American University of Armenia

College of Science and Engineering

Numerical Analysis Project

Edge Detection

Team: Karlos Muradyan and Thomas Petrossian

Fall, 2017

Abstract

Edge detection problem in computer sciences is one that gets the attention of many professors and programmers. Today, there are lots of different ways to detect edges of the picture, but none of them is perfect yet. In our project, firstly we used Fourier Transforms to get the representation of the desired picture in frequency domain. After which we were able to easily do High-Pass filtering to get rid of low frequencies. But as the output was not clear enough, we decided to use in our project Butterworth Filtering, which made the resulting picture “smoother”, so after doing inverse Fourier Transform edges can be viewed more or less clear. So, the code of our project gets a picture and displays edges of it.

Keywords: edge detection, Fourier Transform, Fourier Series, Butterworth Filter, High-Pass filtering, Low-Pass filtering, edges of the image**,** time domain, frequency domain.

Contents

Chapter 1

Introduction

1.1 Problem Setting and Description

Edges are one of the basic features of the image. Edge is called the bound of the two different objects in an image. Edge detection is the action after which the edges, that is the bounds of the two different objects can be found. It is very important operation in image processing, because its use can significantly reduce the time and amount of data used for doing different filtering and operation with images. It can help to solve very difficult high level image processing task, such as image segmentation. The last can be easily done after feature detection and feature extraction of the images, that is the main goal of edge detection algorithms. It specifies the regions that the intensity changes remarkably, so that the color, texture, intensity or grey level changes sharply in small distance of the image.

1.2 Known methods of implementation

There are many ways of detecting edges, but almost all of them use the common ideas behind the problem. They rely on brightness discontinuities or strict frequency changes. Some of the well-known ways of detecting edges are Sobel, Canny, Prewitt, fuzzy logic methods. Sobel method uses discrete differentiation operator and computes the approximation if the gradient of the image intensity function. Its advantage is its relative inexpensiveness for the computer, as it just uses filters in the horizontal and vertical directions that return integer values. Fuzzy Logic implementation, in contrast uses some membership functions to understand if the pixel is inside the object region or in the boundary of two objects. It is more expensive but gives better results in most of the cases. Our approach is a use of high-pass and butterworth filtering on the Fourier Transformed image that also makes computations relatively inexpensive for a computer.

1.3 The structure of the project paper

The project paper consists of six chapters. The first one discussed problem setting and description, as well as some basic information about known methods of implementation. The econd chapter is devoted to theoritical background of the project and gives information about Fourier Series for very basic 2π periodic functions till Fourier Transformation definition for complex functions. Chapter three discusses image filters used in our project, namely Butterworth BandPass and High-Pass, Low-Pass filters. Chapter four discusses result and conclusion of our project and discusses some additions that can be made in the future. Appendices are given in chapter five. Chapter six is the last chapter in our project paper, where bibliography is provided.

Chapter 2

Theoritical Background

2.1 Fourier Series of period 2π

In mathematics, Fourier Series is a way of representation any function f(x) in the terms of sum of simple trigonometric waves. In other words any function can be represented as sum of simple oscillating functions, namely sines and cosines (or complex exponentials for complex valued functions). This representation is very useful for manipulating with the given functions, as manipulation with sine and cosine functions is easier and much more inexpensive.

Many very complex functions that are quite unusual to have such property and for understanding the logic behind the claim let's firstly examine known trigonometric equations:

$$\cos(2t) = -1 + 2\cos^2(t) \quad \text{that is the same as writing} \quad \cos^2(t) = \frac{1}{2}\cos(2t) + \frac{1}{2}$$

By just looking on the right equation, it is obvious that $\cos^2(t)$ is sum of constant member and a function with period of π $\cos(2t)$. Another, more complex trigonometric formula is

$$\sin^4(t) = \frac{3}{8} - \frac{1}{2}\cos(2t) + \frac{1}{8}\cos(4t)$$

And again it is visible that the equation consists of constant member and two cosine functions. As this equation has two functions in the right hand side, we start considering also the frequency each of the functions. Here, the periods and frequences of each functions are the following: $\cos(2t)$ has period of π and frequency of $1/\pi$, $\cos(4t)$ has $\pi/2$ with frequency of $2/\pi$ and finally, $\sin^4(t)$ has period of π with frequency of $1/\pi$. It can be noted that in the right hand side the frequences of the functions are different. In the case if they were the same, cosine functions could have been merged into single one and we would have only one consine function with a different constant multiplicand. Moreover, we can notice another constant value in the sum. The reason for that is the positiveness of the base functions $\cos^2(t)$ and $\sin^4(t)$.

They both are non-negative and as any function with form $\cos(nx)$ or $\sin(nx)$ is symmetric and has zero average value on its period, there should be a constant member that will equalize the values of left and right hand sides.

But what about non-trigonometric functions? At first consider functions that have periods of 2π. Fourier Series assumtion claims that each function can be represented as sum of sine and cosine functions. That means that we can have infinitely many sine and cosine functions, but with different frequences. Let's do another assumtion of its frequences. Take the frequences of each sine and consine functions as two times of the previous one, such that the frequency of the first sine or cosine function is 1/2π, the second one 1/π, the third one 2/π and etc. In other words we expect having frequences $\frac{n}{2\pi}$ where n=1,2,… . Moreover, the right hand side should also have some constant value that will be present if average of our considerable f(x) function is not zero in the specified period. And finally, each sine and cosine function should have constant multiplicand $a_k \text{or} b_k$, that will specify the desired magnitude of the sine or cosine wave. So, the final representation will have the following form in general case:

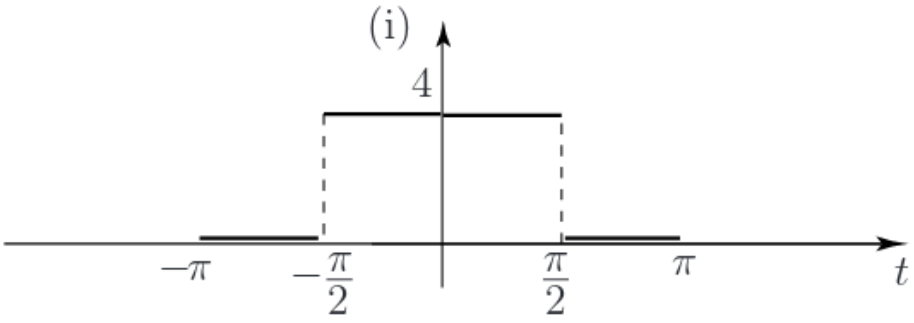
$$f(x)=\frac{1}{2}a_0+a_1\cos(t)+a_2\cos(2t)+...+b_1\sin(t)+b_2\cos(2t)+...$$

where $\frac{1}{2}a_0$ is the constant value, $a_1, a_2, ..., b_1, b_2, ...$ constant multiplicands that also are called ***Fourier Coefficients***.

Let's consider a practical example. Take the following function into consideration:

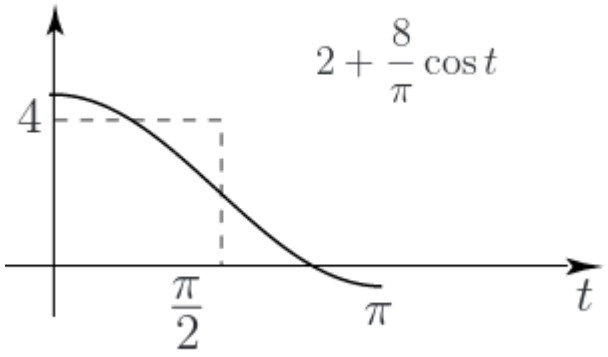
$$f(t) = \begin{cases} 4 & \frac{-\pi}{2} < t < \frac{\pi}{2} \\ 0 & -\pi < t < \frac{-\pi}{2} \end{cases}$$

The function is defined only in the region between – π and π, however it also can be viewed as a periodic function with period if 2 π. In x-y plane the function looks like:

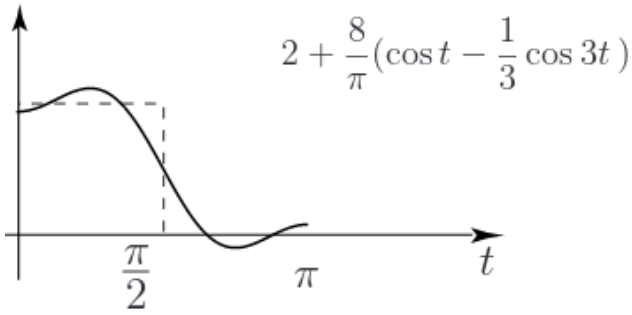


By writing some of the terms of Fourier Series, we can see the approximation based on the number of terms inserted (how the values are obtained, we will discuss later).

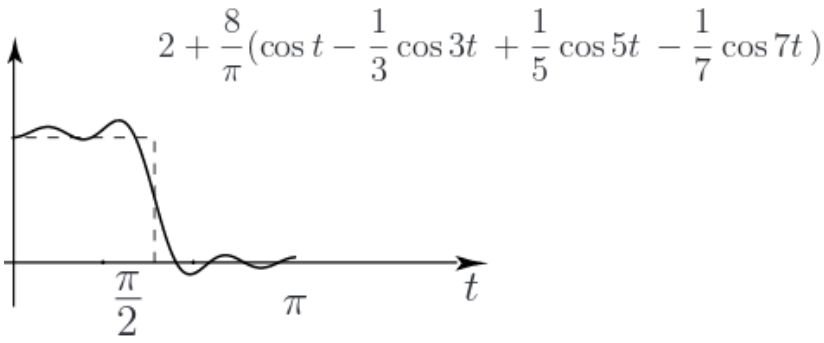
For just one term, the function approximation will be:



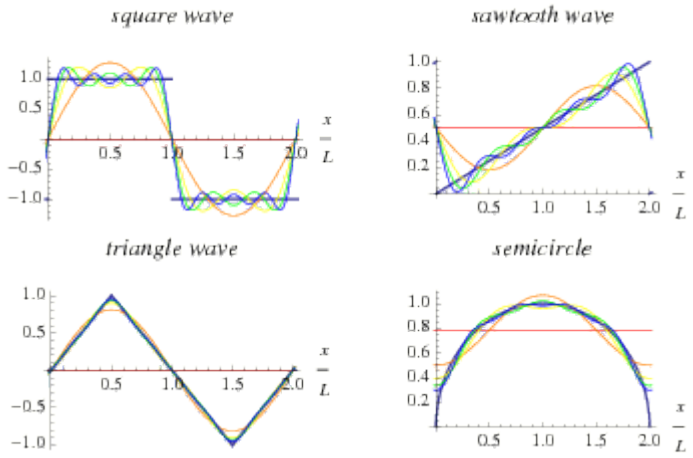
The approximation will become better with addition of the second term:



And even better with third and fourth terms:



Another more complex examples are the following functions that already are shown with their Fourier Series representations:



Fourier Coefficients:

As we already know, periodic function can be represented with the Fourier Series:

$$f(x)=\frac{1}{2}a_0+\sum_{n=1}^{\infty}a_n\cos(nx)+\sum_{n=0}^{\infty}b_n\cos(nx) \qquad (1)$$

Yet, it is still not known how to find Fourier coefficients $a_1,a_2,\ldots,b_1,b_2,\ldots$. Without loss of generality let's consider finding a_n . For that, we firstly multiply both sides by $\cos(mx)$ and integrate both sides from – π to π.

$$\int_{-\pi}^{\pi}f(x)\cos(mx)=\frac{a_0}{2}\int_{-\pi}^{\pi}\cos(mx)\,dx+\sum_{n=1}^{\infty}\left(a_n\int_{-\pi}^{\pi}\cos(nx)\cos(mx)\,dx+b_n\int_{-\pi}^{\pi}\sin(nx)\cos(mx)\,dx\right) \qquad (2)$$

By integral properties, it is possible to show that:

$$\int_{-\pi}^{\pi}\sin(nx)\sin(mx)\,dx=\pi\,\delta_{mn}$$
$$\int_{-\pi}^{\pi}\cos(nx)\cos(mx)\,dx=\pi\,\delta_{mn}$$
$$\int_{-\pi}^{\pi}\cos(nx)\sin(mx)\,dx=0$$

where δ_{mn} is Kronecker Delta. Kronecker delta is function of two variables and has a value 0 if m isn't equal to n. Otherwise it has a value 1.

Therefore, using these integral properties, all the integrals in (2) equation are 0, except the cases when m and n are equal. In that case we will have $\int_{-\pi}^{\pi}\cos^2 mt\,dt=\pi$, so (2) equation will have the following form:

$$\int_{-\pi}^{\pi} f(x)\cos mx\,dx = a_m \pi$$

And by solving that equation in terms of a_m , we will get:

$$a_m = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos mx\,dx \quad \text{the same as} \quad a_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos nx\,dx$$

The same method that we used to compute a_n , b_n also can be computed but instead of multiplying both sides of (1) equation by $\cos(mx)$, we should multiply by $\sin(mx)$ and do the same steps.

2.2 Fourier Series for general period

Yet, we just considered functions with period of 2π and got the equations of Fourier Series for that specific functions. Now, its time to generalize the equations for functions of any period. In the previous chapter we had a discrete variable "x", but for general period we should substitute $x = 2\pi t/P$, where P is the period of the function. Also as we now consider the variable "t" instead of "x", our function f(x) becomes f(t). So, making the substitutions in the equations obtained in the previous chapter, we will get:

$$a_n = \frac{2}{P}\int_{-\frac{P}{2}}^{\frac{P}{2}} f(t)\cos\left(\frac{2n\pi t}{P}\right)dt$$

$$b_n = \frac{2}{P}\int_{-\frac{P}{2}}^{\frac{P}{2}} f(t)\sin\left(\frac{2n\pi t}{P}\right)dt$$

and the Fourier Series representation will be:

$$f(x) = \frac{1}{2}a_0 + \sum_{n=1}^{\infty} a_n \cos\left(\frac{2n\pi t}{P}\right) + \sum_{n=0}^{\infty} b_n \cos\left(\frac{2n\pi t}{P}\right)$$

So, for any periodic function we can construct Fourier Series, i.e. each periodic finction can be represented as sum of sine and cosine functions (in some cases with constant $\frac{1}{2}a_0$ also). But still, it is undefined Fourier Series for non-periodic functions. It turns out that Fourier Series exist also for non-periodic functions, however, that is done with Fourier Transformation. The last is powerful source of doing many things in different dimensions. Not only Fourier Transformation can be done in one dimentional space (audio signals, one variable functions, etc.) but also in two (matrices) and multidimentional domains.

2.3 Fourier Series for complex functions

Fourier Series can also be extended to complex-valued functions with complex coefficients. Consider some real-valued function f(x) and consider the equation:

$$f(x)=\sum_{n=-\infty}^{\infty} A_n e^{inx} \qquad (3)$$

where A_n is some coefficient. Now consider the integral

$$\int_{-\pi}^{\pi} f(x)e^{-imx}dx \qquad (4)$$

and substitute value of (3).

$$\int_{-\pi}^{\pi} f(x)e^{-imx}dx=\int_{-\pi}^{\pi}\sum_{n=-\infty}^{\infty} A_n e^{inx}e^{-imx}dx=\sum_{n=-\infty}^{\infty} A_n\int_{-\pi}^{\pi} e^{i(n-m)x}dx=$$

$$=\sum_{n=-\infty}^{\infty} A_n\int_{-\pi}^{\pi}\{\cos[(n-m)x]+i\sin[(n-m)x]\}dx=\sum_{n=-\infty}^{\infty} A_n 2\pi\delta_{mn}$$

Therefore,

$$\int_{-\pi}^{\pi} f(x)e^{-imx}dx=2\pi A_m,$$

And finally,

$$A_m=\frac{1}{2\pi}\int_{-\pi}^{\pi} f(x)e^{-imx}dx \quad \text{the same as writing} \quad A_n=\frac{1}{2\pi}\int_{-\pi}^{\pi} f(x)e^{-inx}dx$$

Discrete Coefficients can be expressed like:

$$A_m = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(x)e^{-imx}\,dx \quad = \quad \begin{cases} \frac{1}{2}\pi \sum_{n=-\infty}^{\infty} A_n \int_{-\pi}^{\pi} \{\ \cos\left[(nx)+i\sin\left(|n|x\right)\right]\}\,dx & n<0 \\ \frac{1}{2\pi}\int_{-\pi}^{\pi} f(x)\,dx & n=0 \\ \frac{1}{2}\pi \sum_{n=-\infty}^{\infty} A_n \int_{-\pi}^{\pi} \{\ \cos\left[(nx)+i\sin(nx)\right]\}\,dx & n<0 \end{cases}$$

$$= \begin{cases} \frac{1}{2}(a_n + i\,b_n) & \text{for } n<0 \\ \frac{1}{2}(a_0) & \text{for } n=0 \\ \frac{1}{2}(a_n - i\,b_n) & \text{for } n>0 \end{cases}$$

And for the function that is periodic in [-P/2, P/2]:

$$f(x) = \sum_{n=-\infty}^{\infty} A_n e^{i(2\pi nx/P)}$$

$$A_n = \frac{1}{P}\int_{-\frac{P}{2}}^{\frac{P}{2}} f(x)e^{-i(2\pi nx/P)}$$

2.4 Fourier Transformation

As it was mentioned earlier, non-periodic functions can also be represented in Fourier Series form, but that conversion is called Fourier Transformation. In the previous chapters, we considered periodic functions and found the Fourier series representations despite of its periods. Now, let's consider functions that do not have a periods. This means that now, we don't have a discrete coefficient A_n representing Fourier Series, instead we have a continuous function F(n). So instead of (3) equation, now let's consider a new equation using F(n) function:

$$f(x) = \int_{-\infty}^{\infty} F(k) e^{2\pi ikx} dk \qquad (5)$$

where i is complex number.

That F(k) function actually represents Fourier Transform itself and is frequency function of the given f(x). Fourier Transform can be computed by this equation:

$$F(k) = \int_{-\infty}^{\infty} f(x) e^{-2\pi ikx} dk \qquad (6)$$

Computation of the above mentioned integral (equation (6)) is actually called forward Fourier Transformation (-i), whereas computation of the real function f(x) (equation (5)) is called inverse Fourier Transformation (+i). Forward Fourier Transform is the computation that returns a function in frequency domain (more about time and frequency domain is written in Appendice section), while inverse Fourier Transform gives a function in time domain.

In computer computations, this algorithm will take n^2 complexity for n points. Now, there exitst another algorithm that reduces the time complexity of the algorithm that does all the computations for the same nomber n points in nlg(n) time. This algorithm is called ***FFT*** (Fast Fourier Transformation) and is widely used nowadays.

2.5 Fourier Transformation in two dimension

Fourier Transformation is not limited in only one dimension. If function is defined on x-y plane, it is in two dimension and regarding its continuity, it can have 2 different Fourier Trasformation forms (each with its forward and inverse transformations). In discrete case, we have sums as we had in one dimension casse. So, forward Fourier transformation (during which we computed function in frequency domain) will be computed using the following form:

$$F(u,v)=\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} f[m,n]e^{-i2\pi(umx_0+vny_0)}$$

where x_0 and y_0 are the spartial intervals between consecutive signals in the x and y directions .

In continuous case, forward Fourier transformation has little bit different form. The sums become integrals, so that the transformation has this form:

$$F(u,v)=\int_{\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-i2\pi(ux+vy)}\,dx\,dy$$

where u, v are spartial frequences in x and y directions respectively. As in our project we use both forward and inverse Fourier Transformations, therefore we should also define the inverse Fourier Transformation that will compute f(x,y) from spectrum function F(u,v) (that is the same as saying function in frequency domain). So, inverse Fourier Transformation in two dimensions looks like:

$$f(x,y)=\int_{\infty}^{\infty}\int_{-\infty}^{\infty} F(u,v)e^{i2\pi(ux+vy)}\,du\,dv$$

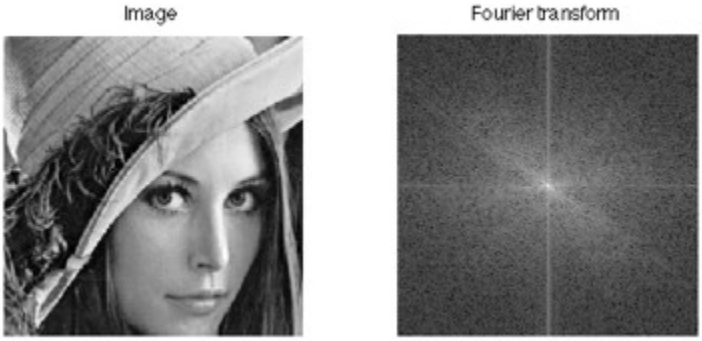
2.6 Importance and Use of Furier Transform

These computations are used in different fields in mathematics and different sciences. In physics Fourier transform is said to be a maker of the frequency function of the input signal. For example, forward Fourier Transformation on signal waves returns the frequency of that signal as a function F(x). If we take a piano as a visual example of frequency domain, we can notice these frequences:

The right part of the piano generates sound waves with less frequences, compared to the ones from the left part

In the sound processing, more daily example can be recording of the sound. This can be done by only recording the waves of the sound. In computer, the format of wave recording is .wav format. However, by using only the waves, device can record only the frequences of it, so less strorage is used and at most not much quality is lost. One of the formats that uses this technology of storing sound is mp3 (uses other tricks also). For playing such file, video player should use inverse Fourier Transform to get the actual wave signal, so that human ear can interpret the sound.

An image is another type of signal that can be Fourier transformed. Image is actually in two dimension and actually is matrix of pixels. When doing Fourier transformation, we get the frequency of the image and very incomprehencive data for human eye.



Low frequences in the Fourier transform (frequency domain) image have high values (in the center the brightness represent high values). That means, pixels' values are not changing rapidly in small distances, which is equal to say that considering their neoghboors, pixels are almost the same. At the same time, high frequences represent the rapid changes over small distances of the image. These values are not important in global, so for example .jpeg image encoding uses this technique (together with other techniques) and discards the high frequences of the image that result not so big changes in the image.

Chapter 3

Filterings

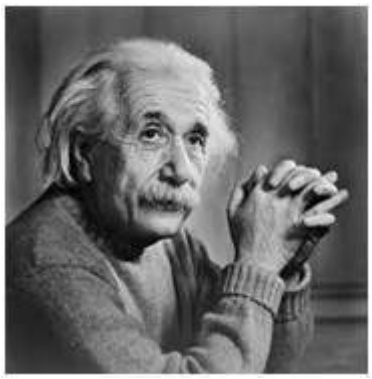
Filtering data in a spreadsheet means to set conditions so that only certain data is displayed. There are enourmous collection of data filtering in different fields, but we will focus on ones that we use in our project.

3.1 Low-Pass and High-Pass Filtering

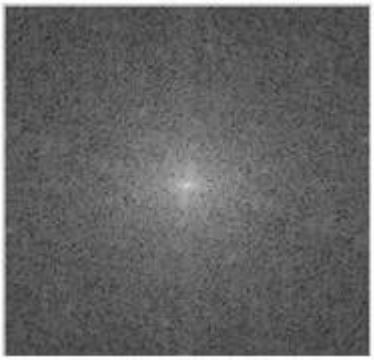
Above mentioned technique that discardes high frequences is called ***low-pass filtering***. Recalling the technique, we can just cancel out the high frequences or simply saying delete (or add some black layer) outer part of the forward Fourier Transformed image, so that in the inverse Fourier Transformed image will not show so small details of the picture and will save lots of data. Of course, the quality of the image reduces, but the data consuptioin at the same time reduces considerably. For example, the above mentioned image after doing Low-Pass filtering will have this look while zooming:

It is obvious that the quality is reduced, however in some cases data size is more important than the quality of the image.

Other example of such transformation is seen in the following picture:

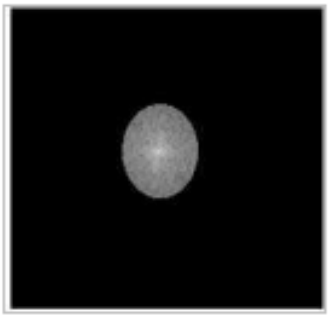


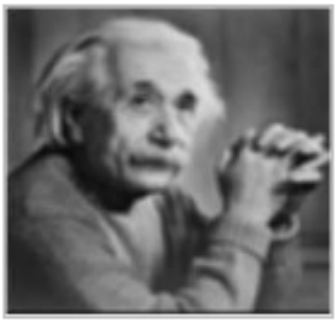
Original Picture



Frequency domain representation

after removing the outer part of the Frequency Domain picture, the resulting image in time domain will be little bit blurred and will lose some details:





In the left it can be seen Low-Pass filtering represented in Frequency domain, while in the right is the output image in Time domain

High Pass Filtering is the visa versa operation of Low-Pass Filter. It consideres low freqeuncy part as not important and considers only the outer part of the frequency domain image. That means after doing inverse Fourier Transform, we will get only the pixels that change their values rapidly in small distances of the picture. An example can be viewed next:

Original Picture

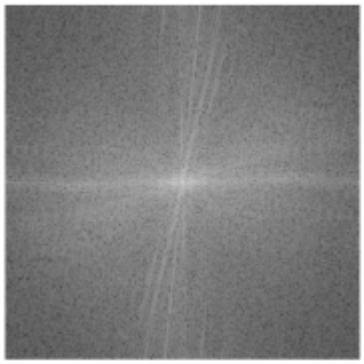
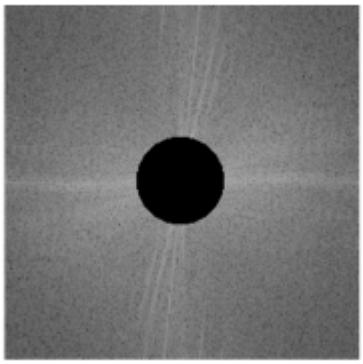


Image in frequency domain

From the left is the original image and from the right the Fourier Transformed image or the image representation in the frequency domain. By doing the same trick here, but taking now the outer part of the image, we can see only the pixels that change rapidly over small distances:



Applying High-Pass filtering

Resulting Image in time domain

The high-pass filtering is a useful tool that can help to determine the borders of the objects in the image. In this case it is obvious that it highlighted for our eye the borders of the windows, building and other objects in the image. The base of our edge detection program is this filtering that takes into consideration only the high frequency data.

3.2 Butterworth BandPass Filter

Butterworth BandPass Filter is another filter that we use in out project. The goal of this filter is to create as smooth pictures as possible. Mathematical view of this filter is following:

$$H_{BP}(u,v) = H_{LP}(u,v) * H_{HP}(u,v)$$

where

$$H_{HP} = 1 - \frac{1}{1 + \left[D(u,v) / D_H \right]^{2n}}$$

$$H_{LP} = 1 - \frac{1}{1 + \left[D(u,v) / D_L \right]^{2n}}$$

In these equations D_H represents cut off frequency of high-pass filter, D_L is the cut off frequency of low-pass filter, D(u,v) distance from origin and n is the order of the filters. If we make n bigger, the output becomes smoother and reducing it makes the output look more like original picture. For example, the fourth order Butterworth Filter will look like:

Chapter 4
Algorithm

% Reads the image from the computer as an input image. We use imread function that is one of the most widely used methods of reading image from graphic file.

input_image = imread('test.jpg');

%We take height and width of the input picture by using size function. In usual, size funtion has three values in x, y and z directions. As our image is 2D, we only can take height and width of the image. For that, we use the second parameter for size() function, specifying the direction of the size value (x, y or z direction). Value 1 is reffered to the height of the image, 2 is reffered to the width, whereas 3 is depth or z axis direction.

height = size(input_image,1);

width = size (input_image, 2);

%here we create 2D-grid coordinates for height and width of the image. meshgrid function can also create 3D-grid coordinates, but 3D is not in our scope of project.

[x,y] = meshgrid (-floor(width/2):floor((width-1)/2), -floor(height/2):floor((height-1)/2));

%creating rectangle with the same size as input image. After which we create circle with radius 30 that will be used for High-Pass filtering.

z = sqrt(x.^2 + y.^2);

c = z>30;

%We also use rgb2gray(RGB) function that converts the truecolor image RGB to grayscale image. It does it job by eliminating hue and saturation information of the input image.

```
inputImage_gray = rgb2gray(input_image);
```

%medfilt2() function performs median filtering of the image in two dimensions. In the output each pixel contains median value in 3 by 3 neighborhood around the viewed pixel. This is done in order to remove the noise of the image and have as much clear input as possible.

```
a = medfilt2(inputImage_gray);
```

%Here we use two Fourier Transform operations. The first one is fft2() that computes the value of the image in frequency domain. It uses FFT (Fast Fourier Transformation) algorithm. After which, the output is passed to fftshift() function which shifts the low frequency values to the center and high frequency to the edges of the spectrum. After this operations we will have image representation in the frequency domain with low frequences in the center.

```
shiftedImage = fftshift(fft2(a));
```

%Here we discard the center of the Fourier Transformed image, so that low frequency values of the input image will not be considered afterwards. In other words here we use High-Pass filtering.

```
HPShiftedImage = shiftedImage.*c;
```

%Now, we apply the next filter, butterworth filter. The implementation of the filter is defined in other file. We passed constant values to the filter 110 and 3, but they can be changed regarding to the input image.

```
hb = butter_filter(HPShiftedImage,110,3);
```

%Here we get the result of the butterworth effect by multiplying the result of the previous step with High-Pass filtered image. This will result getting the final output in frequency domain. For viewing the image we should firstly change it to time domain, so human eye can interprete objects.

```
butterfly_effected = HPShiftedImage.*hb;

%By ifft2() function we do inverse Fast Fourier Transform and get image in the time domain.
changedImage = ifft2(butterfly_effected);

%But there is still a small step to do. After doing Fourier Transform in the beginning of the file, we also concentrated all low frequences in the center of the image, while high freqeunces close to the edges. Now we should do the reverse step, so the image can be interpreted by human eye. The function ifftget() is also defined and implemented in different file.
k = ifftget(changedImage);

%And finally, we have the final image, so we can show it. The last is done by imshow(I) function that displays the argument I, if it is grayscale, RGB or binary image.
figure, imshow(k);
```

Chapter 5

Results and Conclusion

The aim of this project was edge detection using some operations Fourier Transform. Edge Detection is process, in which boundaries of the image objects are detected and displayed. By using high-pass and butterworth filters on the Fourier Transformation of the image, we managed to detect edges and show them to the user. The problem was partially solved, as there were some errors in the resulting image. Mainly, by taking some complex images as input, sometimes, the output is not so good.

In the future it will be very useful to analyse some of the filters used in the frequency domain. Using some filters may improve the results and if solution will be reached that is close to perfect one, certainly Fourier Transformation method may become the best method of finding edges. The reason for the last is its inexpesiveness and fastness of the algorithm.

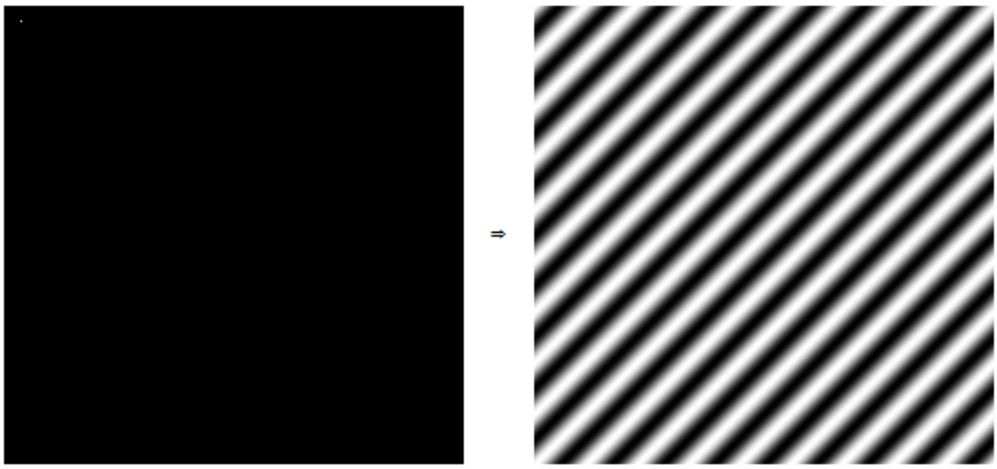
Chapter 6

Appendices

6.1 Time and Frequency domain

In the world, different objects can be represented in different ways. In mathematics and different sciences also, there are different ways of representation. In our project paper we mentioned words “frequency domain” and “time domain” expressions. Also, in some places we noted that saying Fourier Transformed image is the same as to say image in frequency domain. Firstly, let’s define what is ***time domain***. It is the analysis of different signals, mathematical functions or series in respect to time. In other words, values of the functions or signals are changing in respect to time changes.

Images are represented in time domain usually. This is the way of representation that human eye can interpret. We can distinguish objects, detect their borders, but in order to do some difficult operations it is often suitable to change into ***frequency domain*** representation. Frequency domain in mathematics, electronics and statistics is reffered to an analysis of mathematical functions or signals in respect to frequency. For understanding the difference and degree of difficulty in these two domains, we can just consider a small example. In frequency domain a simple dot becomes a complicated image with parralel lines of different color:



It is important to note also that all these transformations from frequency domain to time domain or visa versa can be done with the help of inverse and forward Fourier Transformations respectively.

6.2 Median Filter

Median Filter is filter used in order to reduce noise level of the image. The algorithm of the filter is not complicated. Median Filter takes a “window” of 3x3 pixels and computes the median of the values. After computation, it puts the value in the center of the “window”, resulting the change of the original picture. It runs through all pixels of the input image and does the same steps. Instead of “window” sometimes are used more complex patterns, such as “box” or “cross” patterns.

Bibliography

http://www.testunlimited.com/pdf/an/5988-6765EN.pdf
http://mathworld.wolfram.com/FourierSeries.html
http://mathworld.wolfram.com/KroneckerDelta.html
http://tutorial.math.lamar.edu/Classes/DE/FourierSeries.aspx
https://www.astro.umd.edu/~lgm/ASTR410/ft_ref2.pdf
http://www.erzetich-audio.com/knowledgebase-05-time-vs-frequency
http://www.personal.soton.ac.uk/jav/soton/HELM/workbooks/workbook_23/23_2_rep_periodic_fns_by_fourier_srs.pdf
https://math.stackexchange.com/questions/584113/fourier-series-coefficients-proof
http://mathworld.wolfram.com/FourierTransform.html
http://mathworld.wolfram.com/FastFourierTransform.html
http://www.askamathematician.com/2012/09/q-what-is-a-fourier-transform-what-is-it-used-for/
https://www.tutorialspoint.com/dip/high_pass_vs_low_pass_filters.htm
https://www.researchgate.net/post/What_is_meaning_of_high_frequency_and_low_frequency_in_image_processing
http://www.robots.ox.ac.uk/~az/lectures/ia/lect2.pdf
http://fourier.eng.hmc.edu/e101/lectures/Image_Processing/node6.html
https://www.scribd.com/doc/51981950/Frequency-Domain-Bandpass-Filtering-for-Image-Processing