



Università  
della  
Svizzera  
italiana

Faculty  
of  
Informatics

Bachelor Thesis

June 11, 2023

# Spectrally Accurate Resampling of High Quality Rotated Images

Dylan Reid Ramelli

---

*Abstract*

---

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

---

Advisor's approval (Prof Rolf Krause):

Date:

# 1 Introduction, Motivation

For many years the amount of digital imaging data has been increasing exponentially (citation) and as such the need for acquiring this large data and process it with care has become a central point of focus. The goal of this project is to process high quality signals, in this case images, by rotating them using techniques such as the Fourier Transform, Shift Theorem, FIR filtering and CUDA parallel processing. The accurate rotation of images is very useful in many applications such as Data Augmentation for example. For Data augmentation we want to increase the coverage of a certain dataset. In Convolved 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.

"Explain 2D to 3D."

FT: Direct solver instead of iterative, takes in a lot of data.

## 2 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 bilinear and nearest neighbour interpolation.

Include images from Diego of blood vessels in the heart if I think

## 3 Methodology

To complete this project we made use of different tools such as, Python, Jupyter Notebook, C/C++ and CUDA.

In our project we will use a method of rotation which was described in the article titled "Convolution-based interpolation for fast, high-quality rotation of images"[?] where the rotation matrix

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

is factorized as three matrices each of which represents a shearing 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 again in the  $x$  direction by  $\Delta_x$ .

Given this definition of a 2D rotation we can perform the operation as a sequence of three 1D translations on the 1D input array that represents our image.

### 3.1 Frequencies of a signal

$$freq_n = \lfloor \frac{n}{2} \rfloor + 1 \quad (1)$$

Visual representation with  $n = 3$  of the discrete Fourier transform of a function  $g(x)$

$$G_m = \sum_{k=0}^{N-1} g(k) e^{-i \frac{2\pi}{N} km}$$

To shift a signal by a fractional amount it is imperative to use the correct frequency when doing so.

Given that for  $n$  samples we have  $\lfloor \frac{n}{2} \rfloor + 1$  distinct frequencies

Euler's Formula

$$e^{ix} = \cos(x) + i\sin(x) \quad (2)$$

Useful property:

$$\cos(w) = \frac{1}{2}(e^{iw} + e^{-iw}) \quad (3)$$

**Figure 1.** Unit circle with  $N = 3$ ,

### 3.2 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  $\delta$  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:

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

$$Z_m = e^{-i \frac{2\pi}{N} \delta m} G_m = H_m \cdot G_m \quad (4)$$

The above equation 4 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. Here is a graphical example that was found to be useful: With an  $N = 3$  number of samples of a signal we have exactly 2 distinct frequencies 1 but the total frequencies are actually 3 and they are  $[0 + i0, \frac{1}{2} + i \frac{\sqrt{3}}{2}, \frac{1}{2} - i \frac{\sqrt{3}}{2}]$ . The last two frequencies are one the complex conjugate of the other.

"Explain why".

To solve this we need to take into consideration the negative frequencies and in particular if the number of samples is odd or even. 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:

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

$$H_m = e^{-i \frac{2\pi}{N} \delta \text{wave\_n}(m)}$$

$$Z_m = H_{\text{wave\_n}(m)} \cdot G_m \quad (5)$$

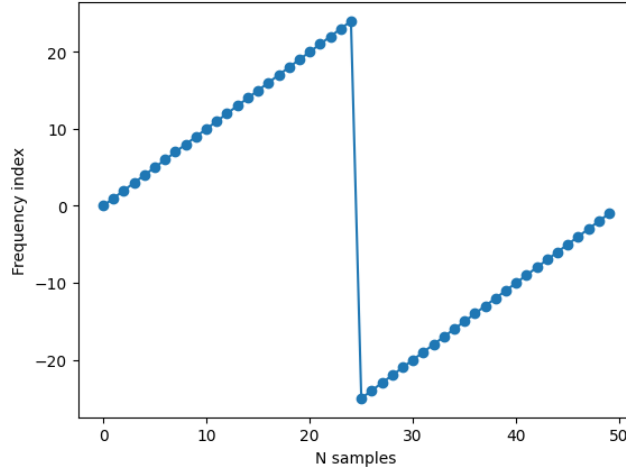


Figure 2. Wavenum function with  $N = 50$

## 4 Implementation and Project Design

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.

So first off we need to build our kernel that will have a shift applied to it according to the fractional part of the shift value. This can be done with simple for loop and in serial since the kernel size will be minimal and the only computationally intensive part would be the inverse Fourier Transform. To start of we create the kernel by defining it in the Frequency domain with the use of the Continuous Fourier transform on a sinusoid function.

$$F(\omega) = \int_{-M}^M \cos\left(\frac{\pi}{M}t\right) e^{-i\omega t} dt \quad (6)$$

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  $M$ .

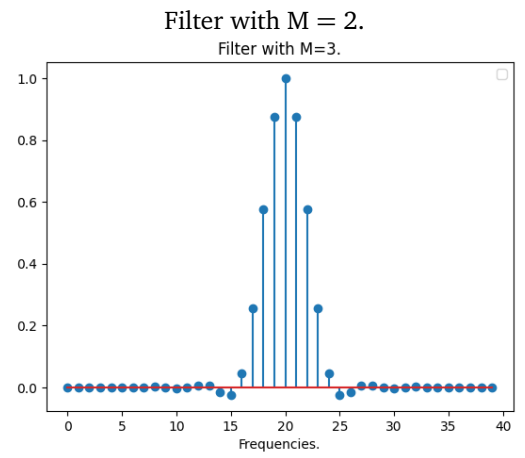
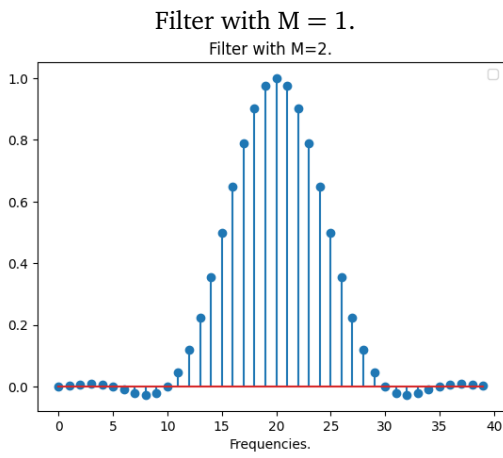
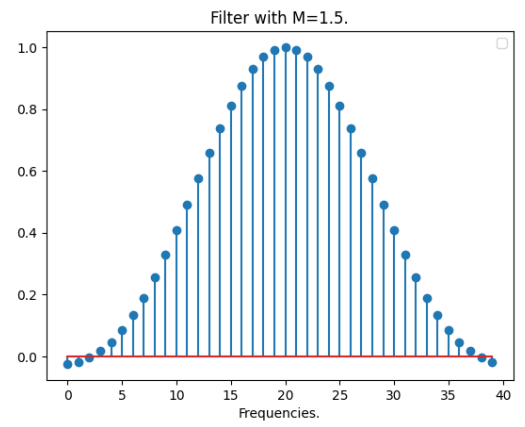
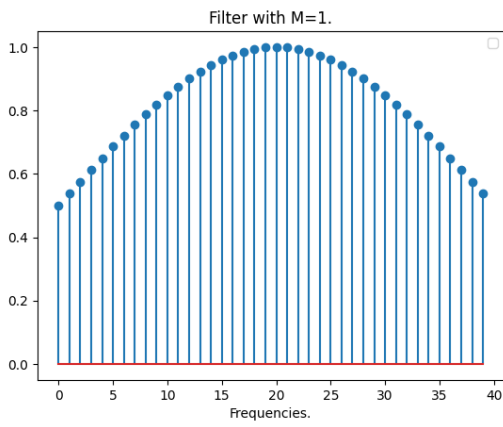
$$F(\omega) = \frac{2M^2\omega \sin(M\omega)}{M^2\omega^2 - \pi^2} \quad (7)$$

Using this function we can build the following filters:

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.

Now that we have our filters we can shift them by the fractional amount in the same way that we defined in equation 4.

To implement the one dimensional convolution we tried to optimize the algorithm by using godbolt. We came up with a solution that creates the required convolution function at compile time that will perform better than with standard for loops: Go into detail.



Filter with  $M = 2$ .

Filter with  $M = 3$ .

Figure 3. Filter