filter with a 3x3 sliding window. Finally, the third image will demonstrate the image filter with a 7x7 sliding window being applied to the original image.

The first C++ file will generate 3 different images while utilizing the box filter function and apply the two specified sliding window sizes. While the second C++ file will generate 3 different images while *simulating* the box filter function and applying the two specified sliding windows. To verify the result of these two C++ files we will compare the two files outputs.

Meanwhile the third C++ file will generate 3 different images while *simulating* the Sobel edge filter towards the x-axis edges. The fourth C++ file will generate 3 different images while *simulating* the Sobel edge filter towards the y-axis. The fifth C++ file will essentially combine the code of the third and fourth C++ files as this file will generate 3 different images while *simulating* the Sobel edge filter towards both the x-axis edges and the y-axis edges. The sixth C++ file will generate 3 different images while using the Sobel edge filter function. To verify the results between these four C++ files we will compare the four files outputs.

Finally, the seventh C++ file will generate 3 different images while utilizing and applying the Gaussian filter. To verify the results from the use of the box filter, the Sobel edge detector, and the Gaussian filter we will compare the outputs and how they modify the image

2.3 Test Results

The OpenCV project was operational and accurate (within a margin error) in depicting the effects of applying an image filter on an image. This is illustrated by the program results seen between Figure 2 and Figures 3 - 16.

Figures 3 and 4 depict the results of the C++ file simulates the box filter. When comparing the original image seen in Figure 2 with Figures 3 and 4, the image blurs as the file smooths the image. However, when comparing the output of Figure 3 with Figure 4, the file illustrates that the box filter seems sufficient when dealing with smaller sliding windows like 3x3 and 5x5 while becoming less effective when dealing with larger sliding windows like 7x7. This is further illustrated by the images displayed in Figures 5 and 6 which depict the results from the C++ file that utilizes the box filter.

Figures 7 and 8 depict the results of the C++ file that simulates the Sobel edge filter towards the x-axis edges. When comparing the original image seen in Figure 2 with Figures 7 and 8, the image will blur as the file smooths the image and applies a horizontal mask that makes all the horizontal edges visible. This illustrates that the Sobel edge filter towards the x-axis edges is effective when dealing with larger sliding windows as the horizontal edges are still emphasized when a 7x7 sliding window is applied.

Figures 9 and 10 depict the results of the C++ file that simulates the Sobel edge filter towards the y-axis edges. When comparing the original image seen in Figure 2 with Figures 9 and 10, the image will blur as the file smooths the image and applies a vertical mask that makes all the vertical edges visible. This illustrates that the Sobel edge filter towards the y-axis edges is effective when dealing with larger sliding windows as the vertical edges are still emphasized when a 7x7 sliding window is applied.

Figures 11 and 12 depict the results of the C++ file that simulates the Sobel edge filter towards the x-axis edges and the y-axis edges. When comparing the original image seen in Figure 2 with Figures 11 and 12, the image will blur as the file smooths the image and combines and applies the horizontal mask and vertical mask that will make the edges visible. This illustrates that the Sobel Edge filter is effective when dealing with larger sliding windows though the edges are not as emphasized. This is further illustrated by the images displayed in Figures 13 and 14 which depicts the results from the C++ file that utilizes the Sobel edge filter.

Finally, Figures 15 and 16 depict the results of the C++ file that utilizes the Gaussian filter and smooths the image causing the image. When comparing the original image seen in Figure 2 with Figures 15 and 16, the file illustrates that the Gaussian filter is more efficient at handling larger sliding windows especially when compared to the results from the file utilizing the box filter. This is illustrated by doing a side-by-side comparison of Figure 6, the box filter with a 7x7 sliding window, and Figure 16, the Gaussian filter with a 7x7 sliding window.

2.4 Conclusion

Image filters are an important feature of image processing that modifies the image without changing the pixel data and . Consequently, in this project we learned that the box filter and the Gaussian filter will smooth the image through different methods and cause the image to blur. The method the box filter uses is calculating the local average of the sliding window. The method the Gaussian filter uses is calculating the Gaussian function, shown in Figure 1, while applying the sliding window. Meanwhile the Sobel edge filter not only smooths the image with a Gaussian function but also applies a horizontal and vertical mask that will indicate the edges in the image. Furthermore, the project has illustrated that the Gaussian filter is adept at handling the larger sliding windows for image smoothing when compared to the box filter.

In the future, I would like to make several improvements that would help improve the efficiency and usability of the OpenCV project. For instance, one possible improvement I would make to the OpenCV project is to enable user input that will allow the user to input the sliding window or mask size into the file. Another improvement I would make to my OpenCV project is to increase the accuracy of the files that do not utilize the built-in functions. The results from the files that do not utilize the built-in functions are not as accurate as the results from the files that utilize the built-in functions. This is emphasized when doing a side-by-side comparison of Figures 3 and 4 to Figures 5 and 6. Finally, another improvement I would make to my OpenCV project is to employ a better function for smoothing the image in the files that simulate the Sobel edge filter towards the x-axis or the y-axis.

References

[1] “Tutorial 1: Image Filtering.” Stanford Artificial Intelligence Laboratory, ai.stanford.edu/~syyeung/cvweb/tutorial1.html.

[2] Abid, Muhammad. Lecture 06 Slides, CANVAS *Concise Computer Vision*.

[3] Abid, Muhammad. Lecture 07 Slides, CANVAS *Concise Computer Vision.*

[4] “Gaussian Blur.” *Wikipedia*, Wikimedia Foundation, 20 Sept. 2020, en.wikipedia.org/wiki/Gaussian_blur.

[5] Toshniwal, Arjun. *Opencv C++ Code for Low Pass Averaging Filter*, 1 Jan. 1970, opencv-tutorials-hub.blogspot.com/2016/02/opencv-code-for-low-pass-averaging-filter.html.

[6] “Sobel Operator.” *Tutorialspoint*, www.tutorialspoint.com/dip/sobel_operator.htm.

[7] *OpenCV Filters - BoxFilter*, milindapro.blogspot.com/2015/05/opencv-filters-boxfilter.html.

[8] “Sobel Derivatives.” *OpenCV*, docs.opencv.org/3.4/d2/d2c/tutorial_sobel_derivatives.html.

[9] Programming Techniques. “Sobel and Prewitt Edge Detector in C++: Image Processing.” *Programming Techniques*, 30 Jan. 2019, www.programming-techniques.com/2013/03/sobel-and-prewitt-edge-detector-in-c-image-processing.html.

[10] “Image Filtering.” *OpenCV*, docs.opencv.org/3.4/d4/d86/group__imgproc__filter.html.

[11] “OpenCV - Sobel Operator.” *Tutorialspoint*, www.tutorialspoint.com/opencv/opencv_sobel_operator.htm.