1 Introduction

1.1 Structure of the report

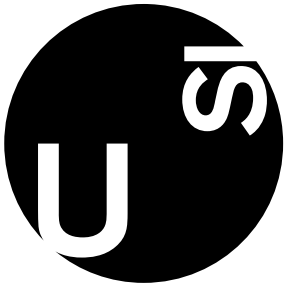
We will first analyze the approach used in the reference article(cite) (Section Methodology). In the same section we will also propose our approach to the rotation which involves a general explanation of the Fourier Transform and it's properties, how to design a filter,

2 Methodology

3 Implementation

4 Results

5 Conclusion



Università
della
Svizzera
italiana

Faculty
of
Informatics

Bachelor Thesis

June 23, 2023

Rotation of multi-dimensional signals with spectrally accurate schemes

Dylan Reid Ramelli

Abstract

The idea to take on this project began on the assumption that rotating an image might be a considered a trivial task. It became clear that rotation is a multi faceted operation that involves different steps. First we must acquire the image and to do that we must be able to sense it. Just like a robot arm in an assembly line needs to know which parts of the car go where, we must be able to take a discrete sample from a continuous signal. Once the image is acquired we need tools that are as accurate as possible and immune to disturbances to be able to perform accurate rotations on the image. This project, which is based on a research paper on rotating images using a convolution based interpolation [2] will try to explore a similar way to rotate an image while maintaining a high degree of accuracy and performance. To do this the Fourier Transform's property of shift and of convolution of a signal with a filter kernel will be taken advantage of. What will also be shown is how the filter kernel was created in the frequency domain and the problems that the choice of filter might pose to the result of the shift.

Advisor
Prof Rolf Krause
Co-Advisors
Dr Diego Rossinelli, Dr Patrick Zulian

Advisor's approval (Prof Rolf Krause): Date:

Contents

6 Introduction, Motivation

Mini scaletta:

- Concetto generale
- Motivazione sul perche ce bisogno di rotazioni.
- Vantaggio della rotazione spettrale
- Quali sono i contributi che sono stati fatti in questo lavoro
- spigazione di un paragrafo su cosa ogni capitolo si parla

In many scientific fields, rotation of multi-dimensional signals plays an important role in digital image processing. For instance, it is used in selective plane illumination microscopy(SPIM)

What is the contribution in the project: How to create a filter, show implementation and maybe study the numerical performance Accurate rotation requires the use of some fundamental signal processing concepts: Fourier domain and convolution of a signal with a filter. Filters are necessary in this work for their simplicity and ease of computational complexity. It will be shown how to create a filter to use as kernel for convolution with the original signal to rotate by fractional amounts. Something that is quite difficult to do accurately with traditional methods of interpolation for example.

This work could become very useful in many applications that rely on accurate signal rotations such as Data Augmentation for example. In this case we want to increase the coverage of a certain dataset. In Convoluted Neural Networks we can augment the dataset by rotating the input images by some random rotation and then feeding them to the model to improve it without having to collect more data externally. Another application of this technique can be seen in Data Visualization for multi-modal imaging. Images are taken at different times or in different ways, such as in a PET-CT scan and rotation can enable us to re-align the elements present in these images. In this case we can either acquire images at different times and then combine them together so that we can manipulate the images by rotation for example to align them correctly. Go into more detail.. ask Diego.

The goal of this project is to process high quality multi-dimensional signals, by rotating them using techniques such as the Fourier Transform and filter convolution.

7 State of the art

For certain applications such as... it is required to be able to rotate images while maintaining the highest quality possible. To achieve satisfying results the general approach is to use bi-linear and nearest neighbor interpolation.

Include images from Diego of blood vessels in the heart iI think

8 Methodology

In this project we will use the same method to rotate an image as in the original paper [2] except we will be creating our own convolution kernel in the frequency domain by using a cosine function as base. Defining the function in the Fourier space allows us to fine tune the filter to have a better control over the smoothing effect that will occur when we shift by fractional amounts. Since the kernel will be small we will also just use a normal Inverse Fourier Transform instead of a IFFT.

In the article that this project takes inspiration from it is shown that the traditional rotation matrix

$$R(\theta) = \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix}$$

can be factorized as three matrices each of which represents a shearing of the image in a cardinal direction.

$$R(\theta) = \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix} = \begin{pmatrix} 1 & -tan\theta/2 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 \\ sin\theta & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & -tan\theta/2 \\ 0 & 1 \end{pmatrix}$$

The first matrix shears the image in the x direction by $\Delta_x = -y \cdot tan(\theta/2)$, the second matrix shears the image in the y direction by $\Delta_y = x \cdot sin(\theta)$ and the last matrix shears the image again in the x direction by Δ_x. In our case we can represent the image as a one dimensional array ordered row-wise.

8.1 Fourier Transform

In signal processing working in the frequency domain has a great advantage since a function expressed in this space can be reconstructed completely via an inverse process, with no loss of information.[3] For a function $g(x)$ to be represented in the frequency domain we must then make use of the Discrete Fourier transform which is defined as follows:

$$G_m = \sum_{k=0}^{N-1} g_k e^{-i\frac{2\pi}{N}km} \tag{1}$$

In this project the above equation will not be used since the filter that is going to be defined in one of the next chapters, will be created directly in the Fourier domain. Nevertheless it is useful to show the definition since it is the Inverse Discrete Fourier Transform which will be used to reconstruct the filter in the time domain.

$$g_n = \frac{1}{N}\sum_{k=0}^{N-1} G_k e^{i\frac{2\pi}{N}kn} \tag{2}$$

8.2 Fourier Transform with Shift Theorem

Here we derive the shift theorem for a discrete signal starting from the normal Fourier Transform with g_k being our 1D input array, N the number of samples and m the frequency:

$$G_m = \sum_{k=0}^{N-1} g_k[k] e^{-i\frac{2\pi}{N}km}$$

Now instead of $g_k[k]$, we want $g_k[k-\delta]$ where δ is the amount we want to shift. So the above equation can be rewritten as:

$$Z_m = \sum_{k=0}^{N-1} g_k[k-\delta] e^{-i\frac{2\pi}{N}km}$$

$$Z_m = \sum_{r=0-\delta}^{N-1-\delta} g_k[r] e^{-i\frac{2\pi}{N}km}, r = k-\delta$$

Since $r = k-\delta$ then $k = r+\delta$, and as such:

$$Z_m = \sum_{r=-\delta}^{N-1-\delta} g_k[r] e^{-i\frac{2\pi}{N}(r+\delta)m}$$

We can then separate the exponential:

$$Z_m = \sum_{r=-\delta}^{N-1-\delta} g_k[r] e^{-i\frac{2\pi}{N}rm} e^{-i\frac{2\pi}{N}\delta m}$$

And factor it out of the sum:

$$Z_m = e^{-i\frac{2\pi}{N}\delta m} \sum_{r=-\delta}^{N-1-\delta} g_k[r] e^{-i\frac{2\pi}{N}rm}$$

The sum now has exactly the same range as before:

$$Z_m = e^{-i\frac{2\pi}{N}\delta m} \sum_{k=0}^{N-1} g_k[k] e^{-i\frac{2\pi}{N}km}$$

$$Z_m = e^{-i\frac{2\pi}{N}\delta m} F_m = H_m \cdot G_m \tag{3}$$

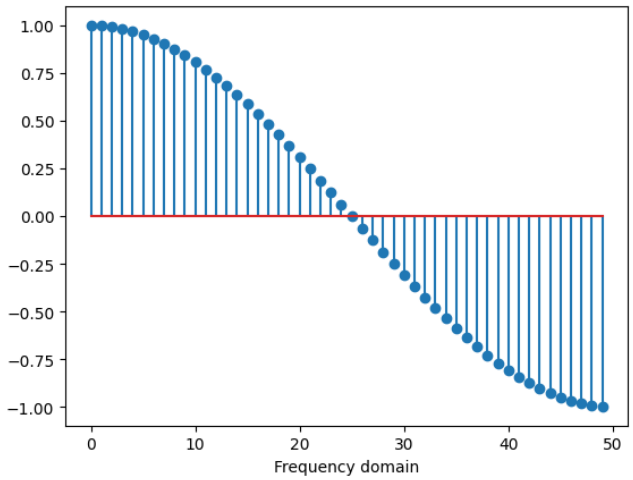


Figure 1. Shift kernel of size $n = 50$

8.3 Frequencies

The final equation from the previous section 3 works well for any kind of shift that we want to perform on our signal but for any fractional amount we encounter some problems with the use of the correct frequency indexes.

If we use the corresponding frequency indexes in 3 we will get a shift kernel:

And we can see that starting from $N/2$ we start getting negative values for the magnitude of each frequency. We do not want this(Explain)

To solve this we need to take into consideration the negative frequencies. Normally we multiply each sample by its corresponding phase, which is based on the frequency number m but since we need to take into consideration the negative frequencies we can define a function called wavenum that returns the correct frequency index to use in the phase shift:

$$wave\_n(m) = (m + \lfloor N/2 \rfloor) \bmod N - \lfloor N/2 \rfloor$$

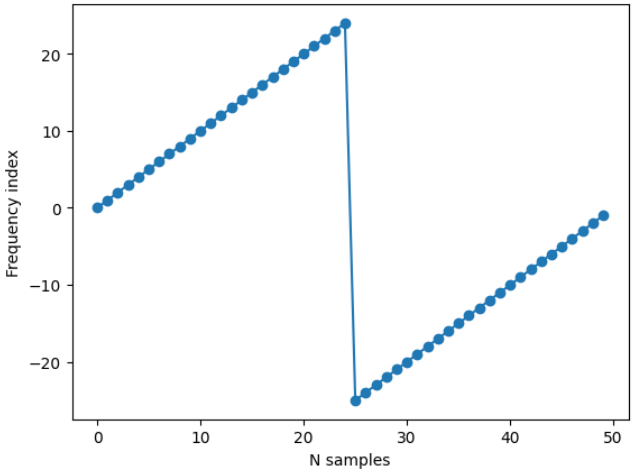


Figure 2. Wavenum function with $N = 50$

This will give us the correct shift kernel:

Modify original equation to use wavenum.

$$H_m = e^{-i\frac{2\pi}{N}\delta m}$$

$$Z_m = H_{\text{wave\_n}(m)} \cdot G_m \tag{4}$$

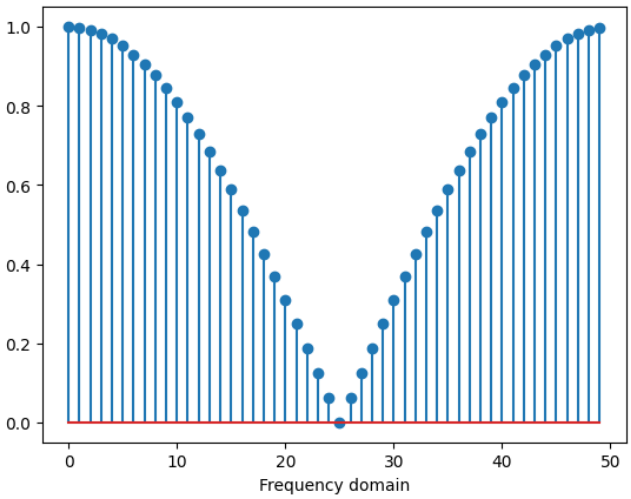


Figure 3. Shift kernel of size $n = 50$

8.4 Filter

To start off we create the kernel by defining it in the Frequency domain with the use of the Continuous Fourier transform on a sinusoid function.

$$F(\omega) = \int_{-M}^{M} \cos(\frac{\pi}{M}t)e^{-i\omega t}, dt \tag{5}$$

This equation allows us to define our CFT in an interval M, that we choose based on the fractional amount we want to shift, as to alleviate the oscillations and smoothing that occur with different sizes of M.

$$F(\omega) = \frac{2M^2\omega \sin(M\omega)}{M^2\omega^2 - \pi^2} \tag{6}$$

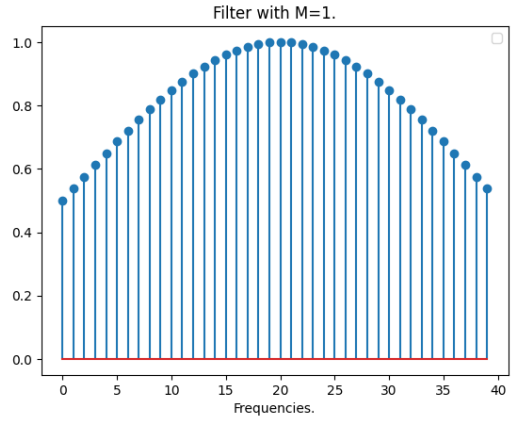
Using this function we can build the following filters:

In figure 4 we can see the general shape of the function for each M and the number of samples it will use for convolution. It does not however give us a good idea on how it is going to modify the signal. In the next figure we can see the same filters but in the time domain.

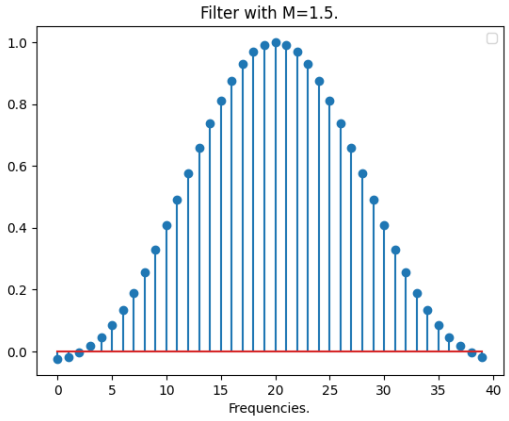
For better accuracy we generally choose our interval M to be small when we have a small fractional shift and big when we have a fractional shift that gets closer to 0.5. We could write a simple expression that defines M as:

$$M = \begin{cases} 1.5 & if\, 0.1 < x < 0.3 \\ 2 & if\, 0.3 < x < 0.5 \end{cases}$$

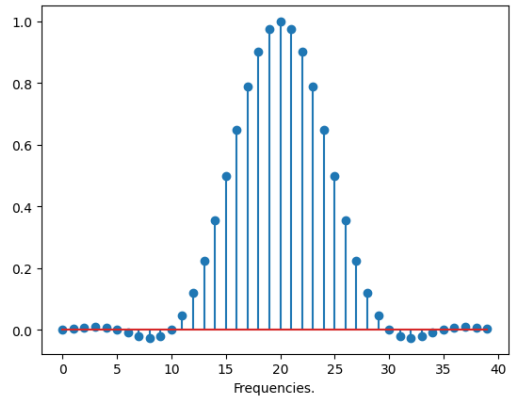
With some experimenting we found that the filter with $M = 3/2$ is a good compromise between smoothing the values and keeping them as accurate as possible.



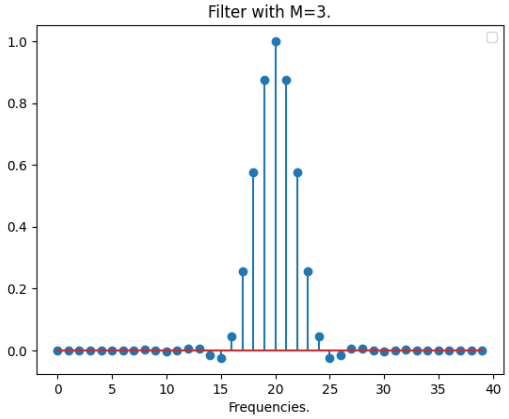
Filter with M = 1.



Filter with M = 2.



Filter with M = 2.



Filter with M = 3.

Figure 4. Filter

8.5 Creating a Filter in the Frequency domain

So first off we need to build our kernel. As an example we will create one of size 40 with $M = 2$ since we will be shifting by a fractional amount. To create the kernel then we follow equation 6, while also paying attention to evaluating the limit when $\omega = \frac{\pi}{M}$ or $\omega = -\frac{\pi}{M}$. This gives us:

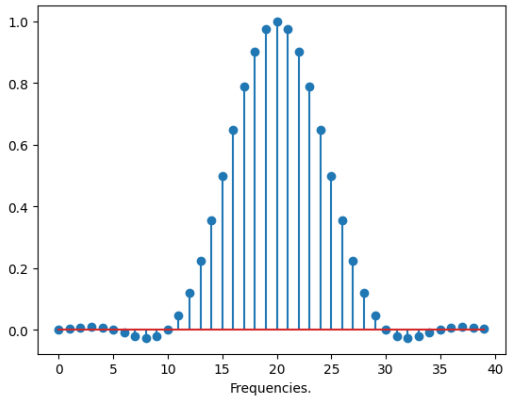


Figure 5. Filter with $M = 2$

Now let us say that the fractional amount to shift is 0.25. To implement this we can use a simple for loop that multiplies each frequency value of the filter by it's corresponding shift 4.

Afterwards we implemented a simple version of the inverse Fourier transform, which give us our signal in the time domain. Before returning the result however we need to perform the equivalent of numpy's fftshift function to have the filter values centered. This whole operation can be done in serial and with no real optimization of the code since the kernel size will always be much smaller than the input signal and as such will no be computationally expensive.

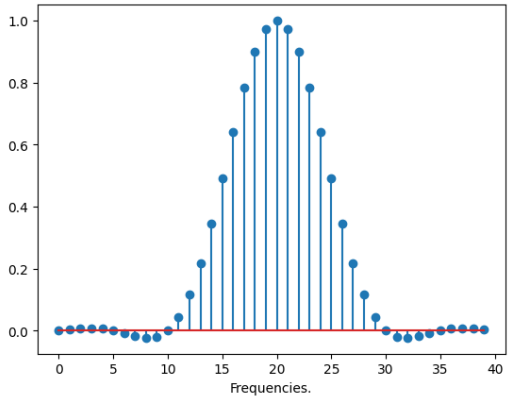


Figure 6. Filter with $M = 2$ after shift of 0.25

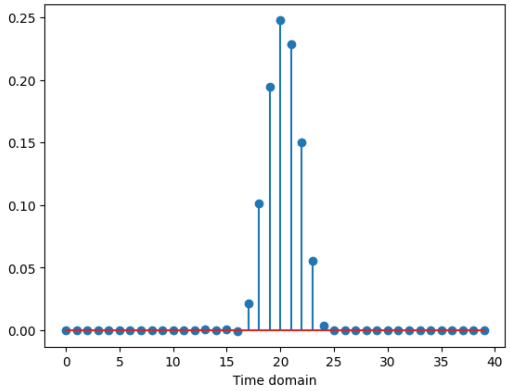


Figure 7. Filter with $M = 2$ in spatial domain.

Here is some pseudo-code that explain what is being done to our original filter:

Algorithm 1 Filter shift

Write convolution algorithm defined with compiler explorer

9 Implementation and Project Design

- Link github page and wxaplin hwo to run an example
- creazione filtro e spiegare perche IFT invece di ifft
- perche direct convolution invece di fft
- code generation at compile time with python script
- piccola spiegazione su SIMD e metodo di convoluzione per facilitare la vettorizzazione dell'assembly code

Since the premise of this project is based on the fact that a rotation can be performed with three 1D operations, the bottom-up approach was a no-brainer. We started with the fundamentals of signal processing by exploring the Fourier Transform and Shift Theorem to build our shift kernel. We used Jupyter Notebook extensively to test our method and quickly plot the results, while also documenting everything so that it would be easy to write this report. Since we used the bottom-up approach we were able to re-use parts of the code to test some other aspect of the shift. We started with simple shifts of a 1D signal by an integer value which proved to be quite simple and then we tried to shift the same signal by a fractional amount. This is where some problems arose because some care is needed while shifting each frequency by the correct phase shift. The complete the project the tools that were used were mainly: MakeFile, CMake, Jupyter Notebook and some built-in libraries of C and Python like numpy and matplotlib.

9.1 Libraries

Numpy and Matplotlib were a huge help in plotting the results of our initial tests of the Fourier Transform and of the final results. These libraries were also useful for when we started writing code in C. Where we would output the binary result to a file and then read it and plot the result in python. There are a few plotting libraries for C but we felt that this was just the faster/easier solution. The whole project started out as a CMake and this was chosen mainly because I was already familiar with this tool and was able to quickly setup an environment that would build the executable. However towards the end we decided to just use a plain Makefile to keep everything nice and simple.

9.2 Convolution Implementation

To keep our one dimensional convolution as simple and as efficient as possible we analyzed it using a Web tool called "Compiler Explorer"[1]. This tool allows us to see which assembly code is being used to perform the convolution operation and choose which implementation should run faster on certain compilers and CPU's.

9.3 Integer Shift

Now that the fractional part of the shift has been taken care of we need to shift the signal by the integer value. First of all it should be clarified that if the shift does not have a fractional part then we skip the convolution with the filter entirely. To shift a signal by an integer amount we used the C function *memcpy* to copy the values of the original array to a resulting array but at a different position, which is defined by the integer shift. We chose *memcpy* instead of just moving the values inside the same array with *memmove* since *memcpy* is designed to be the fastest library routine to copy data from memory to memory.

10 Results

10.1 Results with 1D signals.

One dimensional signals are the starting point of our tests and as such we chose a pretty aggressive signal to test out. In this example we are using a starting signal of binary value of length $n = 100$ samples and it is defined as:

$$f(x) = \begin{cases} 1 & \text{if } x \leq \frac{n}{2} \\ 0 & \text{otherwise} \end{cases}$$

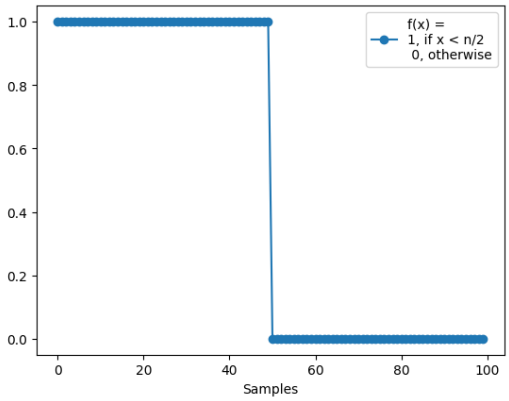
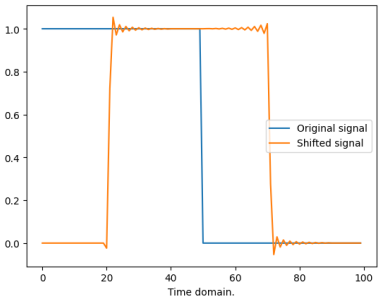
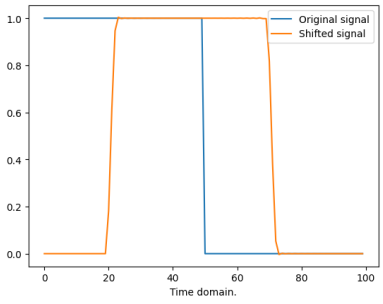


Figure 8. Original signal of size $n = 100$ samples.

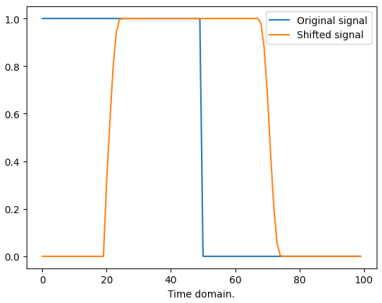
By taking this signal and shifting it by 20.25 we get a different result depending on the M chosen when creating the filter.



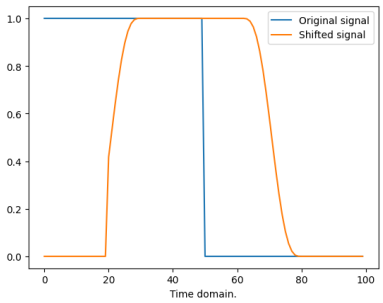
Filter with M = 1.



Filter with M = 3/2



Filter with M = 2.



Filter with M = 3.

Figure 9. Shift by 20.25

At first glance we can see that with $M = 1$ we have too many oscillations present in the resulting signal and with $M = 3$ the signal is smoothed out too much. We can see though in figure 10 that if we set $M = 1$ and the fractional value is small, for example 0.01 we have less oscillations and the signal is more similar to the original.

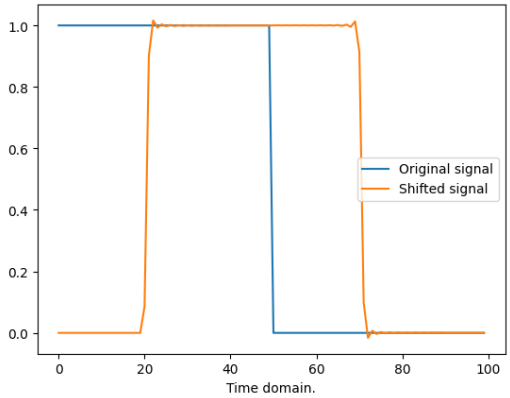
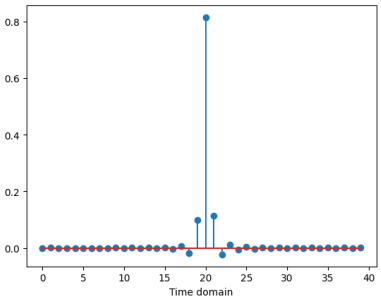
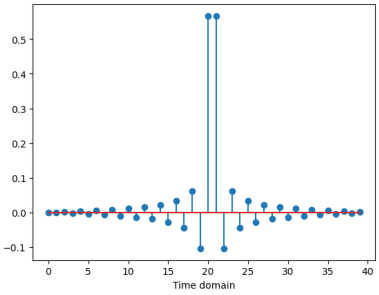


Figure 10. Shift by 20.01 with $M = 1$

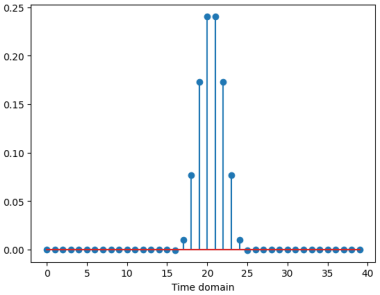
This result is attributed to the fact that the filter contains fewer oscillations as well. In fact if we take a look at the filter defined for $M = 1$ we can see that we have maximum oscillations exactly at 0.5. This is why for a fractional shift that gets closer to 0.5 it is necessary to use a bigger value for M such as $\frac{3}{2}$ or 2 to smooth out the oscillations.



Filter shifted by 0.01 with $M = 1$



Filter shifted by 0.5 with $M = 1$



Filter shifted by 0.5 with $M = 2$

Figure 11. Difference in oscillations with varying fractional shifts.

10.2 Results on 2D signals, i.e images

11 Conclusion

Summary of the report, What was achieved, limits of the method, any improvements,... One way to improve upon this work would be to implement the code with CUDA, create a function that chooses which M to use based on the fractional shift and the error.

References

[1] Matt Godbolt.

[2] IEEE Philippe Thévenaz Michel Unser, Senior Member and Leonid Yaroslavsky. Convolution-based interpolation for fast, high-quality rotation of images.

[3] Richard E. Woods Rafael C. Gonzalez. *Digital Image Processing*.