3-22-2012

Determining Angular Frequency from a video with a Generalized Fast Fourier Transform

Lindsay N. Smith

Determining angular frequency from video with a generalized fast Fourier transform

THESIS

Lindsay N. Smith, 2d Lt, USAF

AFIT/GAM/ENC/12-02

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government.

AFIT/GAM/ENC/12-02

Determining angular frequency from video with a generalized fast Fourier transform

THESIS

Presented to the Faculty

Department of Mathematics and Statistics

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science

Lindsay N. Smith, 2d Lt, USAF, BS

March 2012

AFIT/GAM/ENC/12-02

Determining angular frequency from video with a generalized fast Fourier transform

Lindsay N. Smith, 2d Lt, USAF, BS

Approved:

/signed/	16 February 2012
Dr. Matthew Fickus (Chairman)	Date
/signed/	16 February 2012
Dr. Mark Oxley (Member)	Date
/signed/	16 February 2012
Lt Col Brian McBee (Member)	Date

Abstract

Suppose we are given a video of a rotating object and suppose we want to determine the rate of rotation solely from the video itself and its known frame rate. In this thesis, we present a new mathematical operator called the Geometric Sum Transform (GST) that can help one determine the angular frequency of the object in question. The GST is a generalization of the discrete Fourier transform (DFT) and as such, the two transforms have much in common. However, whereas the DFT is applied to a sequence of scalars, the GST can be applied to a sequence of vectors. Most importantly, we show that the GST, like the DFT, can (1) be used to estimate frequency and (2) can be computed surprisingly quickly. Indeed, we provide a Fast Geometric Sum Transform (FGST) algorithm that computes the GST in $\mathcal{O}(N \log N)$ matrix-vector multiplications, where N is the number of images in the video sequence. This is a vast improvement over the $\mathcal{O}(N^2)$ such multiplications required for a direct computation of the GST. The remainder of this thesis is devoted to proving other properties of the GST and giving proof-of-concept numerical examples.

Acknowledgements

I dedicate this thesis to my husband, Michael Smith, who encouraged me to challenge myself by getting a master's degree in mathematics. His loving support allows me to accomplish more than I ever could achieve alone. I also thank my mother, Melissa Weidenhammer, and my mother-in-law, Jackie Smith, for their many prayers on my behalf. I appreciate my mentors Dr. Melody Massar and Dr. Analee Miranda, who both served as inspiration for academic accomplishment. I also extend deep appreciation to Dr. Mark Oxley and Lt Col Brian McBee for their contribution as thesis committee members. Most sincerely, I thank my academic advisor, Dr. Matthew Fickus, for his mathematical instruction and career guidance. His investment of time and teaching has helped me learn more about mathematics than I ever thought possible. Dr. Fickus has cultivated my academic understanding more than any other teacher.

Lindsay N. Smith, 2d Lt, USAF

Table of Contents

Page

Determining angular frequency from video with a generalized fast Fourier transform

I. Introduction

Consider a video sequence of image frames in which an object moves throughout the sequence, such as those presented in Figure 1. This figure is an image sequence of a rotating jet of plasma which is being ejected through a small propulsive device called a thruster. Suppose you want to know the rate in which this jet is rotating, but only the video sequence and its sampling rate is known. In this thesis we introduce an operator called the Geometric Sum Transform (GST) that, when applied to such a video sequence, allows us to estimate this rate of rotation. Specifically, this operator is useful in finding the rate of translation or alternatively, the rate of rotation of an object captured on video. It turns out that the GST is closely related to the Discrete Fourier Transform (DFT). As such, we begin here by recalling the basic properties of the DFT.

The DFT is an operator which is widely used in signal and image processing. Letting $N \in \mathbb{N}$, the DFT is defined over the following sequence space of all discrete-time, complex-valued N-periodic signals:

$$\ell(\mathbb{Z}_N, \mathbb{C}) := \{x : \mathbb{Z} \to \mathbb{C} \mid x[n+N] = x[n], \forall n \in \mathbb{Z}\}. \tag{1}$$

Specifically, for any $x \in \ell(\mathbb{Z}_N, \mathbb{C})$, define the DFT of x to be $\mathrm{DFT}(x) \in \ell(\mathbb{Z}_N, \mathbb{C})$,

$$\mathrm{DFT}(x)[n] := \sum_{n'=0}^{N-1} \mathrm{e}^{-\frac{2\pi i n n'}{N}} x[n']. \tag{2}$$

There are three main reasons why the DFT is popular: (1) it determines frequency, (2) there is a fast algorithm to compute it, and (3) it helps us understand

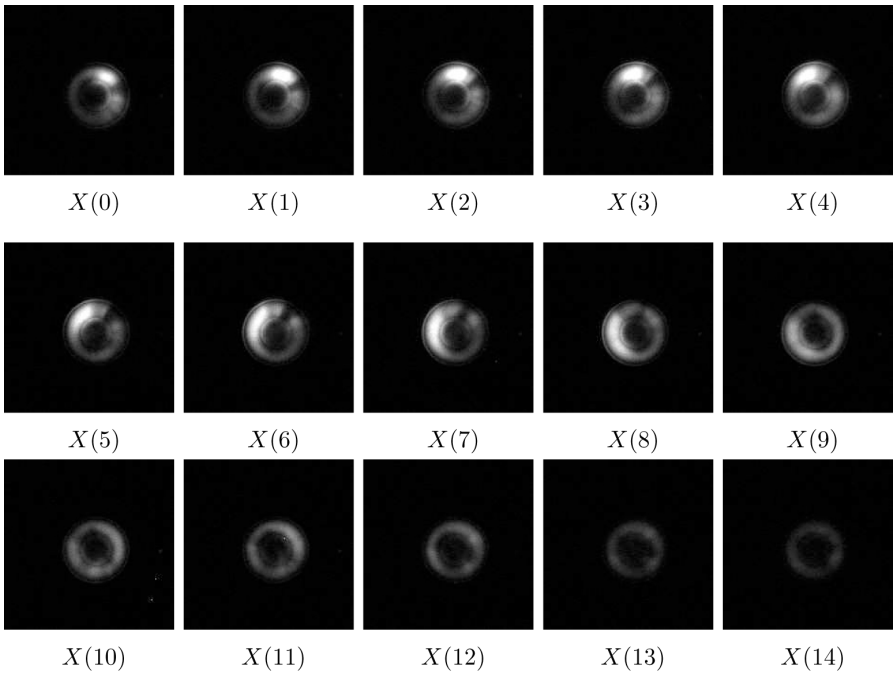


Figure 1: This 15-frame image sequence shows a rotating jet of plasma coming out of a 200-Watt Hall thruster, which is a small device intended to propel a satellite. Not only is the plasma ejected from the thruster rotating, the brightness of the plasma is also not constant. These images were collected by Liu as discussed in [8], but the true rate of rotation of the thruster was an unknown parameter. Applying the GST to a sufficiently well-sampled video sequence will provide a good estimate of the true rate of the plasma's rotation.

filters. The GST generalizes the first two of these properties to vector-valued functions. We introduce this new operator by generalizing the scalar field $\mathbb{C}$ to any Hilbert space $\mathbb{H}$, and also generalizing the root of unity $\mathrm{e}^{\frac{2\pi \mathrm{i}}{N}}$ in (2) to a unitary transformation $U : \mathbb{H} \to \mathbb{H}$ of finite order. In particular, whereas the DFT is defined over $\ell(\mathbb{Z}_N, \mathbb{C})$, as given in (1), the GST is an operator over the space of all discrete-time, *vector*-valued N-periodic sequences:

$$\ell(\mathbb{Z}_N, \mathbb{H}) := \{X : \mathbb{Z} \to \mathbb{H} \mid X[n+N] = X[n], \forall n \in \mathbb{Z}\}. \tag{3}$$

Specifically, the GST is an operator from this space into itself:

Definition 1. Let U be a unitary operator on $\mathbb{H}$ with the property that $U^N = \mathrm{I}$. The Geometric Sum Transform of any $X \in \ell(\mathbb{Z}_N, \mathbb{H})$ is $\mathrm{GST}_U(X) \in \ell(\mathbb{Z}_N, \mathbb{H})$ whose nth element is the vector:

$$\mathrm{GST}_U(X)[n] := \sum_{n'=0}^{N-1} U^{-nn'} X[n']. \tag{4}$$

Note that the *order* of a unitary operator U is defined as the least nonnegative integer K such that $U^K = \mathrm{I}$. In order to define the GST, the order of U must necessarily divide N.

Like the DFT, the GST (4) can be used to estimate frequency. While the DFT determines the frequency of a sequence of numbers by measuring how quickly they circle the origin in the complex plane, the GST estimates the frequency of a sequence of vectors with respect to the action of U. To be precise, if U is a rotation operator then $\mathrm{GST}_U(X)$ can be used to estimate the angular frequency of objects seen rotating in the video sequence X. Meanwhile, if U is a cyclic translation operator, then $\mathrm{GST}_U(X)$ can be used to estimate lateral velocity.

To see this, consider the toy example depicted in Figure 2. There, $N = 8$, the Hilbert space is $\mathbb{H} = L^2(\mathbb{R}^2)$, and the video sequence is thus a member of $\ell(\mathbb{Z}_8, L^2(\mathbb{R}^2))$. The sequence of images $X[n]$ is depicted in the first row, where $n = (0, \ldots, 7)$. It turns out the Mercedes-Benz shape in these images is rotating by a factor of $3(\frac{2\pi}{8})$ radians in each image frame, namely by $135\,^\circ$ counterclockwise from the previous frame. Suppose we want to use the GST to determine this fact. Since $N = 8$, we can only apply the GST with an operator U that satisfies $U^8 = I$. We choose U to be rotation by $\frac{2\pi}{8}$, namely $45\,^\circ$ in the counterclockwise direction.

To see what the GST does in this example, note that the $U^{-nn'}$ term in the GST definition (4) fixes n and then sums over different values of n'. Specifically, for a conjectured frequency n, the GST simulates taking a long exposure as the camera rotates backwards—in the clockwise direction—at a rate of n complete revolutions over the N image frames. To see this, the $X[0]$ column of Figure 2 is equivalent to holding the camera still, that is, rotation by 0 radians per image. Meanwhile, the $X[1]$

column corresponds to rotating the camera by $-\frac{2\pi}{8}$ radians per image. This process is continued for each possible frequency until the camera has been rotated by $-7(\frac{2\pi}{8})$ radians in the final column.

When the conjectured frequency n corresponds to the true rotational frequency, the two rates of rotation, both simulated and actual, cancel each other perfectly. This yields a coherent sum, resulting in a sharp image. For any other frequency, the corresponding sum is at least partially incoherent, resulting in a blurry image. One can thus determine the rate of rotation by computing the sharpness of the sum. In Figure 2, GST($X[3]$) appears to be the sharpest GST image; this confirms that the Mercedes-Benz shape indeed makes 3 complete revolutions over the course of this simulated video sequence.

Meanwhile, for the 100-frame thruster video sequence, 15 frames of which are depicted in Figure 1, the resulting GST consists of 100 images. Rather than plot each of these images and look for the one which is the sharpest, we can alternatively plot a sequence of their norms, as shown in Figure 3. Intuitively, the peak of this norm plot corresponds to the true rate of rotation, as we expect coherent sums to have larger norms than incoherent ones. Later in this thesis, we formally show that this is indeed the case.

From the above examples, we see that the GST is useful. However, the question now arises of whether or not it is practical to compute. Since n and n' both have N possible values in (4), the computation of the GST appears to require $\mathcal{O}(N^2)$ matrix-vector multiplications. When N is large, this can be prohibitively expensive. Fortunately, in this thesis we show that like the DFT, there exist a fast algorithm for computing the GST that reduces this computational burden to $\mathcal{O}(N \log N)$ matrix-vector multiplications.

1.1 Major contributions

The most significant contribution of this thesis is the Fast Geometric Sum Transform (FGST), a fast algorithm to compute the GST. Applied to a video sequence, the FGST significantly decreases the amount of matrix-vector multiplications required to compute the GST. Indeed, the FGST improves the GST computations required from $\mathcal{O}(N^2)$ to $\mathcal{O}(N \log N)$ matrix-vector multiplications. In this way, we see that the GST mimics two important facets of the DFT: (1) it can be used to estimate frequency, and (2) it can be computed surprisingly quickly. Additionally, there are other properties of the GST that are similar to those of the DFT, and are proven in this thesis. For example, the GST is a linear operator, has an adjoint, and can formally be shown to provide a good frequency estimate for a certain class of signals. Further, we perform a spectral analysis which shows that the GST is invertible under certain conditions, and moreover shows that the GST can indeed be regarded as a system of DFTs.

Several of these results are novel contributions to mathematical literature. In particular, as far as we know, our definition of the GST (4) is the first time anyone has generalized the DFT in a way that allows one to estimate rotational frequency from video. Moreover, the proof of the FGST algorithm (Theorem 5), though heavily inspired by the well-known FFT algorithms, is substantially more succinct and straightforward than all other similar decimation-in-frequency arguments that we could find in the existing literature. Finally, the spectral theory of Chapter V, which formally shows that the GST is, in fact, equivalent to computing a system of DFTs, is completely new.

1.2 Outline

Chapter II provides a summary of previous literature in this area of research. Since this thesis introduces the GST, we focus on background theory about the DFT, as these two transforms have many of the same properties. Many basic properties of the GST are provided in Chapter III. Then, Chapter IV includes a clear way to

define a generalized bit reversal, and most importantly, provides the FGST algorithm and the proof of its validity. A spectral analysis of the GST is given in Chapter V and we conclude in Chapter VI with more examples and numerical experimentation concerning the GST. The appendix includes MATLAB® code for the implementation of the FGST, so that this research can be more easily applied in future research.

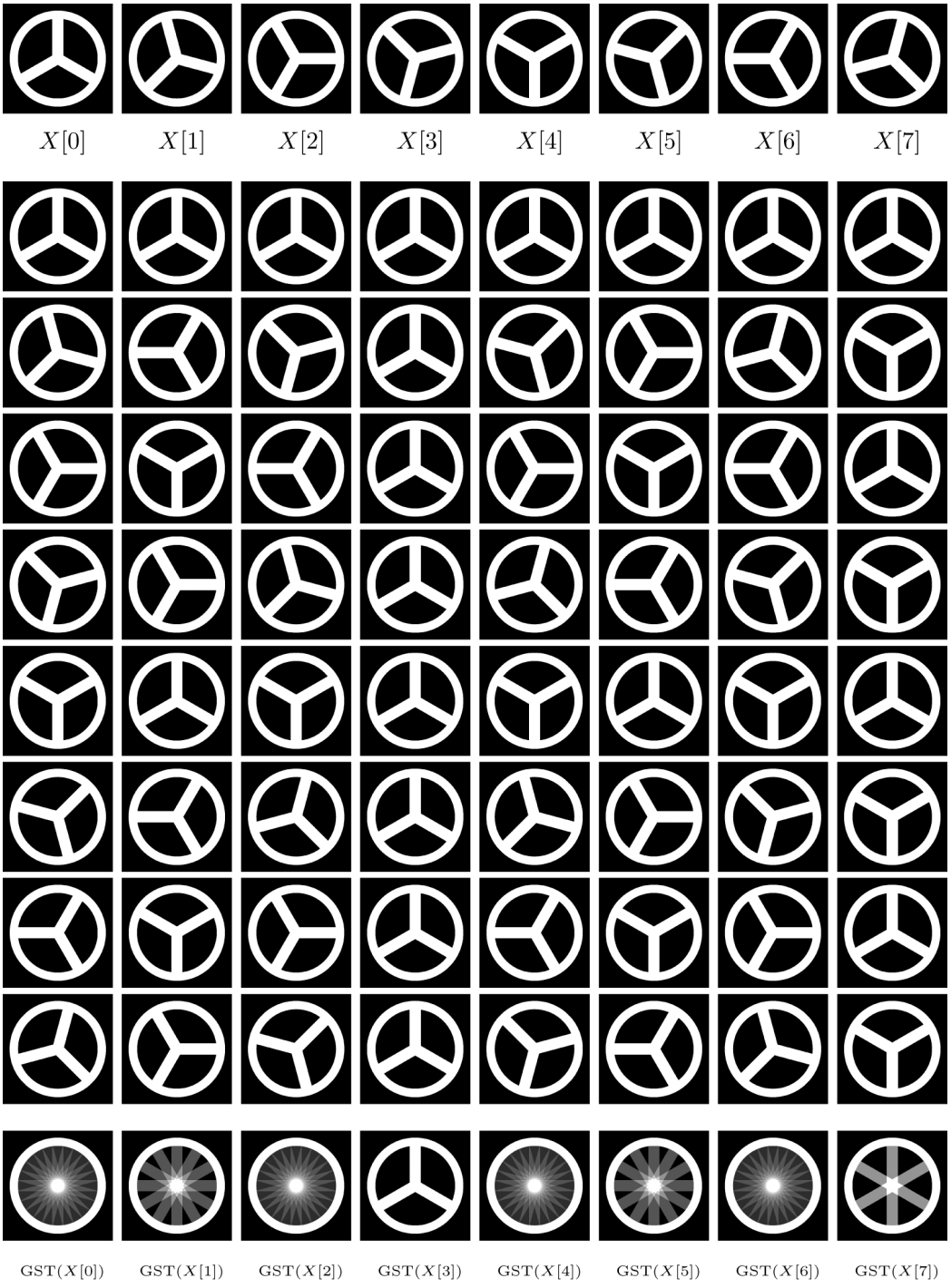


Figure 2: This is a GST rotation example of data length $N = 8$ applied to the Mercedes-Benz shape. The first row is the original X sequence. Each column under that initial row shows successively rotated images that are added together to form the final image in each column, namely the GST of a given $X[n]$.

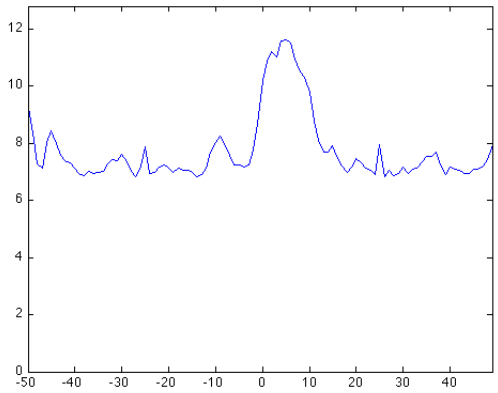


Figure 3: This plot shows the ℓ^∞ norm of the GST of a thruster video, several image frames of which are depicted in Figure 1. In this example, $N = 100$ and U is rotation by $\frac{2\pi}{100}$. Intuitively, it makes sense that the correct rate of rotation corresponds to the peak in the plot, since at the true rate of rotation, we expect the image frames to add coherently, in particular at their brightest point. Here, that peak occurs at $n = 8$, which once converted to standard units means the plasma is rotating at a rate of 80 Kilohertz.

II. Literature Review

The GST (4) is a new way to generalize the DFT (2). Indeed, it is a natural generalization because of the relationship between the DFT and the GST. Recall that the FFT is a fast algorithm to compute the DFT. In this thesis, we provide the FGST, a similarly fast algorithm for computing the GST. In this chapter, we focus on background theory about the DFT alone, since the GST and the DFT share many of the same properties, and no previous literature about a GST-like operator could be found.

Three properties of the DFT that are related to the work presented in this thesis are (1) the DFT has an inverse, (2) the DFT is a linear transform, and (3) the FFT assumes periodicity [11]. In addition, the FFT computes the finite Fourier transform of a length N data set in $\mathcal{O}(N \log N)$ operations [4]. Contrast this difference in efficiency to the $\mathcal{O}(N^2)$ operations required to compute the DFT directly.

Two main reasons the FFT has a computing advantage over directly computing the DFT is because it applies known periodicity of the sine and cosine functions and also exploits indices that contain a factor that is a power of two [4]. Cooley and Tukey [3] showed how to compute an FFT whose length is a highly composite number N, that is, an integer with many prime factors. Their research showed that an FFT of size $N = p_1 p_2 \dots p_m$ takes $N(p_1 + p_2 + \dots + p_m)$ operations.

Often, the first step in many FFT algorithms of length N is to reverse the order of the bit representation of the indices $n = (0, \dots, N-1)$. Such bit reversal can be found by using Buneman's algorithm, as presented by Walker in [14]. For example, the number 5 has a bit representation of 0101, which when reversed becomes 1010 and corresponds to the number:

$$\beta(5) = 1(2^3) + 0(2^2) + 1(2^1) + 0(2^0) = 10.$$

This description of bit expansion and this example are described in more detail in [1]. In Chapter IV, we generalize this notion of bit reversal to arbitrary n-ary expansion.

Although all FFT algorithms incorporate bit reversal, each algorithm can be computed using either *decimation in time* or *decimation in frequency*. The 1965 Cooley-Tukey algorithm is an example of decimation in time, that is, where each computation group gets bigger than the previous one [11]. On the other hand, the 1966 Gentleman-Sande algorithm [5] is an example of decimation in frequency, that is, where each computation group is smaller than the previous level. These two different computation methods are best visualized with well-known FFT "butterfly diagrams," as shown in [1], [2], and [14].

According to Heideman [6], in 1805 preceding Fourier's introduction of sinusoidal expansion, Gauss created an algorithm that closely resembles the FFT, but his algorithm did not quantify computational complexity to the $N \log N$ operations FFT algorithm we now use. Gauss's algorithm computed the coefficients of a finite Fourier series, and was similar to a decimation in frequency algorithm that has been modified for a real data sequence [6]. Note, the FGST algorithm we present in Chapter IV is also an example of decimation in frequency, where the initial input length value is factored and then computation groups are continually broken down into smaller groups.

Now, more than two hundred years after Gauss's work, there have been many different FFT algorithms created. In [13], Van Loan presents a summary of many popular FFT algorithms, and categorizes them based on the size of each framework. That is, the frameworks are distinguished by how an FFT of length N is broken into computation groups. For instance, there is a class of algorithms specifically created for computing the FFT of a length $N = 2^k$ signal for $k \in \mathbb{N}$. Several examples of these algorithms are the Cooley-Tukey algorithm, the Stockham algorithm, and the Pease algorithm, all presented in [13].

Variations of each algorithm are also presented in [13], including ways of splitting the computational groups, such as Radix-2, Split-Radix, and Mixed-Radix. A Radix-

2 framework is applied to a signal of length $N = 2^k$, but a Mixed-Radix framework is applied to a signal of length $N = p_1 \cdots p_k$ where each p_i is a factor of N.

Most of the FFT algorithms we have mentioned up to this point are similar to the FGST algorithm we introduce in Chapter IV. However, there are some FFT algorithms that exactly follow the approach we pursue in this thesis. For example, in 1968, Rader created a way to compute an FFT of prime length by treating the signal like a cyclic convolution [10]. Then, in 1999, Mohlenkamp found a fast transform for spherical harmonics, that is, problems where the Fourier exponential functions are generalized to functions over the sphere S^2 in $\mathbb{R}^3$ [9]. More recently in 2006, Rokhlin and Tygert presented a fast algorithm for forward and inverse spherical harmonic transforms [12].

III. Basic properties of the GST

In this chapter we introduce and prove several of the basic properties of the GST (4).

3.1 $\ell(\mathbb{Z}_{\mathbf{N}}, \mathbb{H})$ *is a Hilbert space*

Recall the GST (4) is defined over the space $\ell(\mathbb{Z}_N, \mathbb{H})$ defined in (3). We now verify that this space is indeed a Hilbert space with inner product:

$$\langle X, Y \rangle_{\ell(\mathbb{Z}_N, \mathbb{H})} := \sum_{n=0}^{N-1} \langle X[n], Y[n] \rangle_{\mathbb{H}}. \tag{5}$$

To verify this, we just need to show that $\langle X, Y \rangle_{\ell(\mathbb{Z}_N, \mathbb{H})}$ satisfies the four properties of an inner product space. First, for any $X, Y, Z \in \ell(\mathbb{Z}_N, \mathbb{H})$:

$$\begin{aligned}
\langle X + Y, Z \rangle_{\ell(\mathbb{Z}_N, \mathbb{H})} &= \sum_{n=0}^{N-1} \langle X[n] + Y[n], Z[n] \rangle_{\mathbb{H}} \\
&= \sum_{n=0}^{N-1} (\langle X[n], Z[n] \rangle_{\mathbb{H}} + \langle Y[n], Z[n] \rangle_{\mathbb{H}}) \\
&= \sum_{n=0}^{N-1} \langle X[n], Z[n] \rangle_{\mathbb{H}} + \sum_{n=0}^{N-1} \langle Y[n], Z[n] \rangle_{\mathbb{H}} \\
&= \langle X, Z \rangle_{\ell(\mathbb{Z}_N, \mathbb{H})} + \langle Y, Z \rangle_{\ell(\mathbb{Z}_N, \mathbb{H})}.
\end{aligned}$$

Second, for any $\alpha \in \mathbb{C}$ and any $X, Y \in \ell(\mathbb{Z}_N, \mathbb{H})$:

$$\langle X, \alpha Y \rangle_{\ell(\mathbb{Z}_N, \mathbb{H})} = \sum_{n=0}^{N-1} \langle X[n], \alpha Y[n] \rangle_{\mathbb{H}} = \sum_{n=0}^{N-1} \langle X[n], \alpha Y[n] \rangle_{\mathbb{H}} = \alpha \langle X, Y \rangle_{\ell(\mathbb{Z}_N, \mathbb{H})}.$$

Third, for any $X, Y \in \ell(\mathbb{Z}_N, \mathbb{H})$:

$$\begin{aligned}
\langle X, Y \rangle_{\ell(\mathbb{Z}_N, \mathbb{H})} &= \sum_{n=0}^{N-1} \langle X[n], Y[n] \rangle_{\mathbb{H}} \\
&= \sum_{n=0}^{N-1} \langle Y[n], X[n] \rangle_{\mathbb{H}}^* \\
&= \left(\sum_{n=0}^{N-1} \langle Y[n], X[n] \rangle_{\mathbb{H}} \right)^* \\
&= \langle Y, X \rangle_{\ell(\mathbb{Z}_N, \mathbb{H})}^*.
\end{aligned}$$

Lastly, for $X \in \ell(\mathbb{Z}_N, \mathbb{H})$ with $X \neq 0$, we have $X[n] \neq 0$ for some n and so:

$$\langle X, X \rangle_{\ell(\mathbb{Z}_N, \mathbb{H})} = \sum_{n=0}^{N-1} \langle X[n], X[n] \rangle_{\mathbb{H}} = \sum_{n=0}^{N-1} \|X[n]\|_{\mathbb{H}}^2 > 0.$$

Therefore (5) is indeed an inner product space on (3), as claimed.

3.2 $\mathrm{GST}_{\mathbf{U}}$ *is a linear operator*

To better understand the GST operator, we first show that GST_U is, for any unitary operator U, a well-defined linear operator from $\ell(\mathbb{Z}_N, \mathbb{H})$ into itself. That is, the domain of the definition of GST_U is $\ell(\mathbb{Z}_N, \mathbb{H})$. Then for any $X \in \ell(\mathbb{Z}_N, \mathbb{H})$, the fact that $\mathrm{GST}_U(X)$ is N-periodic follows from the assumption that $U^N = \mathrm{I}$:

$$\begin{aligned}
\mathrm{GST}_U(X)[n+N] &= \sum_{n'=0}^{N-1} U^{-n'(n+N)} X[n'] \\
&= \sum_{n'=0}^{N-1} U^{-nn'} \left(U^N\right)^{-n'} X[n'] \\
&= \sum_{n'=0}^{N-1} U^{-nn'} X[n'] \\
&= \mathrm{GST}_U(X)[n].
\end{aligned}$$

Having this fact, we next show that GST_U is linear. Let X_1, X_2 be vectors of length N such that both $X_1, X_2 \in \ell(\mathbb{Z}_N, \mathbb{H})$. Then for all $n \in \mathbb{Z}_N$ and scalars $\alpha_1, \alpha_2 \in \mathbb{C}$, we fix n and have:

$$\begin{aligned}
\Big[\mathrm{GST}_U(\alpha X + \beta Y)\Big][n] &= \sum_{n'=0}^{N-1} U^{-n'n}\Big(\alpha X[n'] + \beta Y[n']\Big) \\
&= \sum_{n'=0}^{N-1} \Big(\alpha U^{-n'n} X[n'] + \beta U^{-n'n} Y[n']\Big) \\
&= \alpha \sum_{n'=0}^{N-1} U^{-n'n} X[n'] + \beta \sum_{n'=0}^{N-1} U^{-n'n} Y[n'] \\
&= \alpha \mathrm{GST}_U(X)[n] + \beta \mathrm{GST}_U(Y)[n] \\
&= \Big[\alpha \mathrm{GST}_U(X) + \beta \mathrm{GST}_U(Y)\Big][n].
\end{aligned}$$

Since this is true for all $n \in \mathbb{Z}_N$, $\alpha, \beta \in \mathbb{C}$ and $X, Y \in \ell(\mathbb{Z}_N, \mathbb{H})$, then GST is a linear operator.

As discussed in the introduction, the GST is a generalization of the DFT. Moreover, one of the well-known properties of the DFT is that conjugating a translation operator by the DFT results in a modulation operator. Specifically, defining the operator $\mathrm{T}^k : \ell(\mathbb{Z}_N, \mathbb{H}) \to \ell(\mathbb{Z}_N, \mathbb{H})$, $(\mathrm{T}^k x)[n] := x[n-k]$, it is easy to show that $\mathrm{DFT}(\mathrm{T}^k x)[n] = \mathrm{e}^{-\frac{2\pi \mathrm{i} kn}{N}} \mathrm{DFT}(x)[n]$ for any $k, n \in \mathbb{Z}$. If we generalize this notion of translation to the operator $\mathrm{T}^k : \ell(\mathbb{Z}_N, \mathbb{H}) \to \ell(\mathbb{Z}_N, \mathbb{H})$, $(\mathrm{T}^k X)[n] := X[n-k]$ then, making the substitution $n'' = n' - k$ gives a similar result:

$$\begin{aligned}
\mathrm{GST}_U(\mathrm{T}^k X)[n] &= \sum_{n'=0}^{N-1} U^{-nn'} X[n'-k] \\
&= U^{-kn} \sum_{n''=0}^{N-1} U^{-nn''} X[n''] \\
&= U^{-kn} \mathrm{GST}_U(X)[n].
\end{aligned}$$

3.3 The adjoint of the GST

The GST also generalizes the well-known characteristics of the *adjoint* of the DFT, namely the operator $\mathrm{DFT}^* : \ell(\mathbb{Z}_N) \to \ell(\mathbb{Z}_N)$ where:

$$\mathrm{DFT}^*(x)[n] = \sum_{n'=0}^{N-1} \mathrm{e}^{\frac{2\pi i n n'}{N}} x[n']. \tag{6}$$

In particular, the only difference between DFT (2) and its adjoint (6) is the sign of the exponential. We now show a similar result holds for the GST.

Theorem 2. $\mathrm{GST}_U^* = \mathrm{GST}_{U^*}$.

Proof. It suffices to show that for any two vectors $X, Y \in \ell(\mathbb{Z}_N, H)$:

$$\langle X, \mathrm{GST}_U^*(Y)\rangle_{\ell(\mathbb{Z}_N,\mathbb{H})} = \langle X, \mathrm{GST}_{U^*}(Y)\rangle_{\ell(\mathbb{Z}_N,\mathbb{H})}.$$

By definition of the adjoint and the inner product (5) on $\ell(\mathbb{Z}_N, \mathbb{H})$, we have:

$$\begin{aligned}
\langle X, \mathrm{GST}_U^*(Y)\rangle_{\ell(\mathbb{Z}_N,\mathbb{H})} &= \langle \mathrm{GST}_U(X), Y\rangle_{\ell(\mathbb{Z}_N,\mathbb{H})} \\
&= \sum_{n=0}^{N-1} \Big\langle \mathrm{GST}_U(X)[n], Y[n] \Big\rangle_{\mathbb{H}} \\
&= \sum_{n=0}^{N-1} \Big\langle \sum_{n'=0}^{N-1} U^{-nn'} X[n'], Y[n] \Big\rangle_{\mathbb{H}}.
\end{aligned}$$

Interchanging sums and using the fact that U is unitary gives:

$$\begin{aligned}
\langle X, \mathrm{GST}_U^*(Y)\rangle_{\ell(\mathbb{Z}_N,\mathbb{H})} &= \sum_{n=0}^{N-1}\sum_{n'=0}^{N-1}\left\langle U^{-nn'}X[n'], Y[n]\right\rangle_{\mathbb{H}} \\
&= \sum_{n'=0}^{N-1}\sum_{n=0}^{N-1}\left\langle X[n'], (U^{-nn'})^* Y[n]\right\rangle_{\mathbb{H}} \\
&= \sum_{n'=0}^{N-1}\left\langle X[n'], \sum_{n=0}^{N-1} U^{nn'}Y[n]\right\rangle_{\mathbb{H}} \\
&= \sum_{n'=0}^{N-1}\left\langle X[n'], \mathrm{GST}_{U^{-1}}(Y)[n']\right\rangle_{\mathbb{H}} \\
&= \left\langle X, \mathrm{GST}_{U^*}(Y)\right\rangle_{\ell(\mathbb{Z}_N,\mathbb{H})}.
\end{aligned}$$

□

Despite these many similarities between the DFT and the GST, they are not completely identical. Indeed, the DFT is invertible with $\mathrm{DFT}^{-1} = \frac{1}{N}\mathrm{DFT}^*$. However, this is not always the case for the GST. For example, when $U = I$, GST_I produces N copies of the sum of the entries of X, and so GST_I is clearly not invertible. It will be shown with more in-depth spectral analysis in Chapter V, GST_U is only invertible for certain choices of U.

3.4 Estimating frequency with the GST

Applications discussed in Chapter VI have $\mathbb{H} = L^2(\mathbb{R}^2)$, where the most important property of the GST is its ability to estimate rate of rotation. To be precise, the dominant frequency of a complex-valued signal x is usually taken as the location of the highest peak in its DFT. The GST has a similar property: given video of an object rotating at a uniform rate, the next result shows how this rate of rotation can be computed by finding the "peak" of the GST.

Theorem 3. *If $X \in \ell(\mathbb{Z}_N, \mathbb{H})$ has the property that there exist $n_0 \in \mathbb{Z}$ such that $X[n+1] = U^{n_0}X[n]$ for all $n \in \mathbb{Z}$, then $n_0 = \arg\max_n \|\mathrm{GST}_U(X)[n]\|$, where $\|\cdot\|$ may*

be any U-invariant norm, that is, any norm that has the property that $\|Ux\| = \|x\|$ for all $x \in \mathbb{H}$.

Proof. The given assumption is equivalent to having $X[n] = U^{nn_0}X[0]$ for all $n \in \mathbb{Z}$. Using this property, we have:

$$\begin{aligned}\|\mathrm{GST}_U(X)[n]\| &= \left\|\sum_{n'=0}^{N-1} U^{-nn'}X[n']\right\| \\ &= \left\|\sum_{n'=0}^{N-1} U^{-nn'}U^{n_0n'}X[0]\right\| \\ &= \left\|\sum_{n'=0}^{N-1} U^{n'(n_0-n)}X[0]\right\|.\end{aligned}$$

By the triangle inequality and the U-invariance of the norm:

$$\left\|\sum_{n'=0}^{N-1} U^{n'(n_0-n)}X[0]\right\| \leq \sum_{n'=0}^{N-1}\left\|U^{n'(n_0-n)}X[0]\right\| = \sum_{n'=0}^{N-1}\|X[0]\| = N\|X[0]\|.$$

Therefore $\|\mathrm{GST}_U(X)\|$ has an upper bound of $N\|X[0]\|$. Moreover, this upper bound is achieved at $n = n_0$:

$$\|\mathrm{GST}_U(X)[n_0]\| = \left\|\sum_{n'=0}^{N-1} U^{n'(n_0-n_0)}X[0]\right\| = \left\|\sum_{n'=0}^{N-1} X[0]\right\| = N\|X[0]\|.$$

Hence, $N\|X[0]\|$ is the maximum value achieved by any choice of n, and so $n_0 = \arg\max_n \|\mathrm{GST}_U(X)[n]\|$, as claimed. □

Note that n_0 is not necessarily unique. For example, when $U = I$ the GST is constant and so any choice of n_0 is equal to $\arg\max_n \|\mathrm{GST}_U(X)[n]\|$. In summary, the previous result shows how a GST, once computed, can provide useful information about the rate of rotation (or translation) in a given sequence of vectors. However, as we shall see in the following chapter, the GST can be expensive to compute, both in time and memory, whenever N or the dimension of $\mathbb{H}$ is large. This raises the question: is there a fast, memory efficient algorithm for computing a GST? In the

next chapter, we show that such an algorithm indeed exists. In particular, since the GST is a generalization of the DFT, we look to generalize the well-known decimation-in-frequency fast Fourier transform algorithm to the GST setting.

IV. A fast algorithm for computing the GST

In the previous chapter, we discussed how the GST is a useful tool for estimating frequency. Note however, that computing the GST directly requires N^2 matrix-vector multiplications, that is N multiplications for each of the N distinct choices of n in (4). In this chapter, we will present a fast algorithm for computing the GST which generalizes the way in which the FFT quickly computes the DFT.

As mentioned in the introduction, Cooley and Tukey's FFT algorithm takes advantage of the property that an DFT of a given nonprime size can be successively broken into smaller DFTs of prime order. When N is not prime, the GST behaves similarly. Indeed, if N is divisible by $P \in \mathbb{N}$, then an N-point GST can be computed in terms of P GSTs of length $\frac{N}{P}$. To see this, note that making the substitution $n' = Pq + p$ gives:

$$\begin{aligned}
\mathrm{GST}_U(X)[n] &= \sum_{n'=0}^{N-1} U^{-nn'} X[n'] \\
&= \sum_{p=0}^{P-1} \sum_{q=0}^{\frac{N}{P}-1} U^{-(Pq+p)n} X[Pq+p] \\
&= \sum_{p=0}^{P-1} U^{-pn} \sum_{q=0}^{\frac{N}{P}-1} \left(U^P\right)^{-qn} X[Pq+p].
\end{aligned}$$

By the definition of GST, we have:

$$\mathrm{GST}_U(X)[n] = \sum_{p=0}^{P-1} U^{-pn} \mathrm{GST}_{U^P}(X[P(0:\tfrac{N}{P}-1)+p])[n],$$

where the notation $X[P(0:\frac{N}{P}-1)+p]$ denotes the subsequence of X whose indices lie in the coset $\{Pq+p : q = 0, \ldots, \frac{N}{P}-1\}$.

For example, a 16-point GST can be broken into two GSTs of size 4, or alternatively eight GSTs of size 2, or even two GSTs of size 8. This property also holds for the FFT. In fact, in the literature, a size 2^k FFT for some $k \in \mathbb{N}$ is referred to as a radix-2 FFT. Meanwhile, a FFT of size N where N can be factored into non-primes is

sometimes referred to as a mixed-radix FFT. Later in this chapter we show that the prime factorization is the particular factorization which makes our FGST algorithm perform at its best.

Moreover, in such algorithms, it is not only the factors themselves that matter, their *order* is also significant. Indeed, a fundamental part of many FFT computations, namely the *bit reversal* operation defined below, is defined in terms of this ordered list of factors. Note that we are using the term "bit reversal" to imply an n-ary representation reversal. For example, when $N = 2^k$ for some $k \in \mathbb{Z}$, the binary representation of any number in the range $(0, \ldots, N-1)$ is length k with each "significant digit" having value either 0 or 1. A *bit reversal* reverses the order of these digits. More generally, for any N, the number of prime factors of N determines the length of the n-ary representation, where each significant digit is determined by the prime factors themselves. To be precise, we have the following definition.

Definition 4. Let $N \in \mathbb{N}$ and let $\mathcal{P} = \{P_0 \cdots P_{J-1}\}$ be an ordered factorization of N, that is $N = \prod_{j=0}^{J-1} P_j$, where each P_j is a positive integer greater than 1. We define the partial product sequence as $\mathcal{N} = \{N_0, \ldots, N_J\}$, where for any $j = (0, \ldots, J)$,

$$N_j := \prod_{j'=j}^{J-1} P_{j'}$$

which gives $N_J = 1$, and we also observe:

$$N_j = P_j N_{j+1}, \quad \forall j = (0, \ldots, J-1). \tag{7}$$

For each $j = (1, \ldots, J)$ we define the *bit reversal* β_j as a permutation on $\{0, \ldots, \frac{N}{N_j} - 1\}$. This definition is recursive. Specifically, we let $\beta_1(n) := n$ for all $n = (0 \ldots, P_0 - 1)$. Meanwhile, for any $j = (1 \ldots, J-1)$ we let:

$$\beta_{j+1}(P_j n + p) := \tfrac{N}{N_j} p + \beta_j(n), \quad \forall n = (0, \ldots, \tfrac{N}{N_j} - 1),\ p = (0, \ldots, P_j - 1). \tag{8}$$

For example, suppose $N = 30$ and let $P_0 = 2$, $P_1 = 3$, and $P_2 = 5$. The partial products for this ordered factorization of N are $N_0 = 30$, $N_1 = 15$, $N_2 = 5$, and $N_3 = 1$. So, using (8), $n = 27$ has the n-ary expansion $27 = 2(1) + 2(5) + 1(15) = 2N_3 + 2N_2 + 1N_1$. Then, to compute $\beta_3(27)$, we reverse the bits such that $\beta_3(27) = 1\frac{N}{N_0} + 2\frac{N}{N_1} + 2\frac{N}{N_2} = 1(1) + 2(2) + 2(6) = 17$. However, computing the bit reversal in this way is cumbersome. An equivalent but faster method for computing the bit reversal is to use Kronecker tensor products:

$$\beta_{j+1} := \beta_j \otimes 1_{P_j} + \frac{N}{N_j} 1_{\frac{N}{N_j}} \otimes (0 : P_j - 1). \tag{9}$$

The $\otimes$ symbols above represent the Kronecker product operations, also known as matrix direct products, and the the 1_N denotes an $N \times 1$ vector of ones. This method of computing bit reversal is also implemented in MATLAB code given in the appendix. Returning to the $N = 30$ example, we will now show how to recursively compute bit reversal using (9):

$$\beta_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$

$$\beta_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + 2 \begin{bmatrix} 1 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ 2 \\ 4 \\ 0 \\ 2 \\ 4 \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \\ 4 \\ 1 \\ 3 \\ 5 \end{bmatrix},$$

$$\beta_3 = \begin{bmatrix} 0 \\ 2 \\ 4 \\ 1 \\ 3 \\ 5 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} + 6 \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 2 \\ 2 \\ 2 \\ 2 \\ 2 \\ 4 \\ 4 \\ 4 \\ 4 \\ 1 \\ 1 \\ 1 \\ 1 \\ 3 \\ 3 \\ 3 \\ 3 \\ 5 \\ 5 \\ 5 \\ 5 \\ 5 \end{bmatrix} + \begin{bmatrix} 0 \\ 6 \\ 12 \\ 18 \\ 24 \\ 0 \\ 6 \\ 12 \\ 18 \\ 24 \\ 0 \\ 6 \\ 12 \\ 18 \\ 24 \\ 0 \\ 6 \\ 12 \\ 18 \\ 24 \\ 0 \\ 6 \\ 12 \\ 18 \\ 24 \end{bmatrix} = \begin{bmatrix} 0 \\ 6 \\ 12 \\ 18 \\ 24 \\ 2 \\ 8 \\ 14 \\ 20 \\ 26 \\ 4 \\ 10 \\ 16 \\ 22 \\ 28 \\ 1 \\ 7 \\ 13 \\ 19 \\ 25 \\ 3 \\ 9 \\ 15 \\ 21 \\ 27 \\ 5 \\ 11 \\ 17 \\ 23 \\ 29 \end{bmatrix}.$$

The binary bit reversal discussed in the introduction exploited the characteristics of binary bit flipping. The n-ary bit reversal given in Definition 4 generalizes this operation to any N. Further note that these bit reversals are well-defined permutations. We can see this by induction on j. By definition, $\beta_1(n) = n$, and, thus, β_1 is one-to-one. Now suppose β_j is one-to-one. To see that β_{j+1} is also one-to-one, assume for some $k_1, k_2 = (0, \ldots, \frac{N}{N_{j+1}} - 1)$ that $\beta_{j+1}(k_1) = \beta_{j+1}(k_2)$. Write $k_1 = P_j n_1 + p_1$ and $k_2 = P_j n_2 + p_2$ for some $n_1, n_2 = (0, \ldots, \frac{N}{N_j} - 1)$ and $p_1, p_2 = (0, \ldots, P_j - 1)$, we have:

$$\tfrac{N}{N_j} p_1 + \beta_j(n_1) = \beta_{j+1}(P_j n_1 + p_1) = \beta_{j+1}(P_j n_2 + p_2) = \tfrac{N}{N_j} p_2 + \beta_j(n_2).$$

Subtracting gives $\frac{N}{N_j}(p_1 - p_2) = \beta_j(n_2) - \beta_j(n_1)$. Since $\beta_j(n_1), \beta_j(n_2) \in [0, \ldots, \frac{N}{N_j} - 1]$ this implies $\frac{N}{N_j}(p_1 - p_2) = 0$. Thus, $p_1 = p_2$, and so $\beta_j(n_1) = \beta_j(n_2)$, implying by the inductive hypothesis that $n_1 = n_2$. Thus, $k_1 = k_2$ and so β_{j+1} is one-to-one, as claimed.

In the next result, we show how this notion of bit reversal can be used in a fast, memory efficient algorithm for computing the GST. In short, this result shows that the decimation-in-frequency method for computing an FFT generalizes to the GST setting.

Theorem 5. *The* GST *of any* $X \in \ell(\mathbb{Z}_N, \mathbb{H})$ *can be computed by the following Fast Geometric Sum Transform (*FGST*)algorithm:*

(00) *let* $N \in \mathbb{N}$, $J \in \mathbb{N}$ *with* $1 < J < N$

(01) *let* $X^{(0)}[n] = X[n]$ *for all* $n = (0, \ldots, N-1)$

(02) *for* $j = (0, \ldots, J-1)$

(03) *for* $k = (0, \ldots, N_{j+1} - 1)$

(04) *for* $l = (0, \ldots, \frac{N}{N_j} - 1)$

(05) *for* $p = (0, \ldots, P_j - 1)$

(06)
$$X^{(j+1)}[N_{j+1}(P_j l + p) + k] = \sum_{q=0}^{P_j - 1} \left(U^{N_{j+1}}\right)^{-q\beta_{j+1}(P_j l + p)} X^{(j)}[N_{j+1}(P_j l + q) + k]$$

(07) *end*

(08) *end*

(09) *end*

(10) *end*

Then for all $n = (0, \ldots, N-1)$*:*

$$X^{(J)}[n] = \mathrm{GST}_U(X)[\beta_J(n)]. \tag{10}$$

Proof. For brevity of notation, let $N \in \mathbb{N}$ and $J \in \mathbb{N}$ with $J < N$, then: $\mathrm{GST}_U(X) := \mathrm{GST}(X, U)$. For any $j = (0, \ldots, J)$ we claim it suffices to show the values of the jth iteration of the GST algorithm are given by:

$$X^{(j)}[N_j a + b] = \mathrm{GST}(X[N_j(0 : \tfrac{N}{N_j} - 1) + b], U^{N_j})[\beta_j(a)] \tag{11}$$

for all $a = (0, \ldots, \frac{N}{N_j} - 1)$ and $b = (0, \ldots, N_j - 1)$, where β_j is the $\frac{N}{N_j}$-ary bit reversal permutation on $\{0, \ldots, \frac{N}{N_j} - 1\}$ given in Definition 4. Indeed, if (11) holds for all $j = (0, \ldots, J)$, then, in particular, when $j = J$ we have $a = (0, \ldots, N-1)$ and $b = 0$.

Letting $a = n$, (11) becomes:

$$X^{(J)}[n] = \mathrm{GST}(X, U)[\beta_J(n)].$$

We prove (11) by induction on j. When $j = 0$, we have $a = 0$ and $b = (0, \ldots, N-1)$ where the right hand side of (11) becomes:

$$\mathrm{GST}(X[b], I)][\beta_0(0)] = X[b] = X^{(0)}[b].$$

Hence (11) holds for $j = 0$. Now, assume (11) holds for a given $j \in \{0, \ldots, J\}$. We want to show (11) holds for $j + 1$, that is:

$$\begin{aligned} &X^{(j+1)}[N_{j+1}(P_j l + p) + k] \\ &= \mathrm{GST}(X[N_{j+1}(0 : \tfrac{N}{N_{j+1}} - 1) + k], U^{N_{j+1}})[\beta_{j+1}(P_j l + p)] \end{aligned} \tag{12}$$

where $l = (0 \ldots, \frac{N}{N_j} - 1)$, $p = (0, \ldots, P_j - 1)$, and $k = (0, \ldots, N_{j+1} - 1)$. The left hand side of (12) is given by Line (06) of the algorithm. To see that this equals equals the right hand side we expand it:

$$\begin{aligned} &\mathrm{GST}(X[N_{j+1}(0 : \tfrac{N}{N_{j+1}} - 1) + k], U^{N_{j+1}})[\beta_{j+1}(P_j l + p)] \\ &= \sum_{c=0}^{\frac{N}{N_{j+1}}-1} (U^{N_{j+1}})^{-c\beta_{j+1}(P_j l + p)} X[N_{j+1} c + k]. \end{aligned} \tag{13}$$

Letting $c = P_j a + q$ where $a = (0, \ldots, \frac{N}{N_j} - 1)$ and $q = (0, \ldots, P_j - 1)$, (13) becomes:

$$\begin{aligned}
&\mathrm{GST}(X[N_{j+1}(0 : \tfrac{N}{N_{j+1}} - 1) + k], U^{N_{j+1}})[\beta_{j+1}(P_j l + p)] \\
&= \sum_{q=0}^{P_j - 1} \sum_{a=0}^{\frac{N}{N_j} - 1} (U^{N_{j+1}})^{-(P_j a + q)\beta_{j+1}(P_j l + p)} X[N_{j+1}(P_j a + q) + k] \\
&= \sum_{q=0}^{P_j - 1} (U^{N_{j+1}})^{-q\beta_{j+1}(P_j l + p)} \sum_{a=0}^{\frac{N}{N_j} - 1} (U^{N_j})^{-a\beta_{j+1}(P_j l + p)} X[N_j a + (N_{j+1} q + k)] \\
&= \sum_{q=0}^{P_j - 1} (U^{N_{j+1}})^{-q\beta_{j+1}(P_j l + p)} \\
&\qquad \cdot \mathrm{GST}(X[N_j(0 : \tfrac{N}{N_j} - 1) + (N_{j+1} q + k)], U^{N_j})[\beta_{j+1}(P_j l + p)]. \qquad (14)
\end{aligned}$$

By definition, $\beta_{j+1}(P_j l + p) = \frac{N}{N_j} p + \beta_j(l)$. It is also simple to show that the GSTs in (14) are $\frac{N}{N_j}$-periodic. Thus:

$$\begin{aligned}
&\mathrm{GST}(X[N_j(0 : \tfrac{N}{N_j} - 1) + (N_{j+1} q + k)], U^{N_j})[\beta_{j+1}(P_j l + p)] \\
&= \mathrm{GST}(X[N_j(0 : \tfrac{N}{N_j} - 1) + (N_{j+1} q + k)], U^{N_j})[\tfrac{N}{N_j} p + \beta_j(l)] \\
&= \mathrm{GST}(X[N_j(0 : \tfrac{N}{N_j} - 1) + (N_{j+1} q + k)], U^{N_j})[\beta_j(l)] \\
&= X^{(j)}[N_j l + N_{j+1} q + k] \qquad (15)
\end{aligned}$$

where the final equality follows by the inductive hypothesis (11). Then substituting (15) into (14) gives:

$$\begin{aligned}
&\mathrm{GST}(X[N_{j+1}(0 : \tfrac{N}{N_{j+1}} - 1) + k], U^{N_{j+1}})[\beta_{j+1}(P_j l + p)] \\
&= \sum_{q=0}^{P_j - 1} (U^{N_{j+1}})^{-q\beta_{j+1}(P_j l + p)} X^{(j)}[N_j l + N_{j+1} q + k] \\
&= \sum_{q=0}^{P_j - 1} (U^{N_{j+1}})^{-q\beta_{j+1}(P_j l + p)} X^{(j)}[N_{j+1}(P_j l + q) + k] \qquad (16)
\end{aligned}$$

which is Line (06) of the algorithm. Thus, (11) holds for all $j = (0, \ldots, J)$. □

The computational requirement to run a GST of size P_j is only P_j places in memory. MATLAB code to compute the FGST provided in the appendix advantageously uses in-place computation for memory efficiency. Note that at the heart of the algorithm of Theorem 5 lies a GST of size P_j. Indeed, the quantity computed in Line (06) can be rewritten as:

$$\begin{aligned}
&\sum_{q=0}^{P_j-1} (U^{N_{j+1}})^{-q\beta_{j+1}(P_j l+p)} X^{(j)}[N_{j+1}(P_j l+q)+k] \\
&= \sum_{q=0}^{P_j-1} (U^{N_{j+1}})^{-q(\frac{N}{N_j}p+\beta_j l)} X^{(j)}[N_{j+1}(P_j l+q)+k] \\
&= \sum_{q=0}^{P_j-1} (U^{\frac{N}{P_j}})^{-qp} U^{-N_{j+1}q\beta_j(l)} X^{(j)}[(N_j l+k)+N_{j+1}q]. \qquad (17)
\end{aligned}$$

which is precisely the pth value of GST of:

$$\{U^{-N_{j+1}q\beta_j(l)} X^{(j)}[(N_j l+k)+N_{j+1}q]\}_{q=0}^{P_0-1}.$$

That is, the algorithm of Theorem 5 indeed breaks a single large GST computation into many smaller ones.

To see that the FGST is indeed faster than a direct computation of (4), note that for any fixed $j = (0, \ldots, J-1)$, Lines (03)–(09) involve $(N_{j+1})(\frac{N}{N_j})(P_j)(P_j) = NP_j$ matrix-vector multiplications. Thus, the total matrix-vector multiplications is $N\sum_{j=0}^{J-1} P_j$. In the case when $N = P^J$, that total becomes $PN\log_P N$ matrix-vector multiplications. For a nonprime N, this is less than the N^2 such multiplications used in a direct computation of (4).

Note that the FGST algorithm of Theorem 5 depends on a given ordered factorization of N, as seen in Definition 4. From the point-of-view of minimizing computation, the optimal factorization of N is its prime factorization. Indeed, if the factors of N are taken to be $\{P_j\}_{j=0}^{J-1}$ where $P_{j_0} = Q_0 Q_1$ is not prime, then a more factored

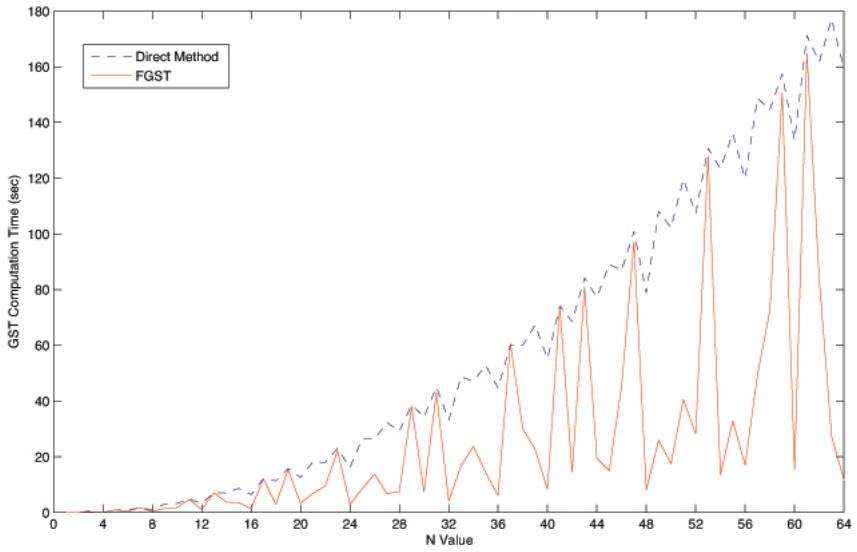


Figure 4: A comparison of the GST computational time using the direct method versus using the FGST, for length N ranging from 1 to 64.

algorithm is faster; we have:

$$N \sum_{j=0}^{J-1} P_j \geq N(Q_0 + Q_1 + \sum_{\substack{j=0 \\ j \neq j_0}}^{J-1} P_j)$$

since $Q_0 + Q_1 \leq Q_0 Q_1 = P_{j_0}$.

MATLAB code implementing the FGST algorithm with a full prime factorization is given in the appendix. When N itself is a prime number, the direct method of computing the GST takes the same amount of time as the FGST, both using N^2 matrix-vector multiplications. In fact, from Figure 4 it is clear that the FGST is always at least as fast as the direct method, and is often significantly faster.

Note that here we are measuring computational complexity solely in terms of the number of matrix-vector multiplications. The true computational cost depends on the expense of each such multiplication, which, in turn, depends on the nature of U. We discuss these ideas in greater detail in the following chapter.

V. Spectral analysis of the GST

In the previous chapter, we gave a fast algorithm for computing the GST that was inspired by the classical FFT decimation-in-frequency approach. In this chapter, we show this similarity is not a coincidence; the GST can be regarded as a system of DFTs, provided we know the eigenvectors and eigenvalues of U. To be precise, assume $\mathbb{H}$ is M-dimensional and note that since U is unitary, then it is normal. Then, by Shur's Theorem U can be unitarily diagonalized as $U = VDV^*$, where D is a diagonal matrix and V is unitary.

Since U is unitary, we further know the diagonal entries of D, namely the eigenvalues of U, are unimodular, that is, have modulus one. Moreover, since $U^N = I$, we know these eigenvalues are actually Nth roots of unity. Specifically, letting $\{\lambda_m\}_{m=0}^{M-1}$ denote the eigenvalues of U, we have $\lambda_m^N = 1$ for all m. Letting v_m denote the eigenvector corresponding to λ_m, and writing $\lambda_m = \mathrm{e}^{\frac{2\pi \mathrm{i}\theta_m}{N}}$, we have the following result.

Theorem 6. *For any eigenvector v_m of U with corresponding eigenvalue $\mathrm{e}^{\frac{2\pi \mathrm{i}\theta_m}{N}}$,*

$$\langle v_m, \mathrm{GST}_U(X)[n]\rangle = \sum_{n'=0}^{N-1} \mathrm{e}^{-\frac{2\pi \mathrm{i}\theta_m nn'}{N}} \langle v_m, X[n']\rangle.$$

Proof. By the definition of the GST given in (4),

$$\langle v_m, \mathrm{GST}_U(X)[n]\rangle = \left\langle v_m, \sum_{n'=0}^{N-1} U^{-nn'} X[n'] \right\rangle = \sum_{n'=0}^{N-1} \langle v_m, U^{-nn'} X[n']\rangle.$$

Then since $U^* = U^{-1}$, this becomes:

$$\begin{aligned}
\sum_{n'=0}^{N-1} \langle v_m, U^{-nn'} X[n']\rangle &= \sum_{n'=0}^{N-1} \langle U^{nn'} v_m, X[n']\rangle \\
&= \sum_{n'=0}^{N-1} \left\langle \mathrm{e}^{\frac{2\pi \mathrm{i}\theta_m nn'}{N}} v_m, X[n']\right\rangle \\
&= \sum_{n'=0}^{N-1} \mathrm{e}^{-\frac{2\pi \mathrm{i}\theta_m nn'}{N}} \langle v_m, X[n']\rangle. \qquad \square
\end{aligned}$$

We now use Theorem 6 to investigate the invertibility of the GST. In particular, applying this result to U^{-1} gives:

$$\langle v_m, \mathrm{GST}_U^*(Y)[n]\rangle = \langle v_m, \mathrm{GST}_{U^{-1}}(Y)[n]\rangle = \sum_{n''=0}^{N-1} \mathrm{e}^{\frac{2\pi \mathrm{i}\theta_m nn''}{N}} \langle v_m, Y[n'']\rangle.$$

Let $Y = \mathrm{GST}_U(X)$ such that:

$$\begin{aligned}
\langle v_m, \mathrm{GST}_U^*(\mathrm{GST}_U(X))[n]\rangle &= \sum_{n''=0}^{N-1} \mathrm{e}^{\frac{2\pi \mathrm{i}\theta_m nn''}{N}} \langle v_m, \mathrm{GST}_U(X)[n'']\rangle \\
&= \sum_{n''=0}^{N-1} \mathrm{e}^{\frac{2\pi \mathrm{i}\theta_m nn''}{N}} \sum_{n'=0}^{N-1} \mathrm{e}^{-\frac{2\pi \mathrm{i}\theta_m n'n''}{N}} \langle v_m, (X)[n']\rangle \\
&= \sum_{n'=0}^{N-1} \langle v_m, (X)[n']\rangle \sum_{n''=0}^{N-1} \mathrm{e}^{\frac{2\pi \mathrm{i}\theta_m (n-n')n''}{N}}. \qquad (18)
\end{aligned}$$

By the Geometric Sum Formula,

$$\sum_{n''=0}^{N-1} \mathrm{e}^{\frac{2\pi \mathrm{i}\theta_m (n-n')n''}{N}} = \begin{cases} N, & \mathrm{e}^{\frac{2\pi \mathrm{i}\theta_m (n-n')}{N}} = 1 \\ \frac{1-\mathrm{e}^{2\pi \mathrm{i}\theta_m (n-n')}}{1-\mathrm{e}^{\frac{2\pi \mathrm{i}\theta_m (n-n')}{N}}}, & \mathrm{e}^{\frac{2\pi \mathrm{i}\theta_m (n-n')}{N}} \neq 1 \end{cases} = \begin{cases} N, & \frac{\theta_m (n-n')}{N} \in \mathbb{Z} \\ 0, & \frac{\theta_m (n-n')}{N} \notin \mathbb{Z} \end{cases} \qquad (19)$$

Then substituting (19) into (18) gives:

$$\begin{aligned}
\langle v_m, \mathrm{GST}_U^*(\mathrm{GST}_U(X))[n]\rangle &= N \sum_{\substack{n'=0 \\ N|\theta_m(n-n')}}^{N-1} \Big\langle v_m, X[n']\Big\rangle \\
&= N\Big\langle v_m, \sum_{\substack{n'=0 \\ N|\theta_m(n-n')}}^{N-1} X[n']\Big\rangle.
\end{aligned}$$

Thus, we see that the mth component of $\frac{1}{N}\mathrm{GST}_U^*(\mathrm{GST}_U(X))$ is the like component of the sum of the entries of X whose indices lie in a certain coset of $\mathbb{Z}_N$. When each θ_m is relatively prime to N, each of these cosets only consists of a single index, meaning the GST is invertible:

Corollary 7. *If θ_m is relatively prime to N for all m, then the* GST *is invertible with* $\mathrm{GST}_U^{-1} = \frac{1}{N}\mathrm{GST}_U^*$.

If, on the other hand, even a single θ_m shares a common factor with N, then this summing process destroys some of the frequency information in the sequence $\{\langle v_m, X[n]\rangle\}_{n=0}^{N-1}$. This implies $\mathrm{GST}_U^* GST_U$, and so GST_U, is not invertible in such cases.

The DFT can be viewed as a special case of the GST that is invertible. To be precise, let $\mathbb{H} = \mathbb{C}$, that is, $N = 1$, and let U be the act of multiplying by $\mathrm{e}^{\frac{2\pi \mathrm{i}}{N}}$, an operation whose only eigenvalue is $\mathrm{e}^{\frac{2\pi \mathrm{i}}{N}}$, that is, $\theta_1 = 1$. Since θ_1 is relatively prime to N, Corollary 7 states that the DFT is invertible with $\mathrm{DFT}^{-1} = \frac{1}{N}\mathrm{DFT}^*$.

However, for the angular-frequency-determination problem that motivated this work, the relatively prime condition of Corollary 7 does not hold, meaning the corresponding GST is not invertible. Indeed, letting U be a rotation operator, the constant vector v_0 is an eigenvector for U with eigenvalue equal to 1, and so $\theta_0 = 0$, which is not relatively prime to N. In this case, Theorem 6 states:

$$\langle v_0, \mathrm{GST}_U(X)[n]\rangle = \sum_{n'=0}^{N-1} 1\langle v_0, X[n']\rangle = \langle v_0, \sum_{n'=0}^{N-1} X[n']\rangle,$$

meaning the DC component of every entry of the GST is the DC component of the sum of all the video frames. Such lossy behavior is accentuated in the extreme case where $U = I$. Here,

$$\mathrm{GST}_I X[n] = \sum_{n'=0}^{N-1} X[n'],$$

meaning every entry of the GST is a sum of all the entries of X.

As these examples illustrate, the idea of computing a GST as a system of M DFTs made explicit in Theorem 6 is a useful theoretical concept. However, we must point out that it offers no real advantages from a computational point of view. This method of computing the GST requires us to first know the eigenvalues and eigenvectors of U. When U is a rotation operator, we do not have these eigenvalues and

eigenvectors explicitly. Moreover, even if these values were known, such as in the case where U is a translation operator, the lengthy computation time of this method makes it undesirable.

Indeed, the Theorem 6 method for finding the GST requires us to first compute MN inner products of the form $\langle v_m, X[n']\rangle$ in $\mathbb{H}$, where $\mathbb{H}$ is M-dimensional. Thus, in general, when the $v_m's$ are not sparse, we expect to need $\mathcal{O}(M^2N)$ operations just to compute these inner products. We then must compute M FFTs of size N to find the values $\langle v_m, \mathrm{GST}_U(X)[n]\rangle$, requiring $\mathcal{O}(MN\log N)$ operations total. We then reconstruct $\mathrm{GST}_U X[n]$ according to:

$$\mathrm{GST}_U X[n] = \sum_{m=0}^{M-1} \langle v_m, \mathrm{GST}_U X[n]\rangle v_m, \tag{20}$$

which uses an additional $\mathcal{O}(M^2N)$ operations, for a total complexity of $\mathcal{O}(M^2N + MN\log N)$. If we assume U has sparse eigenvectors, this value decreases to $\mathcal{O}(MN + MN\log N)$.

Meanwhile, computing the GST using the FGST algorithm of Theorem 5 requires $\mathcal{O}(N\log N)$ matrix-vector products. The imrotate command in MATLAB applies a sparse neighbor interpolation to complete each matrix-vector product, each requiring $\mathcal{O}(M)$ operations. Hence, the total computation requirement for this method is $\mathcal{O}(MN\log N)$. Meanwhile, computing the GST directly from its definition requires $\mathcal{O}(N^2)$ matrix-vector products, and is therefore the slowest method overall. Note that although the methods of Theorems 5 and 6 both require $\mathcal{O}(MN\log N)$ operations, the FGST is less complicated and does not require the eigenvalues and eigenvectors of U to be known.

A similar comparison can be made when U is a circular translation operator, namely, when $\mathbb{H} = \ell(\mathbb{Z}_N)$, $U = \mathrm{T}^1$, and $M = N$. Here, we first need $\langle v_m, X[n']\rangle$ for all m, n' where the v_m's are the Fourier basis, namely the eigenvectors of $U = \mathrm{T}$. For any

fixed n', this task reduces to computing the DFT of $X[n']$. Indeed, since the number of n's is equal to N, and the computational cost of each DFT is $\mathcal{O}(N \log N)$.

We then use Theorem 6 to compute $\langle v_m, \mathrm{GST}_U(X)[n] \rangle$ for all choices of m and n, incurring an additional cost of N DFTs of order N. Next, we reconstruct according to (20) using an additional N DFTs of size N. Therefore, in the case of the translation operator, the total number of operations use to compute the GST according to Theorem 6 is $\mathcal{O}(N^2 \log N)$.

Meanwhile, applying the FGST algorithm of Theorem 5 to the circular translation operator requires $\mathcal{O}(N \log N)$ matrix-vector products, each of which can be implemented in $\mathcal{O}(N)$ operations using MATLAB's circshift command. These circshifts only involve data transfer in memory, that is, no multiplications. Together, the total cost of computing a GST using Theorem 5 when U is a circular translation operator is thus $\mathcal{O}(N^2 \log N)$.

Although the methods of Theorems 5 and 6 are comparable in terms of order of operations, the FGST is less complicated and involves no complex arithmetic. In conclusion, we see that in every case in which we wish to compute the GST it is better, from a computational point of view, to generalize the FFT algorithm in the form of our FGST algorithm (Theorem 5) than to write a GST as a system of FFTs in the spectral domain (Theorem 6). In fact, in the applications discussed in the next chapter, we use an FGST with MATLAB's imrotate command directly without needing to have the eigenvalues and eigenvectors of our rotation operator.

VI. Applications

We now consider several examples in which we apply the FGST algorithm of Theorem 5 in order to estimate angular frequency. Consider our first application of the GST, a pinwheel rotating at a constant unknown rate. To obtain this experimental data, we set up a digital camera in front of a pinwheel that was rotating at a constant rate with air flowing from a nearby fan.

We first shot a series of still frames in continuous shooting mode. According to Landau in [7], every finite energy signal with bandwidth W Hz can be completely recovered by taking samples at the rate of $2W$ per second, which is known as the Nyquist rate. However, as soon as the GST was applied to this image sequence it was evident by every GST image appearing blurry that the continuous shooting frame rate did not meet the Nyquist requirement.

We then recorded 6 seconds of digital video and converted it into a sequence of 192 image frames in MATLAB. The top left image of Figure 5 shows an example of one of these pinwheel images. After computing the GST of this sequence—computing the GST of each of the red, green, and blue channels separately an then combining the result—the frames 5–7 GST images and frames 10–13 GST images are examples of blurry GST images. Note the GST images from frames 1–4 and frames 13–192 are not shown since they look similar to the blurry images already displayed. These blurry images with no color distinction between the shades indicate the rate of rotation does not match that particular choice of n.

In the frame 8 and 9 GST images however, the GST is are clearly sharper and has separated colors. It makes intuitive sense that the two sharper GST images correspond to the true rate of rotation, because the colors have been added coherently on top of each other. During this video sequence, we visually estimated that the pinwheel completed 8.5 revolutions in 6 seconds. Then, the GST algorithm corroborated this as the total number of revolutions of the pinwheel, as seen in the location of the peak in Figure 6.

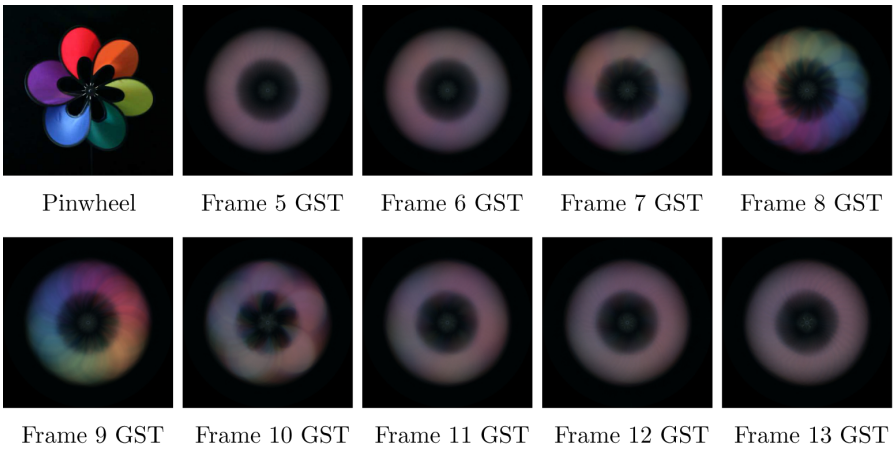
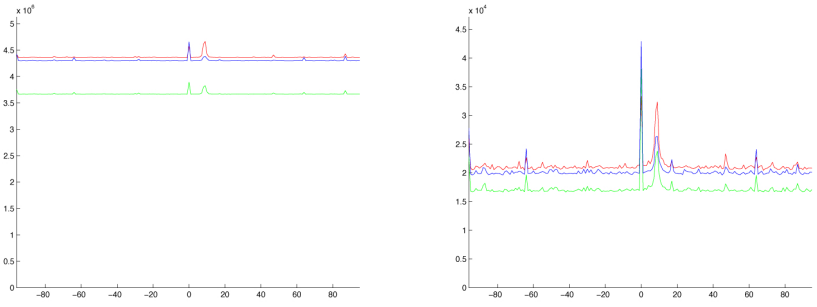


Figure 5: On the top left, the pinwheel image shown is one of the 192-frame pinwheel video sequence. The frame 8 and 9 GST images correspond to the true rate of rotation of the pinwheel, while the other blurry GST images do not.

For our second GST example, we obtained a 100-frame video sequence of real data from a 200-Watt thruster, as discussed in the introduction. This thruster was operating at 80 Volts, and Liu in [8] used a camera to capture these images at one million frames per second with an exposure time of 750 nanoseconds. The top left image in Figure 7 is an example of plasma being ejected from the thruster. Each image from the thruster sequence is 312×260 pixels, and in light of Figure 4, we did not attempt to compute the GST directly. On a standard laptop with a 2.26 GHz Intel Core 2 Duo processor, it took 36 seconds to compute the GST using the FGST algorithm.

After computing the GST of this sequence of thruster images, the frame 1–7 GST images and frame 9–14 GST images are examples of blurry GST images. Like the pinwheel example, these blurry images indicate the rate of rotation does not match that particular choice of n. While challenging to verify visually, the frame 8 GST image corresponds to our true rate of plasma rotation, and is confirmed by the peak of the norms shown in Figure 8. Interpreting this result implies the plasma was rotating at approximately 80 Kilohertz, which is reasonable given the parameters specified in [8].



ℓ^2 Norm ℓ^∞ Norm

Figure 6: The ℓ^2 norm and the ℓ^∞ norm of the GST of the pinwheel video sequence. In both norm plots there is a peak at approximately 8.5, corresponding to the number of revolutions the pinwheel completed in 6 seconds. Note, we do not show the graph of the ℓ^1 norm because it does not provide any information about the rate of rotation. Indeed, since the GST is computed by summing up rotations of the pinwheel frames, the total sums as calculated by the ℓ^1 norm are equal in all GSTs of the video sequence.

As these examples have shown, the GST can be applied to real data and provide an accurate estimate of the rate of rotation. While not shown in this chapter, the GST can also be applied to translation problems to provide an accurate estimate of the lateral velocity. In fact, the appendix includes MATLAB code already prepared with circshift commands, so that is can be applied to translation problems. Further guidance is also included in the appendix about how to change the code appropriately for rotation problems.

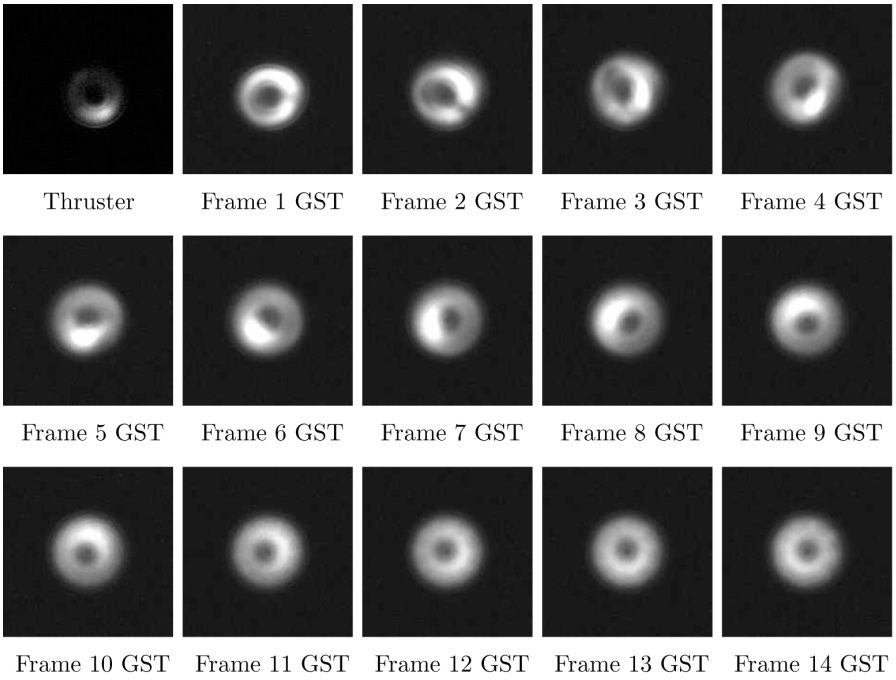


Figure 7: On the top left, the thruster image shown is one of the 100-frame thruster sequence collected by Liu in [8]. The frame 8 GST corresponds to the true rate of rotation of the plasma, while the other blurry GST images do not. The frame 15–192 GSTs are not shown since they look blurry just like the ones already displayed.

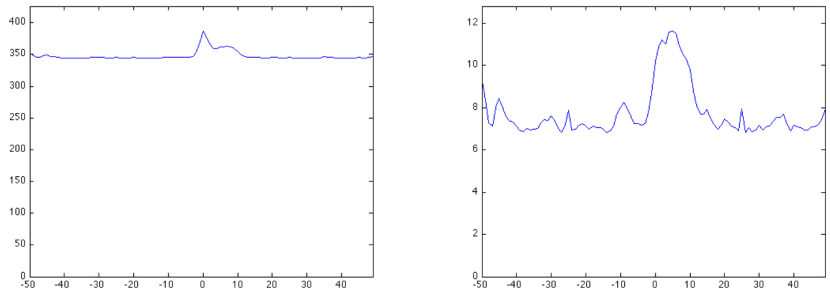


Figure 8: The ℓ^2 norm and the ℓ^∞ norm of the GST of the thruster video sequence are shown above. In both norm graphs there is a peak at approximately 8, indicating the true rate of rotation of the plasma is 80 Kilohertz.

Algorithm for MATLAB *Code*

The following MATLAB code shows algorithms for the direct GST and the FGST. The direct GST should not be implemented since it is not as computational efficient as the FGST, but is presented below for comparison. In both versions of the code, X is a video sequence (3D data cube), and must be defined by the user before the function can be called. Both code versions can also be easily generalized to the case of cyclic translations provided the imrotate commands are replaced by uses of circshift.

The following code computes the GST directly:

```
N = size(X,3);
Y = zeros(size(X));
for n = 0:N-1
   for m = 0:N-1
      Y(:,:,n+1) = Y(:,:,n+1)+imrotate(X(:,:,m+1),-1*m*n);
   end
end
```

This second set of code computes the FGST according to the algorithm of Theorem 5 and is implemented by only using a small amount of memory:

```
% Computing the prime factorization of N and resulting partial products
% according to Definition 4.
CalP = factor(size(X,3))';
J = length(CalP);
CalN = [ flipud(cumprod(flipud (CalP))) ; 1];

% Computing the FGST in bit reversal order according to Theorem 5.
CumProd = [1; cumprod(CalP)];
B = 0;
for j = 0:J-1;
```

```
% Computing the bit reversal permutation (8) according to (9).
  B = kron(B, ones(CalP(j+1),1))
   +CumProd(j+1)*kron(ones(CumProd(j+1),1),(0:CalP(j+1)-1)');

   for k = 0:CalN(j+2)-1;
      for l = 0:CalN(1)/CalN(j+1) -1;
         Indices = CalN(j+2)*(CalP(j+1)*l+(0:CalP(j+1) -1))+k;
         X1 = X(:,:,Indices+1);
         for p = 0:CalP(j+1) -1;
            X(:,:,Indices(p+1)+1)=zeros(size(X(:,:,Indices(p+1)+1)));
            for q = 0:CalP(j+1)-1;
              X(:,:,Indices(p+1)+1) = X(:,:,Indices(p+1)+1)
               +imrotate(X1(:,:,q+1),-CalN(j+2)*q*B(CalP(j+1)*l+p+1));
            end
         end
      end
   end
end

% Permuting the bit-reversed FGST into standard order.
for n=0:CalN(1)-1;
   index= find(B==n);
   X1 = X(:,:,index);
   X(:,:,index) = X(:,:,n+1);
   X(:,:,n+1) = X1;
   B(index) = B(n+1);
   B(n+1) = n;
end
```

Bibliography

1. Brigham, E. O. *The Fast Fourier Transform.* New Jersey: Prentice-Hall, 148–159, 1974.
2. Cochran, W. T., J. W. Cooley, D. L. Favin, H. D. Helms, R. A. Kaenel, W. W. Lang, et. al. "What Is the Fast Fourier Transform?" *Proceedings of the IEEE.* 55: 1664–1674, 1967.
3. Cooley, J. W. and J. W. Tukey. "An algorithm for the Machine Calculation of Complex Fourier Series." *Mathematics of Computation.* 19: 297–301, 1965.
4. Cooley, J. W., P. A. W. Lewis, and P. D. Welch. "Historical Notes on the Fast Fourier Transform." *Proceedings of the IEEE.* 55: 1675–1677, 1967.
5. Gentleman, W. M. and G. Sande. "Fast Fourier Transforms: for fun and profit." *1966 Fall Joint Computer Conf., AFIPS Proc.* 29: 563–578, 1966.
6. Heideman, M. T., D. H. Johnson, and C. S. Burrus. "Gauss and the History of the Fast Fourier Transform." *IEEE ASSP Magazine.* 14–21 (October 1984).
7. Landau, H. J. "Sampling, Data Transmission, and the Nyquist Rate." *Proceedings of the IEEE.* 55: 1701–1706, 1967.
8. Liu, D. "Two-Dimensional Time-Dependent Plasma Structures of a Hall Effect Thruster." Ph.D. Thesis. Air Force Institute of Technology. 2011.
9. Mohlenkamp, M. J. "A fast transform for spherical harmonics." *J. Fourier Anal. Appl.* 5 (2-3): 159–184, 1999.
10. Rader, C. M. "Discrete Fourier transforms when the number of data samples is prime." *Proc. IEEE* 56: 1107–1108, 1968.
11. Ramirez, R. W. *The FFT Fundamentals and Concepts.* New Jersey: Prentice-Hall, 124–125, 1985.
12. Rokhlin, V., Tygert, M. "Fast algorithms for spherical harmonic expansions." *SIAM J. Sci. Computing.* 27 (6): 1903–1928, 2006.
13. Van Loan, C. *Computational frameworks for the fast Fourier transform.* Pennsylvania: SIAM, 1992.
14. Walker, J. S. *Fast Fourier Transforms.* Florida: CRC Press, 1991.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
22 Mar 2012	Master's thesis	Aug 2010–Mar 2012

4. TITLE AND SUBTITLE

Determining angular frequency from video with a generalized fast Fourier transform

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Smith, Lindsay N, 2d Lt

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (EN)
2950 Hobson Way
Wright-Patterson AFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GAM/ENC/12-02

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Intentionally Left Blank

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Suppose we are given a video of a rotating object and suppose we want to determine the rate of rotation solely from the video itself and its known frame rate. In this thesis, we present a new mathematical operator called the Geometric Sum Transform (GST) that can help one determine the angular frequency of the object in question. The GST is a generalization of the discrete Fourier transform (DFT) and as such, the two transforms have much in common. However, whereas the DFT is applied to a sequence of scalars, the GST can be applied to a sequence of vectors. Most importantly, we show that the GST, like the DFT, can (1) be used to estimate frequency and (2) can be computed surprisingly quickly. Indeed, we provide a Fast Geometric Sum Transform (FGST) algorithm that computes the GST in $\mathcal{O}(N \log N)$ matrix-vector multiplications, where N is the number of images in the video sequence. This is a vast improvement over the $\mathcal{O}(N^2)$ such multiplications required for a direct computation of the GST. The remainder of this thesis is devoted to proving other properties of the GST and giving proof-of-concept numerical examples.

15. SUBJECT TERMS

Fourier transform, angular frequency, image processing, harmonic analysis

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Matthew C. Fickus
U	U	U	UU	48	19b. TELEPHONE NUMBER *(include area code)* (937) 255-3636 x 4513