Factorization of the translation kernel for fast rigid image alignment

Aaditya Rangan,[1] Marina Spivak,[2] Joakim Andén,[2] and Alex Barnett[2]

[1] Courant Institute, New York University, New York, NY and Center for Computational Mathematics, Flatiron Institute, New York, NY

E-mail: rangan@cims.nyu.edu

[2] Center for Computational Mathematics, Flatiron Institute, New York, NY

E-mail: mspivak@flatironinstitute.org, janden@flatironinstitute.org, abarnett@flatironinstitute.org

Abstract. An important component of many image alignment methods is the calculation of inner products (correlations) between an image of $n \times n$ pixels and another image translated by some shift and rotated by some angle. For robust alignment of an image pair, the number of considered shifts and angles is typically high, thus the inner product calculation becomes a bottleneck. Existing methods, based on fast Fourier transforms (FFTs), compute all such inner products with computational complexity $\mathcal{O}(n^3 \log n)$ per image pair, which is reduced to $\mathcal{O}(Nn^2)$ if only N distinct shifts are needed. We propose to use a factorization of the translation kernel (FTK), an optimal interpolation method which represents images in a Fourier–Bessel basis and uses a rank-H approximation of the translation kernel via an operator singular value decomposition (SVD). Its complexity is $\mathcal{O}(Hn(n+N))$ per image pair. We prove that $H = \mathcal{O}((W + \log(1/\epsilon))^2)$, where $2W$ is the magnitude of the maximum desired shift in pixels and ϵ is the desired accuracy. For fixed W this leads to an acceleration when N is large, such as when sub-pixel shift grids are considered. Finally, we present numerical results in an electron cryomicroscopy application showing speedup factors of 3–10 with respect to the state of the art.

Keywords: single-particle electron cryomicroscopy, rigid image alignment, Fourier–Bessel basis, interpolation, singular value decomposition

Submitted to: *Inverse Problems*

1. Introduction

Rigid alignment of images is a ubiquitous task that arises in many computer vision problems, such as motion tracking, video stabilization, summarization, image stitching and the creation of mosaics [36]. It is also widely used in the analysis of biomedical data. One such example, which serves as the motivation for this paper, is single particle electron cryomicroscopy (cryo-EM). For an overview of cryo-EM, see [6, 23, 32, 24, 8].

Within this context, the basic task is to compare (i.e., correlate) pairs of 2D images: each image is sampled on a uniform square pixel grid, and the goal is to find the rigid transformation—a rotation followed by a translation—of one image which maximizes its consistency with the other. This is necessary at several stages in the cryo-EM pipeline [40, 33, 31, 11, 39, 16, 43, 45, 20], and plays a major role in many popular software packages [28, 29, 14, 21, 13, 37]. Outside of cryo-EM, the rigid alignment problem also occurs in certain extensions of non-local means denoising [5, 19] where patches are aligned rotationally instead of just translationally [46, 34].

We consider the very common setting where the metric of comparison between images is their *inner product*, equivalent (up to normalization) to their cross-correlation. The challenge is then to solve the image-alignment problem to high (often sub-pixel) accuracy, robustly and rapidly. As Figure 1 illustrates, the inner product, viewed as a function of shift and angle, may have a complicated landscape with many local maxima. This is particularly true in the high-noise setting relevant to cryo-EM. Accurate and robust alignment thus requires computing inner products for a large set of transformations, which is expensive. In the literature, there are two general approaches for tackling this difficulty. Some methods conduct an exhaustive search over a grid of rotations and translations, calculating inner products for every transformation on this grid. Others take shortcuts by using various optimization methods to resolve only a portion of the full grid, yielding a higher resolution than permitted by an exhaustive calculation.

In this paper, we present a new strategy for solving this problem, which combines many of the advantages of the methods discussed above, and is an optimal interpolation of the inner product as a function of translation. We first review some of the correlation-based methods currently in use, and then describe our approach in greater detail.

Exhaustive methods. The simplest approach to the 2D image alignment problem is an exhaustive calculation of inner products over a dense grid of shifts and angles. The high grid density makes direct calculation computationally prohibitive (see Table 1), and several accelerations have been proposed [17].

Two widely used accelerations are based on the convolution theorem, which states that cross-correlation in real space is equivalent to multiplication in Fourier space. The first, which we refer to as "brute-force rotations" (BFR), accelerates the correlation across a full image-sized grid of translations through 2D FFT-accelerated convolutions, then steps through all rotated versions of each image via brute force. The second method, "brute-force translations" (BFT), uses the convolution theorem to treat rotations efficiently

via 1D FFTs along rotation angles, while stepping through all required translations by brute force [17, 1]. Their complexities are listed in the middle two rows of Table 1.

All such exhaustive methods have the advantage of resolving the full landscape of inner products on a pre-defined roto-translation grid, ensuring robust maximization in the case of multiple local maxima. Moreover, sampling this entire landscape is necessary in a Bayesian framework, where an integral of a probability distribution over shifts and angles is used to marginalize the likelihood [29, 28]. Unfortunately, these methods all become very costly when sub-pixel accuracy is desired for shifts and angles.

Optimization-based methods. In contrast to the exhaustive methods above, other methods try to find the best alignment quickly by iteratively maximizing the inner product as a function of shift and angle. Since this does not explore the entire landscape of inner products, it can be much faster.

One strategy alternates between maximization over rotation angles (fixing the shift) and shifts (fixing the angle) [27, 15]. Another method conducts an exhaustive search over a coarse grid to find an approximate maximum that is then refined [43].

In favorable situations (e.g., high signal-to-noise ratio images with dominant low-frequency content), these optimization-based methods quickly converge to the best alignment without being tethered to a pre-specified grid. Unfortunately, these methods may converge to a locally optimal alignment far from the true global optimum. In addition, because they only sample part of the inner product landscape, these methods are not immediately useful in a Bayesian setting.

Our approach. In this paper, we present the "factorization of the translation kernel" (FTK) method, a new algorithm which efficiently evaluates inner products over all rotation angles and over all shifts lying in a predetermined domain. It also enables exhaustive high density (sub-pixel) sampling at very little extra computational cost.

The essential ingredients in our approach are (i) the convolution theorem applied to rotation angles (as leveraged in BFT) and (ii) a truncated functional SVD of the translation kernel in the *Fourier–Bessel basis* compatible with this angular convolution. This allows us to form, and then evaluate, an optimal low-rank interpolant for the inner product landscape at all rotation angles and all shifts up to a certain magnitude.

The Fourier–Bessel basis we use to represent our images, along with many similar analytical tools, have also been used by Park *et al.* [26] for deblurring the class averages of previously aligned 2D images. This representation was also used by Barnett *et al.* [1] to assign viewing angles to images within the cryo-EM reconstruction pipeline.

Unlike many other exhaustive algorithms, our algorithm does not demand a pre-specified grid of translations. Instead, the speed of our algorithm depends on two parameters: the desired accuracy of the calculation and the maximum magnitude of the desired shifts. As the desired accuracy increases, the rank of the truncated SVD grows, giving a mild increase in overall computational cost. As the maximum shift magnitude increases, the structure of the corresponding set of translation operators becomes richer,

Algorithm	Effort per image–template pair
Direct method	$\mathcal{O}(Nn^3)$
Brute force rotations (BFR)	$\mathcal{O}(n^3 \log n)$
Brute force translations (BFT)	$\mathcal{O}(Nn^2)$
Proposed method (FTK)	$\mathcal{O}(Hn(n+N))$

Table 1. Complexity of some algorithms for rigid image alignment by computation of sets of translated and rotated inner products. The setting is that all N_I^2 pairwise alignments must be found between N_I images and N_I templates. For $N_I \gg 1$, precomputations that scale as N_I are negligible, so we give in the second column only the leading term scaling with N_I^2. Each image (and template) is $n \times n$ pixels. We assume that the desired number of rotation angles is $\mathcal{O}(n)$, while the number of translations is N. The parameter H is the total number of SVD terms for all the Bessel orders in the expansion (see Section 4.5); by Theorem 1 we have $H = \mathcal{O}((W + \log(1/\epsilon))^2)$, where $2W$ is the maximum translation magnitude in pixels, and ϵ is the desired accuracy.

and more terms in the SVD are required to achieve the same level of accuracy, also increasing the computational cost.

Importantly, given a fixed level of accuracy and range of translations, the computational cost of FTK scales with the number of non-negligible terms in the SVD. Once these terms are calculated and integrated against the given images, they can be used to recover the inner product values for a fine sub-pixel translation and rotation grid cheaply. Table 1 compares the complexity of our proposal to the other exhaustive direct, BFR, and BFT methods.

We demonstrate the utility of FTK by applying it to what is often the bottleneck in cryo-EM reconstruction: the task known as "template matching." This requires the inner product evaluation between all pairs of single-particle cryo-EM images and templates, over a range of sub-pixel translations and all in-plane rotations. We achieve a substantial speedup of a factor 3–10 over BFT or BFR methods for a wide range of translation grid sizes, while recovering at least two digits of accuracy (sufficient for most cryo-EM applications). We also show that, for higher accuracies, FTK is several orders of magnitude more efficient than the commonly-used method of linear interpolation from a translation grid.

The rest of this paper is organized as follows. Section 2 describes basic aspects of Fourier representation of images, and defines the inner product function. Section 3 reformulates this in the Fourier–Bessel basis and explains its numerical discretization. The main SVD algorithm is explained in the context of optimal interpolation schemes in Section 4. In Section 5 we prove a theorem bounding the rank growth in terms of the maximum translation in pixels and the desired relative mean squared error in the inner product calculation. Section 6 presents numerical results and a comparison with the BFR and BFT methods in the cryo-EM setting. In Section 7 we describe how FTK extends to other 2D kernels and to 3D volume alignment. We conclude in Section 8. A Python implementation is available at `https://github.com/flatironinstitute/ftk`.

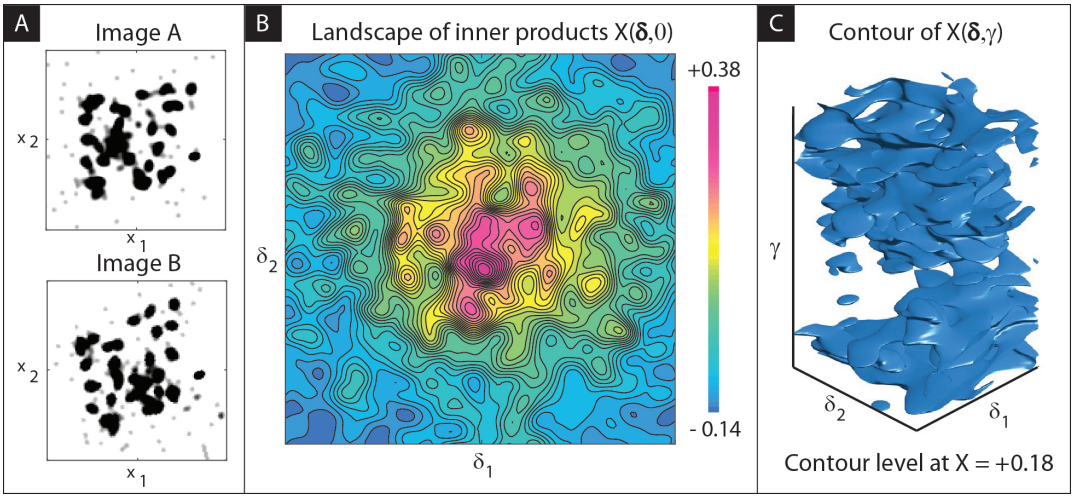

Figure 1. In panel A we show two images, $A(\boldsymbol{x})$ and $B(\boldsymbol{x})$, for $\boldsymbol{x} \in [-1,1]^2$. Each of these images is generated by summing several anisotropic Gaussian functions, and then normalizing so that the resulting image has L^2-norm 1. In panel B we show the inner product function $\mathcal{X}(\boldsymbol{\delta}, 0)$ (see (17)) obtained by translating the images with respect to one another. Note that the landscape is quite complicated, with multiple local maxima (see colorbar to the right). The 3D landscape obtained by including rotations is even more complex: one level set of $\mathcal{X}(\boldsymbol{\delta}, \gamma)$ is shown in panel C.

2. Problem setup and Fourier representation

We use the vectors $\boldsymbol{x}, \boldsymbol{\delta} \in \mathbb{R}^2$ to represent positions and displacements in the spatial domain, while $\boldsymbol{k} \in \mathbb{R}^2$ represents spatial frequencies. We define the polar coordinates of these vectors as follows:

$$\boldsymbol{x} = (x\cos\theta, x\sin\theta) \tag{1}$$

$$\boldsymbol{\delta} = (d\cos\omega, d\sin\omega) \tag{2}$$

$$\boldsymbol{k} = (k\cos\psi, k\sin\psi). \tag{3}$$

The Fourier transform of a two-dimensional function $A \in L^2(\mathbb{R}^2)$ is defined as

$$\hat{A}(\boldsymbol{k}) := \iint_{\mathbb{R}^2} A(\boldsymbol{x}) \mathrm{e}^{-\mathrm{i}\boldsymbol{k}\cdot\boldsymbol{x}}\, d\boldsymbol{x} \ . \tag{4}$$

The inner product between two functions $A, B \in L^2(\mathbb{R}^2)$ is written

$$\langle A, B\rangle = \iint_{\mathbb{R}^2} A(\boldsymbol{x}) B(\boldsymbol{x})^*\, d\boldsymbol{x} \ , \tag{5}$$

where z^* is the complex conjugate of $z \in \mathbb{C}$. We will also need Plancherel's theorem [4],

$$\langle A, B\rangle = \frac{1}{(2\pi)^2}\langle \hat{A}, \hat{B}\rangle \ , \qquad \forall A, B \in L^2(\mathbb{R}^2) \ . \tag{6}$$

We represent any image as a function $A \in L^2(\mathbb{R}^2)$, with values corresponding to the image intensity at each location. We recover A from $\hat{A}$ using the inverse Fourier transform [4]:

$$A(\boldsymbol{x}) = \frac{1}{(2\pi)^2} \iint_{\mathbb{R}^2} \hat{A}(\boldsymbol{k}) \mathrm{e}^{\mathrm{i}\boldsymbol{k}\cdot\boldsymbol{x}}\, d\boldsymbol{k} \, . \tag{7}$$

Let us denote the disk of radius $R > 0$ by

$$\Omega_R := \{\boldsymbol{x} \in \mathbb{R}^2 \text{ such that } \|\boldsymbol{x}\| \leq R\} \ . \tag{8}$$

We shall assume that all images are supported in Ω_1. This ensures that $\hat{A}$ is smooth on the $\mathcal{O}(1)$ frequency scale, enabling its approximate reconstruction from its values on grids with spacing $\mathcal{O}(1)$. We furthermore assume that $\hat{A}$ is concentrated in a disk of radius K, so that the inversion

$$A(\boldsymbol{x}) \approx \frac{1}{(2\pi)^2} \iint_{\Omega_K} \hat{A}(\boldsymbol{k}) \mathrm{e}^{\mathrm{i}\boldsymbol{k}\cdot\boldsymbol{x}} \, d\boldsymbol{k} \tag{9}$$

holds to desired working accuracy. These assumptions will ensure that we do not lose too much accuracy transitioning to a discrete setting.

Remark 1 *The maximum frequency, K, is related to the number of pixels across the image, n, as follows. The effective image support Ω_1 lives in $[-1, 1]$, so that the pixel spacing is $\Delta x = 2/n$, giving the Nyquist spatial frequency $K = \pi/\Delta x = (\pi/2)n$. For cryo-EM applications, n is typically* 100 *to* 400*.*

Remark 2 *In this paper, for simplicity, will assume that K is always the Nyquist frequency, so that scalings in K and n are equivalent. There do exist applications where K may be much lower than $(\pi/2)n$, such as frequency marching [1].*

We shall abuse notation slightly and write A and $\hat{A}$ in polar coordinates as

$$A(x, \theta) := A(\boldsymbol{x}) = A(x\cos\theta, x\sin\theta) \tag{10}$$

$$\hat{A}(k, \psi) := \hat{A}(\boldsymbol{k}) = \hat{A}(k\cos\psi, k\sin\psi). \tag{11}$$

With these reparametrizations, we may now define the rotation of A by $\gamma \in [0, 2\pi)$ as

$$R_\gamma A(x, \theta) := A(x, \theta - \gamma). \tag{12}$$

Since rotations commute with the Fourier transform, we have the same relationship in the frequency domain:

$$\widehat{R_\gamma A}(k, \psi) = R_\gamma \hat{A}(k, \psi) = \hat{A}(k, \psi - \gamma). \tag{13}$$

Let us similarly define translation of an image A by the shift vector $\boldsymbol{\delta} \in \mathbb{R}^2$ as

$$T_{\boldsymbol{\delta}} A(\boldsymbol{x}) := A(\boldsymbol{x} - \boldsymbol{\delta}). \tag{14}$$

In the Fourier domain, abusing notation slightly, we define the action of $T_{\boldsymbol{\delta}}$ on the Fourier transform $\hat{A}$ as giving the Fourier transform of $T_{\boldsymbol{\delta}} A$:

$$T_{\boldsymbol{\delta}} \hat{A}(\boldsymbol{k}) := \widehat{T_{\boldsymbol{\delta}} A}(\boldsymbol{k}) = F(\boldsymbol{\delta}, \boldsymbol{k}) \hat{A}(\boldsymbol{k}), \tag{15}$$

where the *translation kernel* $F(\boldsymbol{\delta}, \boldsymbol{k})$ is defined as

$$F(\boldsymbol{\delta}, \boldsymbol{k}) := \mathrm{e}^{-\mathrm{i}\boldsymbol{\delta}\cdot\boldsymbol{k}} = \mathrm{e}^{-\mathrm{i}\delta k \cos(\psi - \omega)}, \tag{16}$$

which is a plane wave in Fourier space. We introduce W as a dimensionless user-defined parameter that specifies the maximum translation magnitude considered, in wavelength units at the largest frequency K. When K is the Nyquist frequency, this

wavelength is $2\Delta x$. We thus have translations $\boldsymbol{\delta}$ of magnitude $\delta \le D$ (i.e., $\boldsymbol{\delta} \in \Omega_D$), where $D = 2\pi W/K = 2\Delta x W$.

Let $B \in L^2(\mathbb{R}^2)$ be the image against which A is to be aligned with respect to rotations and translations. The desired inner product $\iint_{\mathbb{R}^2} R_\gamma T_{\boldsymbol{\delta}} A(\boldsymbol{x})\, B(\boldsymbol{x})^*\, d\boldsymbol{x}$ is, up to a prefactor (which we drop), well approximated by

$$\mathcal{X}(\boldsymbol{\delta}, \gamma) \;:=\; \iint_{\Omega_K} R_\gamma T_{\boldsymbol{\delta}} \hat{A}(\boldsymbol{k})\, \hat{B}(\boldsymbol{k})^*\, d\boldsymbol{k} = \iint_{\Omega_K} F(\boldsymbol{\delta}, \boldsymbol{k})\, \hat{A}(\boldsymbol{k})\, R_{-\gamma} \hat{B}(\boldsymbol{k})\, d\boldsymbol{k} \;, \quad (17)$$

which follows from (6) and our assumption that $\hat{A}$ and $\hat{B}$ are concentrated in Ω_K. We call (17) the *bandlimited inner product*.

2.1. Conversion of discrete input images to the Fourier domain

The above formulation is described in terms of continuous images A and B. However, we are typically provided images sampled on a discrete grid of size $n \times n$ within $[-1, 1]^2$. Let us consider the pixel grid as representing the set of point samples

$$A_{ij} = A(\boldsymbol{x}_{ij}) \;, \qquad \boldsymbol{x}_{ij} := (i\Delta x - 1, j\Delta x - 1) \;, \quad i, j \in \{0, \dots, n-1\} \quad (18)$$

where $\Delta x = 2/n$ is the pixel spacing. We approximate its Fourier transform $\hat{A}$ at any $\boldsymbol{k} \in \mathbb{R}^2$ by the trapezoid quadrature rule

$$\hat{A}(\boldsymbol{k}) = (\Delta x)^2 \sum_{i,j=1}^{n} A_{ij} \mathrm{e}^{-\mathrm{i}\boldsymbol{k}\cdot\boldsymbol{x}_{ij}} \;. \quad (19)$$

This quadrature is expected to be accurate if the image is sufficiently well sampled at the resolution Δx (i.e., before sampling, its frequency content above the Nyquist frequency was negligible), and $\|\boldsymbol{k}\| \le K$. We similarly define $\hat{B}(\boldsymbol{k})$ given samples B_{ij} for $i, j \in \{0, \dots, n-1\}$. We will need to evaluate $\hat{A}(\boldsymbol{k})$ for $\boldsymbol{k}$ on various grids, which will be discussed later.

Plugging in these Fourier transforms $\hat{A}$ and $\hat{B}$ into (17), we obtain a set of inner products $\mathcal{X}(\boldsymbol{\delta}, \gamma)$ describing the alignment of the discrete images. The goal of this work is to provide an efficient and accurate method of calculating $\mathcal{X}(\boldsymbol{\delta}, \gamma)$ from given pixel values A_{ij}, B_{ij}, for a large number of user-specified values of $\boldsymbol{\delta} \in \Omega_D$ and $\gamma \in [0, 2\pi)$.

3. Fourier–Bessel decomposition

In this section we derive a formula for the inner product $\mathcal{X}(\boldsymbol{\delta}, \gamma)$ defined in (17) that separates the dependence on $\boldsymbol{\delta}$ from the dependence on $\boldsymbol{k}$. For this we first need to introduce the Fourier–Bessel basis representation of each image. At the end of the section we describe how these formulae are evaluated in practice given $n \times n$ pixel images.

For fixed k, $\hat{A}(k, \psi)$ is a 2π-periodic function of ψ. We may therefore decompose the function $\psi \mapsto \hat{A}(k, \psi)$ as a Fourier series, obtaining

$$\hat{A}(k, \psi) = \sum_{q=-\infty}^{\infty} a(k; q) \mathrm{e}^{\mathrm{i}q\psi}, \quad (20)$$

Its coefficients $a(k;q)$ are given by

$$a(k;q) = \frac{1}{2\pi}\int_0^{2\pi} \hat{A}(k,\psi)\mathrm{e}^{-\mathrm{i}q\psi}\,d\psi \ . \tag{21}$$

By substituting (4), changing variable to $\alpha = \theta + \pi/2 - \psi$, and using the integral form of the Bessel function [25, Eq. (10.9.1)],

$$J_n(z) = \frac{1}{2\pi}\int_0^{2\pi} \mathrm{e}^{\mathrm{i}(z\sin\alpha - n\alpha)}d\alpha \ , \tag{22}$$

it is easily verified that the coefficients are, in terms of the original spatial image,

$$a(k;q) = \int_0^{2\pi}\int_0^1 A(x,\theta)\ \left(\mathrm{e}^{\mathrm{i}q(\theta+\pi/2)}J_q(kx)\right)^*\ x dx d\theta \ . \tag{23}$$

This takes the form of an inner product of the image A with the 2D "Fourier–Bessel" function $u(x,\theta) = \mathrm{e}^{\mathrm{i}q(\theta+\pi/2)}J_q(kx)$, being a polar separation of variables solution to the Helmholtz equation $(\Delta + k^2)u = 0$ at frequency k, where Δ is the 2D Laplace operator. This justifies our naming (21) the *Fourier–Bessel coefficients* of the image. A similar basis is used by Zhao *et al.* [45, 44], who make different assumptions regarding the support of the images.

Remark 3 (Decay of Bessel functions) *The large-order asymptotics of $J_q(z)$ may be summarized as follows. For any fixed z, the formula $J_q(z) \sim (2\pi q)^{-1/2}(\mathrm{e}z/2q)^q$ shows super-exponential decay in q [25, Eq. (10.19.1)]. More precisely, for fixed real z, $J_q(z)$ is oscillatory and significant in the order range $q \le z$, and exponentially small for $q \ge z + \mathcal{O}(z^{1/3})$, the latter statement following from the Debye asymptotics [25, Eq. (10.19.8)]. Loosely speaking, this is associated with the "bump" of the Bessel function (as a function of z) centered at $z \approx q$ having width roughly $z^{1/3} \approx q^{1/3}$.*

The above remark, (23), and $x \le 1$ show that the coefficients $a(k;q)$ are exponentially small, and thus numerically negligible, for $|q| > K + \mathcal{O}(K^{1/3})$.

We now derive the actions of rotation and translation on these Fourier–Bessel coefficients. Applying (21) to the rotated Fourier transform $R_\gamma\hat{A}$, we obtain

$$\begin{aligned}\frac{1}{2\pi}\int_0^{2\pi} R_\gamma\hat{A}(k,\psi)\mathrm{e}^{-\mathrm{i}q\psi}\,d\psi &= \frac{1}{2\pi}\int_0^{2\pi}\hat{A}(k,\psi-\gamma)\mathrm{e}^{-\mathrm{i}q\psi}\,d\psi \\ = \frac{1}{2\pi}\int_0^{2\pi}\hat{A}(k,\psi)\mathrm{e}^{-\mathrm{i}q(\psi+\gamma)}\,d\psi &= \mathrm{e}^{-\mathrm{i}q\gamma}a(k;q).\end{aligned} \tag{24}$$

This shows that a rotation acts on the Fourier–Bessel coefficients a as a diagonal multiplication operator,

$$R_\gamma a(k,q) := \mathrm{e}^{-\mathrm{i}q\gamma}\,a(k;q) \ . \tag{25}$$

Here we abuse notation slightly by using the same symbol R_γ that we used for rotations in an angular variable. This diagonal action will allow us to simultaneously compute $\mathcal{X}(\boldsymbol{\delta},\gamma)$ for multiple values of γ in an efficient manner.

Let us similarly denote by $T_{\boldsymbol{\delta}} a(k;q)$ the action of translation by the vector $\boldsymbol{\delta}$ on the Fourier–Bessel coefficients a. Using (15), (16) and (21) we compute

$$\begin{aligned} T_{\boldsymbol{\delta}} a(k;q) &= \frac{1}{2\pi}\int_0^{2\pi} T_{\boldsymbol{\delta}} \hat{A}(k,\psi) \mathrm{e}^{-\mathrm{i}q\psi}\, d\psi = \frac{1}{2\pi}\int_0^{2\pi} \hat{A}(k,\psi) \mathrm{e}^{-\mathrm{i}k\delta\cos(\psi-\omega)} \mathrm{e}^{-\mathrm{i}q\psi}\, d\psi \\ &= \frac{1}{2\pi}\int_0^{2\pi} \hat{A}(k,\psi) \left(\sum_{\ell=-\infty}^{\infty} J_\ell(\delta k) \mathrm{e}^{\mathrm{i}\ell(\psi-\omega-\pi/2)} \right) \mathrm{e}^{-\mathrm{i}q\psi}\, d\psi \\ &= \sum_{\ell=-\infty}^{\infty} J_\ell(\delta k) \mathrm{e}^{-\mathrm{i}\ell(\omega+\pi/2)}\, a(k;q-\ell), \end{aligned} \tag{26}$$

where on the third line we used the Jacobi–Anger expansion [25, 10.12.3],

$$\mathrm{e}^{-\mathrm{i}z\cos\phi} = \sum_{\ell=-\infty}^{\infty} J_\ell(z) \mathrm{e}^{\mathrm{i}\ell(\phi-\pi/2)} \ . \tag{27}$$

In summary, translation acts as a 1D discrete convolution on the coefficients,

$$T_{\boldsymbol{\delta}} a(k,q) := \sum_{\ell=-\infty}^{\infty} f(\boldsymbol{\delta},k;\ell) a(k;q-\ell), \tag{28}$$

where $f(\boldsymbol{\delta},k;\ell)$ denotes the Fourier–Bessel translation kernel

$$f(\boldsymbol{\delta},k;\ell) \ := \ J_\ell(\delta k) \mathrm{e}^{-\mathrm{i}\ell(\omega+\pi/2)} \ , \tag{29}$$

being the ℓth 1D Fourier series coefficient of $F(\boldsymbol{\delta},\boldsymbol{k})$ on the ring $\|\boldsymbol{k}\| = k$ (see (16)). Note that the number of significant terms in the convolution is not large, and scales only like W, the size of the maximum translation in wavelengths. This follows from Remark 3, and (29), where the maximum Bessel argument of $DK = 2\pi W$ shows that $f_\ell(\boldsymbol{\delta},k)$ is exponentially small for $|\ell| > 2\pi W + \mathcal{O}((2\pi W)^{1/3})$.

It will turn out that each of the terms in the above convolution is amenable to factorization, which will help us consider multiple translations $\boldsymbol{\delta}$. Indeed the factors dependent on $\boldsymbol{\delta}$ (δ and ω) and those dependent on $\boldsymbol{k}$ (k and ψ) are *almost* separable from one another. The exception is the Bessel term $J_\ell(k\delta)$, which we return to in Section 4.5.

Now we turn to the desired inner product. From (21), on each fixed k ring we have the 1D Plancherel formula

$$\int_0^{2\pi} \hat{A}(k,\psi)\hat{B}(k,\psi)^* d\psi = 2\pi \sum_{q=-\infty}^{\infty} a(k;q) b(k;q)^* \ . \tag{30}$$

Using this we rewrite the inner product (17) as

$$\begin{aligned} \mathcal{X}(\boldsymbol{\delta},\gamma) &= \iint_{\Omega_K} R_\gamma T_{\boldsymbol{\delta}} \hat{A}(\boldsymbol{k})\, \hat{B}(\boldsymbol{k})^*\, d\boldsymbol{k} = 2\pi \sum_{q=-\infty}^{\infty} \int_0^K R_\gamma T_{\boldsymbol{\delta}} a(k;q)\, b(k;q)^*\, k dk \\ &= 2\pi \sum_{q=-\infty}^{\infty} \int_0^K \mathrm{e}^{-\mathrm{i}q\gamma} \left(\sum_{\ell=-\infty}^{\infty} f(\boldsymbol{\delta},k;\ell) a(k;q-\ell) \right) b(k;q)^*\, k dk \\ &= 2\pi \sum_{q=-\infty}^{\infty} \sum_{\ell=-\infty}^{\infty} \mathrm{e}^{-\mathrm{i}q\gamma} \int_0^K f(\boldsymbol{\delta},k;\ell) a(k;q-\ell) b(k;q)^*\, k dk \end{aligned} \tag{31}$$

While this expression appears more complicated than our original formula (17), we shall see that it lends itself more naturally to a fast numerical algorithm. For now we will simply remark that this expression displays the *trilinearity* of the inner product: $\mathcal{X}(\boldsymbol{\delta}, \gamma)$ is linear in $\hat{A}(\boldsymbol{k})$, $\hat{B}(\boldsymbol{k})$ and $f(\boldsymbol{\delta}, k; \ell)$. Our method will exploit this by reducing the approximation of $\mathcal{X}(\boldsymbol{\delta}, \gamma)$ to an approximation of the translation kernel $F(\boldsymbol{\delta}, k; \ell)$.

3.1. Discretization and truncation

To numerically approximate $\mathcal{X}(\boldsymbol{\delta}, \gamma)$ evaluated via (31), we make the following approximations. First, we truncate the infinite sums over q and ℓ to finite sums over $|q| \le Q$ and $|\ell| \le L$. Second, we replace the integral over k with a quadrature scheme designed on $[0, K]$. We prefer Gauss–Jacobi quadrature built with a weight function corresponding to the area-element kdk. This quadrature scheme uses Jacobi polynomials of type $\alpha = 0$, $\beta = 1$ (i.e. corresponding to weight k after a change of variable), and produces a set of nodes $k_m \in [0, K]$ and weights w_m for $m \in \{1, \dots, M\}$ such that

$$\int_0^K g(k)\, kdk \;\approx\; \sum_{m=1}^{M} g(k_m) w_m \tag{32}$$

holds to high accuracy for any radial function g that is smooth on the scale of $\mathcal{O}(1)$. In summary, we approximate $\mathcal{X}(\boldsymbol{\delta}, \gamma)$ by

$$X(\boldsymbol{\delta}, \gamma) = 2\pi \sum_{q=-Q}^{Q} \sum_{\ell=-L}^{L} \sum_{m=1}^{M} \mathrm{e}^{-\mathrm{i}q\gamma} a(k_m; q-\ell) f(\boldsymbol{\delta}, k_m; \ell) b(k_m; q)^* w_m. \tag{33}$$

We will not attempt a rigorous convergence analysis here, but rather justify the empirical sizes of the convergence parameters Q, L, and M needed. As discussed in the previous section, following Remark 3, Q need only slightly exceed $K = (\pi/2)n$. Similarly, L need only slightly exceed $2\pi W$. For quadrature, in order to capture the $\mathcal{O}(K)$ oscillations in the integrand, we need $M = \mathcal{O}(K) = \mathcal{O}(n)$. Once M is large enough to resolve these oscillations, the convergence becomes super-algebraic; this follows because the integrands (deriving from Fourier transforms of images with compact support) are analytic. It is thus easy to choose the parameters to achieve a desired accuracy (see Section 6).

Remark 4 *We could reduce the number of operations in the above calculation by making Q depend on k. In particular, we expect Q to vary proportionally to k, as it represents the number of angular modes a ring of circumference $2\pi k$. This would reduce the total number of Fourier–Bessel coefficients by a factor of two. However, this reduction does not necessarily translate to shorter running times, as the resulting triangular sum cannot easily take advantage of vectorized libraries for FFTs and matrix multiplication.*

3.2. Decomposition of discrete images

The approximate inner product (33) is defined using the Fourier–Bessel coefficients $a(k_m, q)$ and $b(k_m, q)$ on all radial quadrature nodes $\{k_m\}_{m=1}^{M}$, for all indices $|q| \le Q$. This is $\mathcal{O}(n^2)$ coefficients. Given $n \times n$ pixel images $\{A_{ij}\}$ and $\{B_{ij}\}$ as in Section 2.1,

we compute these coefficients in optimal complexity as follows. For each image we apply the 2D type 2 non-uniform FFT (NUFFT) [7, 12, 18] (we use the FINUFFT implementation [2]) to evaluate the Fourier transform (19) on a tensor-product polar $\boldsymbol{k}$ grid $\{(k_m, \psi_p)\}_{m=1,\dots,M,\ p=1,\dots,2Q}$ where the uniform angular grid has $\psi_p := \pi(p-1)/Q$. The NUFFT cost depends weakly on a tolerance parameter ϵ, which we ensure is smaller than the desired overall accuracy. The complexity of this step is $\mathcal{O}(n^2 \log n + n^2 \log(1/\epsilon))$.

Finally, on each quadrature ring $m = 1, \dots, M$, we approximate (21) using the $\hat{A}$ values on that ring with the periodic trapezoid rule,

$$a(k;q) = \frac{1}{2\pi} \int_0^{2\pi} \hat{A}(k,\psi) \mathrm{e}^{-\mathrm{i}q\psi}\, d\psi \approx \frac{1}{2Q} \sum_{p=1}^{2Q} \hat{A}(k,\psi_p) \mathrm{e}^{-\mathrm{i}q\psi_p} \ . \tag{34}$$

Since $\psi_p = \pi(p-1)/Q$, this takes the form of a 1D discrete Fourier transform of length $2Q$, which we evaluate using the FFT. The total cost of these $\mathcal{O}(n)$ 1D FFTs is $\mathcal{O}(n^2 \log n)$. In summary, fixing ϵ, the entire precomputation then scales as $\mathcal{O}(n^2 \log n)$ per image.

4. Fast computation of multiple inner products

In this section, we present an SVD interpolation method for computing many translated and rotated inner products (33). For simplicity we consider one image A and one template B. To provide intuition, we start with an algorithm in which translations are treated independently, explain interpolation with respect to translations, then explain our method as an optimal version of such an interpolation.

4.1. Multiple rotation angles

We first present a variant of the BFT algorithm. If we only have a single shift vector $\boldsymbol{\delta}$, we can calculate $X(\boldsymbol{\delta}, \gamma)$ efficiently for a large number of γ values by taking advantage of the diagonal action of R_γ in the Fourier–Bessel coefficients. Indeed, we have

$$X(\boldsymbol{\delta}, \gamma) = 2\pi \sum_{q=-Q}^{Q} \mathrm{e}^{-\mathrm{i}q\gamma} \sum_{m=1}^{M} w_m b(k_m; q)^* \sum_{\ell=-L}^{L} a(k_m; q-\ell) f(\boldsymbol{\delta}, k_m; \ell). \tag{35}$$

This calculation can be achieved in two steps:

$$\textbf{[Step 1]} \quad \hat{X}(\boldsymbol{\delta}, q) = 2\pi \sum_{m=1}^{M} w_m b(k_m; q)^* \sum_{\ell=-L}^{L} a(k_m; q-\ell) f(\boldsymbol{\delta}, k_m; \ell) \tag{36}$$

$$\textbf{[Step 2]} \quad X(\boldsymbol{\delta}, \gamma) = \sum_{q=-Q}^{Q} \mathrm{e}^{-\mathrm{i}q\gamma} \hat{X}(\boldsymbol{\delta}, q). \tag{37}$$

The first step computes the Fourier coefficients $\hat{X}(\boldsymbol{\delta}, q)$, requiring $\mathcal{O}(MQL)$ operations. If we restrict γ to the set $\{0, \pi/Q, \dots, 2\pi(2Q-1)/Q\}$, we may evaluate the second step using a 1D FFT of size $2Q$. The first step dominates, so the total complexity is $\mathcal{O}(n^2 W)$, where we have used $Q = \mathcal{O}(n)$, $M = \mathcal{O}(n)$ and $L = \mathcal{O}(W)$ (see previous section). By zero-padding and using a larger FFT (or NUFFT) in the second step, a large number of

γ values may be sampled essentially for free. A naive method to now handle N different translations $\boldsymbol{\delta} \in \Omega_D$ is to repeat the above two steps, separately for each translation, giving a cost $\mathcal{O}(Nn^2W)$. For each image-template pair, this is W times slower than BFT; see Table 1. However, the above steps, with $f(\boldsymbol{\delta}, k_m; \ell)$ replaced by other functions, will serve as a building block for the much faster algorithm described in the following subsections.

4.2. Linear interpolation over translations

We now present a simple way to accelerate the brute force inner product calculation over multiple translations. Instead of evaluating $X(\boldsymbol{\delta}, \gamma)$ at all values of $\boldsymbol{\delta}$ which are of interest to us, one can use the fact that, for fixed γ, the mapping $\boldsymbol{\delta} \mapsto X(\boldsymbol{\delta}, \gamma)$ is smooth. Indeed, it is the smoothness of the underlying image functions A and B induces smoothness in X. A common approach is to calculate $X(\boldsymbol{\delta}, \gamma)$ at some subset of shifts and use local (low-order) interpolation to obtain its value at other locations.

Let us denote such a set of H interpolation nodes by $\{\boldsymbol{\delta}_1, \ldots, \boldsymbol{\delta}_H\} \subset \Omega_D$. This could be a grid (e.g., Cartesian or polar) covering Ω_D; we assume it is independent of γ. Fixing γ for now, we first compute $X(\boldsymbol{\delta}_\zeta, \gamma)$ for $\zeta \in \{1, \ldots, H\}$, using, for instance the method outlined in the previous section. For other values of $\boldsymbol{\delta}$ at this γ, we interpolate

$$X^{\text{lin}}(\boldsymbol{\delta}, \gamma) = \sum_{\zeta=1}^{H} Y_\zeta^{\text{lin}}(\boldsymbol{\delta}) X(\boldsymbol{\delta}_\zeta, \gamma), \tag{38}$$

where $Y_\zeta^{\text{lin}}(\boldsymbol{\delta})$ is the interpolation weight (basis function) associated with $\boldsymbol{\delta}$ and $\boldsymbol{\delta}_\zeta$. In the bilinear case, it is nonzero only for the four nodes $\boldsymbol{\delta}_\zeta$ that are nearest to $\boldsymbol{\delta}$. The above is repeated for the $\mathcal{O}(K)$ desired γ values. Since the complexity for computing $X(\boldsymbol{\delta}_\zeta, \gamma)$ is $\mathcal{O}(n^2W)$ (see Section 4.1), the overall complexity is $\mathcal{O}(Hn^2W)$ for all ζ (this drops to $\mathcal{O}(Hn^2)$ if we instead use the BFT, see Table 1 and Section 6.2). The cost of local interpolation is $\mathcal{O}(1)$ per lookup, or $\mathcal{O}(Nn)$ per image-template pair. This is advantageous since the original Nn^2 factor is gone. However, to achieve high interpolation accuracy in a low-order scheme, a fine grid of nodes would be needed, hence a large H.

4.3. Least-squares interpolation

We may now ask: can we increase the accuracy without significantly increasing the computational cost? First note that the interpolated result can be rewritten by moving the sum over ζ inside the other sums:

$$X^{\text{lin}}(\boldsymbol{\delta}, \gamma) = 2\pi \sum_{q=-Q}^{Q} \mathrm{e}^{-\mathrm{i}q\gamma} \sum_{m=1}^{M} w_m b(k_m; q)^* \sum_{\ell=-L}^{L} a(k_m; q-\ell) \sum_{\zeta=1}^{H} Y_\zeta(\boldsymbol{\delta}) f(\boldsymbol{\delta}_\zeta, k_m; \ell). \tag{39}$$

Comparing this to the formula for $X(\boldsymbol{\delta}, \gamma)$ in (35), we see that linear interpolation is equivalent to the approximation

$$\sum_{\zeta=1}^{H} Y_\zeta^{\text{lin}}(\boldsymbol{\delta}) f(\boldsymbol{\delta}_\zeta, k; \ell) \;\approx\; f(\boldsymbol{\delta}, k; \ell), \quad \forall k \in [0, K], \ell \in \mathbb{Z}. \tag{40}$$

In other words, due to the linearity of the bandlimited inner product $X(\boldsymbol{\delta}, \gamma)$, an interpolation scheme over $X(\boldsymbol{\delta}, \gamma)$ corresponds to an interpolation scheme over $f(\boldsymbol{\delta}, k; \ell)$.

Fixing $\boldsymbol{\delta}$, we now present a scheme to find the vector of interpolation weights $Y^{\mathrm{LS}}(\boldsymbol{\delta}) := (Y_1^{\mathrm{LS}}(\boldsymbol{\delta}), \ldots, Y_H^{\mathrm{LS}}(\boldsymbol{\delta}))$ that best approximates $f(\boldsymbol{\delta}, k; \ell)$ in a least-squares sense. Equation (39) suggests that one should minimize the sum-of-squares error over the quadrature nodes $k_1, \ldots, k_M$ and angular modes $-L \leq \ell \leq L$; however, treating k as a continuous variable is cleaner, gives essentially the same answer (since the quadrature scheme is accurate), and will allow analysis relevant for the SVD method.

The weights vector is the least-squares solution

$$Y^{\mathrm{LS}}(\boldsymbol{\delta}) = \operatorname*{arg\,min}_{Y=(Y_1,\ldots,Y_H)} \sum_{\ell=-\infty}^{\infty} \int_0^K \left| \sum_{\zeta=1}^{H} Y_\zeta f(\boldsymbol{\delta}_\zeta, k; \ell) - f(\boldsymbol{\delta}, k; \ell) \right|^2 k dk \ . \tag{41}$$

The corresponding normal equations for Y^{LS} form the $H \times H$ linear system

$$\sum_{\zeta=1}^{H} \mathcal{M}^{\mathrm{LS}}(\boldsymbol{\delta}_{\zeta'}, \boldsymbol{\delta}_\zeta) Y_\zeta^{\mathrm{LS}}(\boldsymbol{\delta}) = \mathcal{M}^{\mathrm{LS}}(\boldsymbol{\delta}_{\zeta'}, \boldsymbol{\delta}) \ , \qquad \forall \zeta' \in \{1, \ldots, H\}, \tag{42}$$

where

$$\mathcal{M}^{\mathrm{LS}}(\boldsymbol{\delta}', \boldsymbol{\delta}) := \sum_{\ell=-\infty}^{\infty} \mathrm{e}^{\mathrm{i}\ell(\omega'-\omega)} \int_0^K J_\ell(\delta' k) J_\ell(\delta k) \, k dk \, . \tag{43}$$

These integrals may be calculated analytically:

$$\mathcal{M}^{\mathrm{LS}}(\boldsymbol{\delta}', \boldsymbol{\delta}) = J_0(\tilde{\delta} K) + J_1(\tilde{\delta} K) \ , \tag{44}$$

where $\tilde{\delta} = |\boldsymbol{\delta}' - \boldsymbol{\delta}|$. The resulting linear system is well-conditioned for moderate values of D and K (and hence W), so $Y_\zeta^{\mathrm{LS}}(\boldsymbol{\delta})$ may be calculated to high precision using standard numerical linear algebra techniques. This is an image-independent precomputation required for each $\boldsymbol{\delta}$. Given the same set of $X(\boldsymbol{\delta}_\zeta, \gamma)$ samples (38), the interpolator is now

$$X^{\mathrm{LS}}(\boldsymbol{\delta}, \gamma) = \sum_{\zeta=1}^{H} Y_\zeta^{\mathrm{LS}}(\boldsymbol{\delta}) X(\boldsymbol{\delta}_\zeta, \gamma) \ . \tag{45}$$

This increases the lookup cost from $\mathcal{O}(1)$ per shift to $\mathcal{O}(H)$, but achieves higher accuracy.

We can improve the accuracy by optimizing the node locations $\{\boldsymbol{\delta}_1, \ldots, \boldsymbol{\delta}_H\}$ using, for example, gradient descent. The gradient is readily calculated by noting that

$$\partial_{\tilde{\delta} K} \mathcal{M}^{\mathrm{LS}}(\boldsymbol{\delta}', \boldsymbol{\delta}) = \frac{1}{2} J_{-1}(\tilde{\delta} K) - \frac{1}{2} J_3(\tilde{\delta} K) \ . \tag{46}$$

While this works well for a small number of nodes, this nonconvex optimization problem becomes intractable for larger H, limiting applicability to lower accuracies ($\epsilon \geq 10^{-2}$).

4.4. Generalized least-squares interpolation

One could potentially increase the accuracy of the above method by allowing a *different* vector of interpolation weights $Y_\zeta^{\mathrm{GLS}}(\boldsymbol{\delta}; \ell) := (Y_\zeta^{\mathrm{GLS}}(\boldsymbol{\delta}; \ell))_{\zeta=1}^{H}$ for each mode

$\ell \in \{-L, \dots, L\}$. We refer to this as the *generalized least-squares interpolation* (GLS) method. We now solve

$$Y^{\mathrm{GLS}}(\boldsymbol{\delta};\ell) = \operatorname*{arg\,min}_{Y=(Y_1,\dots,Y_H)} \int_0^K \left| \sum_{\zeta=1}^{H} Y_\zeta f(\boldsymbol{\delta}_\zeta, k;\ell) - f(\boldsymbol{\delta}, k;\ell) \right|^2 k dk\,, \tag{47}$$

Using (29) we see that $Y_\zeta^{\mathrm{GLS}}(\boldsymbol{\delta};\ell) = \mathrm{e}^{-\mathrm{i}\ell(\omega-\omega_\zeta)} U_\zeta^{\mathrm{GLS}}(\delta;\ell)$, where

$$U_\zeta^{\mathrm{GLS}}(\delta;\ell) = \operatorname*{arg\,min}_{U=(U_1,\dots,U_H)} \int_0^K \left| \sum_{\zeta=1}^{H} U_\zeta J_\ell(\delta_\zeta k) - J_\ell(\delta k) \right|^2 k dk\,. \tag{48}$$

As before, $U_\zeta^{\mathrm{GLS}}(\delta;\ell)$ satisfies the normal equations

$$\sum_{\zeta=1}^{H} \mathcal{M}^{\mathrm{GLS}}(\delta_{\zeta'}, \delta_\zeta;\ell) U_\zeta^{\mathrm{GLS}}(\delta;\ell) = \mathcal{M}^{\mathrm{GLS}}(\delta_{\zeta'}, \delta;\ell) \quad \forall \zeta' \in \{1,\dots,H\}, \tag{49}$$

where

$$\mathcal{M}^{\mathrm{GLS}}(\delta', \delta;\ell) = \int_0^K J_\ell(\delta' k) J_\ell(\delta k)\, k dk\,. \tag{50}$$

As before, (49) can be readily solved by analytically evaluating (50) and applying standard numerical techniques.

Given a set of weights $Y_\zeta^{\mathrm{GLS}}(\boldsymbol{\delta};\ell)$, we thus have the steps:

$$\textbf{[Step 1]} \quad \hat{Z}_\zeta^{\mathrm{GLS}}(q,\ell) = 2\pi \sum_{m=1}^{M} w_m b(k_m; q)^* a(k_m; q-\ell) f(\boldsymbol{\delta}_\zeta, k_m;\ell) \tag{51}$$

$$\textbf{[Step 2]} \quad Z_\zeta^{\mathrm{GLS}}(\gamma;\ell) = \sum_{q=-Q}^{Q} \mathrm{e}^{-\mathrm{i}q\gamma} \hat{Z}_\zeta^{\mathrm{GLS}}(q,\ell) \tag{52}$$

$$\textbf{[Step 3]} \quad X^{\mathrm{GLS}}(\boldsymbol{\delta},\gamma) = \sum_{\ell=-L}^{L} \sum_{\zeta=1}^{H} Y_\zeta^{\mathrm{GLS}}(\boldsymbol{\delta};\ell) Z_\zeta(\gamma;\ell). \tag{53}$$

Another advantage over the plain least-squares interpolation method of Section 4.3 is that a different H could be used for each ℓ; for instance, from (29) and Remark 3 one could use fewer nodes for large $|\ell|$ where $J_\ell(k\delta)$ is small. In the next subsection, rather than pursue least-squares methods, we use this intuition to understand an improved SVD-based method.

4.5. Proposed SVD interpolation method: FTK

In the previous section, we optimized weights $Y_\zeta(\boldsymbol{\delta};\ell)$ to minimize the mean squared error between $f(\boldsymbol{\delta}, k;\ell)$ and $\sum_{\zeta=1}^{H} Y_\zeta(\boldsymbol{\delta};\ell) f(\boldsymbol{\delta}_\zeta, k;\ell)$. To further reduce the error, we need to optimize the second factor, $f(\boldsymbol{\delta}_\zeta, k;\ell)$. As discussed in Section 4.5, we could optimize the interpolation nodes, but this is only tractable for low accuracies.

Treating the interpolant samples $f(\boldsymbol{\delta}_\zeta, k;\ell)$ as functions of k, we now replace them by a general set of functions $G_\eta(k;\ell)$, where $\eta = 1,\dots,H_\ell$ indexes the ℓth set. Independently for each mode ℓ, we choose the set $\{G_\eta(k;\ell)\}_{\eta=1}^{H_\ell}$ that optimally interpolates $f(\cdot, k;\ell)$

over the disk Ω_D in the mean-square sense. Not only is this optimization problem more general than the least-squares problems above, but, as it turns out, it is also easier to solve. Thus, for each ℓ, we minimize the mean squared error, which is the squared *Hilbert–Schmidt* (HS) norm of the residual,

$$\begin{aligned} E_{\mathrm{HS}}(\ell)^2 &:= \int_{\Omega_D}\int_0^K \left|\sum_{\eta=1}^{H_\ell} Y_\eta(\boldsymbol{\delta};\ell)G_\eta(k;\ell) - f(\boldsymbol{\delta},k;\ell)\right|^2 k\,dk\,d\boldsymbol{\delta} \\ &= \int_{\Omega_D}\int_0^K \left|\sum_{\eta=1}^{H_\ell} Y_\eta(\boldsymbol{\delta};\ell)G_\eta(k;\ell) - J_\ell(\delta k)\mathrm{e}^{-\mathrm{i}\ell(\omega+\pi/2)}\right|^2 k\,dk\,d\boldsymbol{\delta} \\ &= \int_{\Omega_D}\int_0^K \left|\sum_{\eta=1}^{H_\ell} Y_\eta(\boldsymbol{\delta};\ell)\mathrm{e}^{\mathrm{i}\ell(\omega+\pi/2)}G_\eta(k;\ell) - J_\ell(\delta k)\right|^2 k\,dk\,d\boldsymbol{\delta} \\ &= 2\pi\int_0^D\int_0^K \left|\sum_{\eta=1}^{H_\ell} U_\eta(\delta;\ell)G_\eta(k;\ell) - J_\ell(\delta k)\right|^2 k\,dk\,\delta d\delta\,. \end{aligned} \tag{54}$$

Here we substituted (29), then noted that since $J_\ell(\delta k)$ does not depend on the angle ω, the $Y_\eta(\boldsymbol{\delta};\ell)$ that minimizes the error must be such that the product $Y_\eta(\boldsymbol{\delta};\ell)\mathrm{e}^{\mathrm{i}\ell(\omega+\pi/2)}$ also does not depend on ω. We therefore replace this product by $U_\eta(\delta;\ell)$ (which only depends on the magnitude δ) and set $Y_\eta(\boldsymbol{\delta};\ell) = U_\eta(\delta;\ell)\mathrm{e}^{-\mathrm{i}\ell(\omega+\pi/2)}$. As a result, we seek a rank-H_ℓ separable approximation of the bivariate function $J_\ell(\delta k) : [0,D]\times[0,K]\to\mathbb{R}$ which is mean-square optimal under the linear weight function. Treating this bivariate function as the kernel of a (compact) integral operator, the solution is to truncate to the first H_ℓ terms in the operator SVD [9, Chap. II]. As proven by Schmidt in 1907 [30, §18] (see [35]), the truncated SVD is the optimal fixed-rank approximation to an operator in the Hilbert–Schmidt norm. With our weight function, for each ℓ, the operator SVD is

$$J_\ell(\delta k) \;=\; \sum_{\eta=1}^{\infty} U_\eta(\delta;\ell)\Sigma_\eta(\ell)V_\eta(k;\ell)\;, \qquad \delta\in[0,D], \quad k\in[0,K]\;, \tag{55}$$

where $\Sigma_1(\ell)\geq\Sigma_2(\ell)\geq\ldots\to 0$ are the singular values, the left singular functions $\{U_\eta(\delta;\ell)\}_{\eta=1}^{\infty}$ are orthonormal on $[0,D]$ with respect to $\delta d\delta$, and the right singular functions $\{V_\eta(k;\ell)\}_{\eta=1}^{\infty}$ are orthonormal on $[0,K]$ with respect to $k dk$. Although the singular values appear to depend on both K and D, one may check by changing variables that in fact they depend solely on their product $KD = 2\pi W$. Figure 2 shows the rapid decay of these singular values, and Figure 3 the first few singular functions.

Remark 5 (Approximating the operator SVD) *In practice, to compute the SVD, we use a tensor-product quadrature scheme to project $J_\ell(\delta k)$ onto a product basis of orthogonal Jacobi polynomials with $\alpha = 0$ and $\beta = 1$, rescaled onto the intervals $\delta\in[0,D]$ and $k\in[0,K]$. These Jacobi bases are orthonormal with respect to $\delta d\delta$ and $k dk$, respectively. Given the matrix of coefficients representing J_ℓ in this basis, we compute its SVD using standard dense numerical linear algebra [10, 38]. The left and right singular functions $U_\eta(\delta;\ell)$ and $V_\eta(\delta;\ell)$ are then given by summing the Jacobi bases with the corresponding singular vector entries as coefficients.*

Remark 6 (Alternative weight functions) *In this paper we calculate the SVD of each J_ℓ with the radial weights $\delta d\delta$ and kdk, corresponding to L^2 approximation over the disks Ω_D and Ω_K. One can easily generalize this to accommodate alternative weight functions. For example, when processing images of known power-spectral decay, one should tailor the k-weight to emphasize the lower frequencies. This would allow a smaller number of terms H_ℓ to reach a given accuracy for inner products.*

Truncating (55) to H_ℓ terms, we get,

$$J_\ell(\delta k) \;\approx\; \sum_{\eta=1}^{H_\ell} U_\eta(\delta;\ell)\Sigma_\eta(\ell)V_\eta(k;\ell)\;, \qquad \delta\in[0,D], \quad k\in[0,K]\;, \tag{56}$$

from which we construct, for each ℓ,

$$Y_\eta^{\mathrm{SVD}}(\boldsymbol{\delta};\ell) = U_\eta(\delta;\ell)\mathrm{e}^{-\mathrm{i}\ell(\omega+\pi/2)}\;, \qquad G_\eta(k;\ell) = \Sigma_\eta(\ell)V_\eta(k;\ell)\;. \tag{57}$$

Note that these functions only depend on W. Given these, the proposed algorithm is the following:

$$\textbf{[Step 1]} \quad \hat{Z}_\eta^{\mathrm{SVD}}(q,\ell) = 2\pi\sum_{m=1}^{M} w_m b(k_m;q)^* a(k_m;q-\ell)G_\eta(k_m;\ell) \tag{58}$$

$$\textbf{[Step 2]} \quad Z_\eta^{\mathrm{SVD}}(\gamma;\ell) = \sum_{q=-Q}^{Q} \mathrm{e}^{-\mathrm{i}q\gamma}\hat{Z}_\eta^{\mathrm{SVD}}(q,\ell) \tag{59}$$

$$\textbf{[Step 3]} \quad X^{\mathrm{SVD}}(\boldsymbol{\delta},\gamma) = \sum_{\ell=-L}^{L}\sum_{\eta=1}^{H_\ell} Y_\eta^{\mathrm{SVD}}(\boldsymbol{\delta};\ell)Z_\eta^{\mathrm{SVD}}(\gamma;\ell). \tag{60}$$

The first step is computed directly, requiring $\mathcal{O}(HQM)$ operations, where $H := \sum_\ell H_\ell$ is the total number of terms. The second step is computed with FFTs, yielding a complexity of $\mathcal{O}(HQ\log Q)$. The third step is also computed directly with $\mathcal{O}(NHQ)$ operations. Using $Q=\mathcal{O}(n)$ and $M=\mathcal{O}(n)$, the overall computational complexity of this scheme is therefore $\mathcal{O}(Hn^2+NHn)$.

One key point here is the separation of $J_\ell(\delta k)$ into factors depending either on δ or on k. This separation of variables allows us to efficiently calculate the sums, and the SVD accomplishes this separation with minimal error.

Another key point is that the procedure above—namely calculating a separate SVD for each $J_\ell(\delta k)$—is actually equivalent to a single SVD of the original translation kernel $F(\boldsymbol{\delta},\boldsymbol{k})$, as we now show. Recalling the Jacobi–Anger expansion (27), and (55),

$$\begin{aligned} F(\boldsymbol{\delta},\boldsymbol{k}) = \mathrm{e}^{-\mathrm{i}\boldsymbol{\delta}\cdot\boldsymbol{k}} = \mathrm{e}^{-\mathrm{i}\delta k\cos(\psi-\omega)} &= \sum_{\ell\in\mathbb{Z}} J_\ell(\delta k)\mathrm{e}^{\mathrm{i}\ell(\psi-\omega-\pi/2)} \\ &= \sum_{\ell\in\mathbb{Z}} \mathrm{e}^{\mathrm{i}\ell(\psi-\omega-\pi/2)}\sum_{\eta=1}^{\infty} U_\eta(\delta;\ell)\Sigma_\eta(\ell)V_\eta(k;\ell). \end{aligned} \tag{61}$$

We now relabel the singular values $\Sigma_\eta(\ell)$ from different modes ℓ with a single new index $\zeta=1,2,\ldots$, and refer to the ℓ and η corresponding to each index ζ as $\ell(\zeta)$ and $\eta(\zeta)$. Specifically, we define

$$U_\zeta(\delta) = U_{\eta(\zeta)}(\delta,\ell(\zeta)), \qquad \Sigma_\zeta = \Sigma_{\eta(\zeta)}(\ell(\zeta)), \qquad V_\zeta(k) = V_{\eta(\zeta)}(k,\ell(\zeta)). \tag{62}$$

This relabeling is chosen to have non-increasing ordering of singular values, i.e. $\Sigma_\zeta \geq \Sigma_{\zeta+1}$ for all $\zeta = 1, 2, \ldots$. If H_ℓ is chosen such that $\Sigma_{H_\ell+1}(\ell) \leq \epsilon$ for each ℓ, then by construction $\Sigma_{H+1} \leq \epsilon$. Truncation at a total number of terms H then involves all singular values larger than ϵ, and gives

$$\begin{aligned} F(\boldsymbol{\delta}, \boldsymbol{k}) \approx F^{\mathrm{SVD}}(\boldsymbol{\delta}, \boldsymbol{k}) &= \sum_{\zeta=1}^{H} \mathrm{e}^{\mathrm{i}\ell(\zeta)\psi}\, \mathrm{e}^{-\mathrm{i}\ell(\zeta)(\omega+\pi/2)}\, U_\zeta(\delta)\Sigma_\zeta V_\zeta(k) \\ &= \sum_{\zeta=1}^{H} \mathrm{e}^{-\mathrm{i}\ell(\zeta)(\omega+\pi/2)} U_\zeta(\delta) \cdot \Sigma_\zeta \cdot \mathrm{e}^{\mathrm{i}\ell(\zeta)\psi} V_\zeta(k). \end{aligned} \tag{63}$$

The factors in the last equation correspond respectively to the left singular functions, singular values, and right singular functions of $F(\boldsymbol{\delta}, \boldsymbol{k})$ that one would obtain by minimizing the rank-H approximation

$$E_{\mathrm{HS}}^2 := \iint_{\Omega_D} \iint_{\Omega_K} \left| F(\boldsymbol{\delta}, \boldsymbol{k}) - \sum_{\zeta=1}^{H} U_\zeta(\boldsymbol{\delta})\Sigma_\zeta V_\zeta(\boldsymbol{k}) \right|^2 d\boldsymbol{k} d\boldsymbol{\delta}\,. \tag{64}$$

The relationship of these vectors to those from the separate SVDs is

$$U_\zeta(\boldsymbol{\delta}) = U_\zeta(\delta, \omega) := \mathrm{e}^{-\mathrm{i}\ell(\zeta)(\omega+\pi/2)} U_\zeta(\delta), \qquad V_\zeta(\boldsymbol{k}) = V_\zeta(k, \psi) := \mathrm{e}^{\mathrm{i}\ell(\zeta)\psi} V_\zeta(k). \tag{65}$$

So the SVD approximation described in this section is nothing more than an approximation of $F(\boldsymbol{\delta}, \boldsymbol{k})$ built from these U_ζ, Σ_ζ, and V_ζ. As expected from the structure of the SVD, if $\zeta \neq \zeta'$, then $U_\zeta(\boldsymbol{\delta})$ and $U_{\zeta'}(\boldsymbol{\delta})$ are orthogonal over Ω_D, as are $V_\zeta(\boldsymbol{k})$ and $V_{\zeta'}(\boldsymbol{k})$ over Ω_K. Furthermore, by this orthogonality,

$$E_{\mathrm{HS}}^2 = \sum_{\ell \in \mathbb{Z}} E_{\mathrm{HS}}(\ell)^2 = \sum_{\zeta=H+1}^{\infty} \Sigma_\zeta^2. \tag{66}$$

An illustration of this structure is shown in Figure 2, where the array of values $\Sigma_\eta(\ell)$ is displayed for three choices of W. Using the $W = 1$ case as an example, consider what happens if we set our spectral-norm tolerance to $\epsilon = 10^{-2}$ (we will make this notion more precise in the next section). When $\ell = 0$, $\Sigma_\eta(0) \geq \epsilon$ only when $\eta \leq 4$; i.e., $H_0 = 4$ terms are required to achieve an ϵ-accurate approximation to $J_0(\delta k)$. On the other hand, when $\ell = 3$, $\Sigma_\eta(3) \leq \epsilon$ only when $\eta \leq 2$; i.e., only $H_3 = 2$ terms are required for an ϵ-accurate approximation to $J_3(\delta k)$. In order to combine the SVD expansions of the various $J_\ell(\delta k)$ into an ϵ-accurate expansion of $F(\boldsymbol{\delta}, \boldsymbol{k})$, we need to retain all the terms $\Sigma_\eta(\ell) \geq \epsilon$. With this choice of ϵ, we retain the $H = 34$ largest values of Σ_ζ, which correspond to the $H = \sum_\ell H_\ell = 34$ largest values of $\Sigma_\eta(\ell)$. In the next section will use this structure to discuss the approximation error of this expansion.

We illustrate this SVD approximation in Figure 3 which shows the structure of $U_\eta(\delta; 0)$ and $V_\eta(k; 0)$ for the lowest-order Bessel function $J_0(\delta k)$. Figure 4 illustrates the errors incurred by the SVD expansion when approximating the translation kernel $F(\boldsymbol{\delta}, \boldsymbol{k})$. Note that the SVD approximation provides a successively more accurate approximation of F as the number of terms H increases. In addition, as W increases, a larger H is required to achieve the same level of accuracy (compare Figure 4A with Figure 4C). The next section places rigorous bounds on this growth.

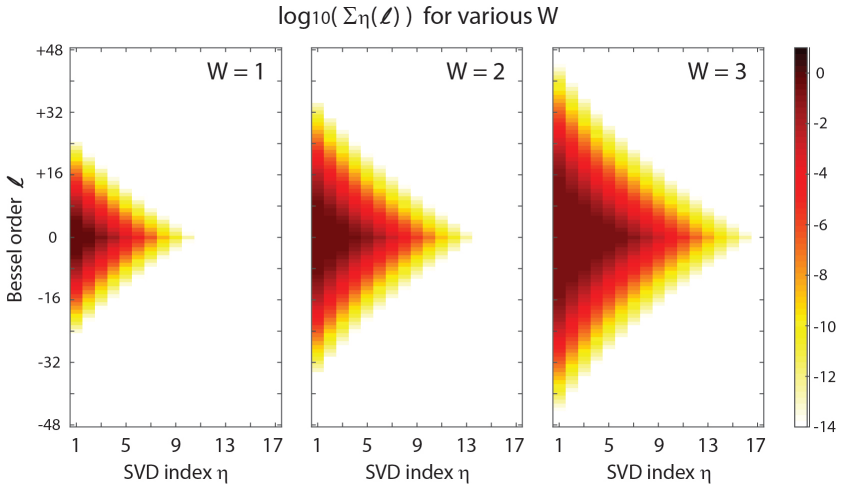

Figure 2. Operator singular values $\Sigma_\eta(\ell)$ of the Bessel kernel $J_\ell(k\delta)$ over $\delta \in [0, D]$ and $k \in [0, K]$, as a function of ℓ and singular value index (see Section 4.5). The cases $W = 1, 2$ and 3 are shown; W is the maximum translation magnitude in wavelengths ($2\Delta x$). In each case the values of $\log_{10}(\Sigma_\eta(\ell))$ are displayed using the colorbar to the far right. The number of SVD terms H equals the number of index pairs (η, ℓ) with $\Sigma_\eta(\ell) \geq \epsilon$; these pairs form an approximately triangular region.

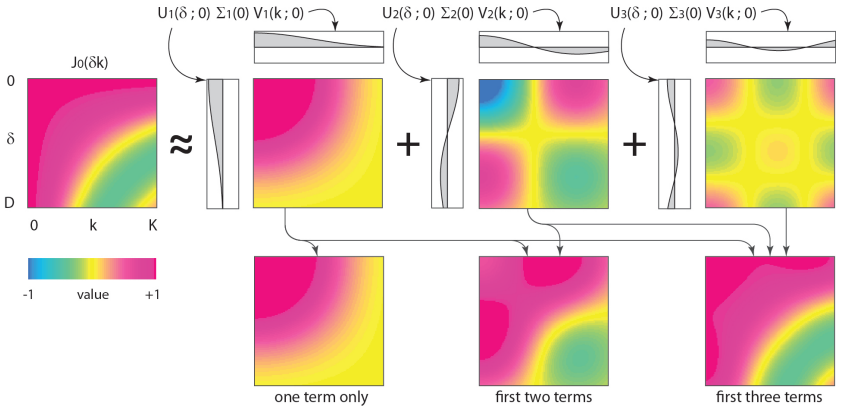

Figure 3. The image in the top left corner shows the Bessel function J_0 as a function of δ (vertical axis) and k (horizontal axis), i.e. the case $\ell = 0$. The domains D and K correspond to $W = 1$. On the left-hand side we show the first three terms in the corresponding SVD. Each image shows the product $U_\eta(\delta; 0)\Sigma_\eta(0)V_\eta(k; 0)$ for $\eta = 1, 2, 3$. To the left of each image is shown $U_\eta(\delta; 0)\sqrt{\Sigma_\eta(0)}$, while on top we show $\sqrt{\Sigma_\eta(0)}V_\eta(\delta; 0)$. Below each SVD outer product term, we plot their cumulative sum. Here $H_0 = 3$ terms is sufficient to capture the coarse features of $J_0(\delta k)$.

5. Error analysis

In this section we prove a rigorous upper bound on the number of singular values needed in FTK to achieve a relative error ϵ in inner product computations, given a maximum dimensionless translation W. We first prove the bound on the ϵ-rank of the translation kernel, then show how this controls the relative error over the set of inner products.

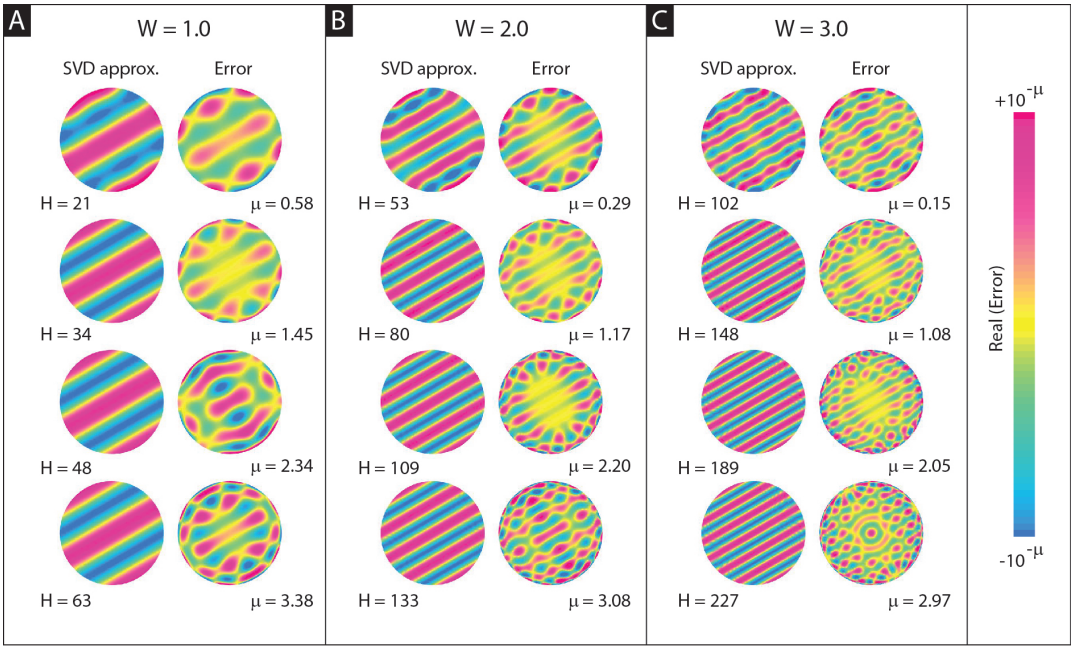

Figure 4. Here we illustrate the SVD approximation to the translation kernel $F(\boldsymbol{\delta}, \boldsymbol{k}) = \mathrm{e}^{-\mathrm{i}\boldsymbol{\delta}\cdot\boldsymbol{k}}$, with $\boldsymbol{\delta}$ chosen so that $\omega = \pi/4$ and $\delta = D = 2\pi W/K$. In panel A we show a pair of columns associated with $W = 1$. Each row in this panel shows the real component of the SVD approximation on the left and the real component of the associated error on the right. The number of terms H increases with each row. Both the left and right columns use the same color scale (far right), but with different limits: the SVD approximation to F has limits of $[-1, 1]$, whereas the error has limits of $[-10^{-\mu}, 10^{-\mu}]$, with the value of μ listed below each row. The value of μ is the number of digits of (pointwise) accuracy achieved by the corresponding SVD approximation. Panel B and panel C display similar results for $W = 2$ and 3.

5.1. A bound on the ϵ-rank of $F(\boldsymbol{\delta}, \boldsymbol{k})$

Here we prove that the number of retained SVD terms need grow at worst quadratically in W and linearly in the desired number of digits of accuracy $\log(1/\epsilon)$.

Theorem 1 *Let $W = DK/2\pi$ be the maximum translation in wavelengths defined in Section 2, and let $\epsilon > 0$ be the desired precision. Let H be the number of singular values Σ_ζ of the translation kernel $F(\boldsymbol{\delta}, \boldsymbol{k})$ larger than ϵ, i.e. H is the ϵ-rank of F. Then*

$$H = \mathcal{O}((W + \log(1/\epsilon))^2) \ . \tag{67}$$

Proof. As discussed in the previous section, if H_ℓ is the number of singular values $\Sigma_\eta(\ell)$ greater than ϵ, for the translation kernel with kernel $J_\ell(\delta k)$, then H in the theorem statement equals $\sum_\ell H_\ell$. The lemma below implies that this is achieved with $0 \le H_\ell \le \max(0, \mathcal{H}_{\epsilon,W} - |\ell|/2)$, with $\mathcal{H}_{\epsilon,W} := \max(\pi \mathrm{e}^2 W, \log(2\pi W/\epsilon)) + 3/2$. This means that $H_\ell = 0$ for $|\ell| > 2\mathcal{H}_{\epsilon,W}$. Then $H = \sum_{\ell\in\mathbb{Z}} H_\ell \le \sum_{|\ell|\le 2\mathcal{H}_{\epsilon,W}} (\mathcal{H}_{\epsilon,W} - |\ell|/2) = 2\mathcal{H}^2_{\epsilon,W}$. Asymptotically, we may summarize $\mathcal{H}_{\epsilon,W} = \mathcal{O}(W + \log(1/\epsilon))$, completing the proof. □

The key lemma below shows that the number of terms H_ℓ grows linearly with W and $\log(1/\epsilon)$, but shrinks linearly with $|\ell|$. This behavior is clear in Figure 2 as the roughly triangular shape of the set of indices with significant singular values.

Lemma 1 *Let $\ell \in \mathbb{Z}$, and let $H_\ell \geq 0$ be the ϵ-rank of the modal translation kernel $J_\ell(\delta k)$ acting from $L^2([0,K])$ to $L^2([0,D])$ with linear weight functions as defined in the previous section. Then $H_\ell \leq \overline{H}_\ell$, where*

$$\overline{H}_\ell \;:=\; \max(0,\, \pi \mathrm{e}^2 W - |\ell|/2,\, \log(2\pi W/\epsilon) + 3/2 - |\ell|/2)\;. \tag{68}$$

Proof. Defining rescaled variables $x := K\delta$ and $y := k/K$, the kernel becomes $J_\ell(\delta k) = J_\ell(xy)$ from functions over $y \in [0,1]$ to those over $x \in [0, 2\pi W]$; one may check that rescaling does not change the singular values $\Sigma_\eta(\ell)$. Consider the case ℓ even. We exploit an identity given by Wimp [41, Eq. (2.22)],

$$J_\ell(xy) = \sum_{\nu=0}^{\infty} C_{2\nu}(x) T_{2\nu}(y), \quad |y| \leq 1, \qquad C_{2\nu}(x) = \varepsilon_\nu J_{\ell/2+\nu}(x/2) J_{\ell/2-\nu}(x/2), \tag{69}$$

where $T_{2\nu}$ is the 2νth Chebyshev polynomial, and $\varepsilon_\nu = 1$ for $\nu = 0$ and 2 otherwise. Applying $|J_{|\ell|/2-\nu}(x/2)| \leq 1$ [25, Eq. (10.14.1)] to the smaller magnitude of the two Bessel orders, we get a bound in terms of the Bessel with larger magnitude order

$$|C_{2\nu}(x)| \leq 2 |J_{|\ell|/2+\nu}(x/2)|\;. \tag{70}$$

Recalling the definition in (68), we now claim that

$$\sum_{\nu \geq \overline{H}_\ell} |C_{2\nu}(x)| \leq \frac{\epsilon}{2\pi W}\;, \qquad \text{for all } x \in [0, 2\pi W]\;. \tag{71}$$

To prove this, for each ν we rescale the Bessel argument in (70) by writing $p := x/(|\ell|+2\nu)$ as the ratio of argument $x/2$ to order $|\ell|/2+\nu$, or fraction of the distance from the origin to the Bessel turning point. We then apply Siegel's bound [25, Eq. (10.14.5)] in the evanescent region $0 \leq p \leq 1$,

$$|J_{|\ell|/2+\nu}((|\ell|/2+\nu)p)| \;\leq\; \mathrm{e}^{(|\ell|/2+\nu)\left(\log p + \sqrt{1-p^2} - \log\left(1+\sqrt{1-p^2}\right)\right)}\;, \tag{72}$$

From (68), we obtain that $\nu \geq \overline{H}_\ell$ implies $2\pi W \leq \mathrm{e}^{-2}(|\ell|+2\nu)$. Since $x \leq 2\pi W$, we then have $p \leq \mathrm{e}^{-2}$ and $0.99 < \sqrt{1-p^2} < 1$. The second factor in the exponent of (72) is therefore less than $-2+1-\log(1.99) < -1$. Consequently, (72) is bounded by simple exponential decay:

$$|J_{|\ell|/2+\nu}(x/2)| \;\leq\; \mathrm{e}^{-(|\ell|/2+\nu)}\;, \quad \text{for all } x \in [0, 2\pi W], \quad \nu \geq \overline{H}_\ell. \tag{73}$$

This bounds the left side of (71) by the geometric series,

$$\sum_{\nu \geq \overline{H}_\ell} |C_{2\nu}(x)| \;\leq\; 2 \sum_{\nu \geq \overline{H}_\ell} \mathrm{e}^{-(|\ell|/2+\nu)} \;\leq\; \frac{2}{1-\mathrm{e}^{-1}} \mathrm{e}^{-(|\ell|/2+\overline{H}_\ell)}\;.$$

Substituting the bound $\overline{H}_\ell \geq \log(2\pi W/\epsilon) + 3/2 - |\ell|/2$ from (68), and using $\mathrm{e}^{3/2} > 2/(1-\mathrm{e}^{-1})$, now establishes the claim (71).

We work in weighted L^2-norms, $\|f\|^2 := \int_0^a |f(x)|^2\, x dx$, where $a = 1$ for the domain or $a = 2\pi W$ for the range. By Allahverdiev's theorem [9, p. 28], the error in the induced operator norm $\|\cdot\|$ of any rank-r approximation is at least the $(r+1)$th operator singular value. Choosing $r = \overline{H}_\ell$ and recognizing the $\nu < r$ truncation of (69) as a particular rank-r approximation to J_ℓ, we insert the definition of the operator norm and get

$$\begin{aligned}
\Sigma_{\overline{H}_\ell+1}(\ell)^2 \le \Big\| J_\ell(\cdot,\cdot) - \sum_{\nu=0}^{\overline{H}_\ell-1} C_{2\nu}(\cdot)T_{2\nu}(\cdot)\Big\|^2 &= \Big\| \sum_{\nu\ge\overline{H}_\ell} C_{2\nu}(\cdot)T_{2\nu}(\cdot)\Big\|^2 \\
&= \sup_{\|f\|=1} \int_0^{2\pi W} \Big| \int_0^1 \sum_{\nu\ge\overline{H}_\ell} C_{2\nu}(x)T_{2\nu}(y) f(y)\, y dy\Big|^2 x dx \\
&\le \sup_{\|f\|=1} \int_0^{2\pi W} \Big(\int_0^1 \Big| \sum_{\nu\ge\overline{H}_\ell} C_{2\nu}(x)T_{2\nu}(y)\Big|^2 y dy\Big) \cdot \Big(\int_0^1 f(y)^2 y dy\Big)\, x dx \\
&\le \int_0^{2\pi W} \Big(\sum_{\nu\ge\overline{H}_\ell} |C_{2\nu}(x)|\Big)^2 \cdot 1\; x dx \;\le\; (2\pi W)^2 \Big(\frac{\epsilon}{2\pi W}\Big)^2 \;\le\; \epsilon^2\,, \qquad (74)
\end{aligned}$$

where to reach the third line we used the y-weighted Cauchy–Schwarz inequality, to reach the fourth line we used the fact that the magnitude of the Chebyshev polynomials are bounded by one, and finally the claim (71). Note that, in second expression above, when $H_\ell = 0$ there are no terms in the sum. The bound just shown, $\Sigma_{\overline{H}_\ell+1}(\ell) \le \epsilon$, is equivalent to the statement that the ϵ-rank is no more than $\overline{H}_\ell$, completing the proof.

The proof for ℓ odd is similar, exploiting the identity [41, Eq. (2.23)]. □

Remark 7 *The above proof gives explicit constants in the upper bound on H_ℓ, but since a small argument-to-order ratio $p < \mathrm{e}^{-2}$ was chosen, these constants are far from being tight and could be improved. The empirical behavior of H_ℓ has the same linear form as (68) but with different constants: a numerical fit gives $H_\ell \approx 2.0(W - |\ell|/2\pi) + 0.5\log(1/\epsilon)$. This is strong evidence that the quadratic power in Theorem 1 is tight.*

5.2. Induced error on inner product $X(\boldsymbol{\delta}, \gamma)$

Consider the induced, or spectral, norm E_2 of the difference between the translation kernel and its rank-H expansion, defined by

$$E_2^2 := \sup_{\|\hat{A}\|_2=1} \iint_{\Omega_D} \Big| \iint_{\Omega_K} \Big(F(\boldsymbol{\delta},\boldsymbol{k}) - F^{\mathrm{SVD}}(\boldsymbol{\delta},\boldsymbol{k})\Big) \hat{A}(\boldsymbol{k})\, d\boldsymbol{k}\Big|^2 d\boldsymbol{\delta}\,, \qquad (75)$$

where $\hat{A} \in L^2(\Omega_K)$, and $\|\hat{A}\|_2$ is the 2-norm on $L^2(\Omega_K)$. By Allahverdiev's theorem [9, p. 28] (see the proof of Lemma 1),

$$E_2 = \Sigma_{H+1}\,. \qquad (76)$$

We now show that FTK induces a small error on the inner product $X(\boldsymbol{\delta}, \gamma)$. To simplify the analysis, we consider the FTK approximation of its continuous analog $\mathcal{X}(\boldsymbol{\delta}, \gamma)$,

$$\mathcal{X}^{\mathrm{SVD}}(\boldsymbol{\delta}, \gamma) = \iint_{\Omega_K} F^{\mathrm{SVD}}(\boldsymbol{\delta},\boldsymbol{k}) \hat{A}(\boldsymbol{k})^* R_{-\gamma} \hat{B}(\boldsymbol{k})\, d\boldsymbol{k}\,. \qquad (77)$$

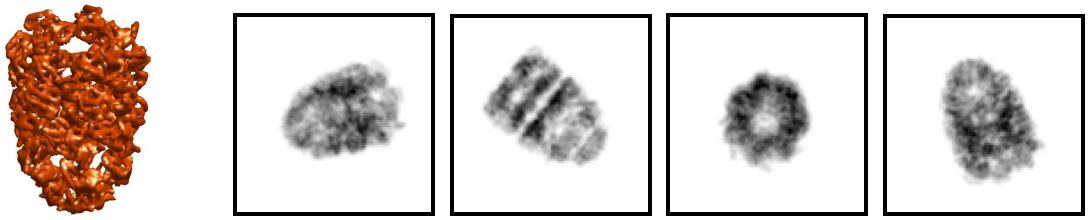

Figure 5. Four projection templates of the 3D electron density shown on the far left, taken at random Euler angles, and sampled on a 128×128 pixel grid. The images are then obtained by randomly rotating and translating these 2D templates with sub-pixel resolution. The task is to recover the true translations and rotations as well as the templates that match each of the images.

The squared L^2-norm of the residual $\mathcal{X}(\boldsymbol{\delta},\gamma) - \mathcal{X}^{\mathrm{SVD}}(\boldsymbol{\delta},\gamma)$ as a function over $\Omega_D \times [0, 2\pi)$ is now bounded as follows:

$$\begin{aligned}
\left\|\mathcal{X} - \mathcal{X}^{\mathrm{SVD}}\right\|_2^2 &= \int_0^{2\pi} \iint_{\Omega_D} \left|\mathcal{X}(\boldsymbol{\delta},\gamma) - \mathcal{X}^{\mathrm{SVD}}(\boldsymbol{\delta},\gamma)\right|^2 d\boldsymbol{\delta}\, d\gamma \\
&= \int_0^{2\pi} \iint_{\Omega_D} \left| \iint_{\Omega_K} \Big(F(\boldsymbol{\delta},\boldsymbol{k}) - F^{\mathrm{SVD}}(\boldsymbol{\delta},\boldsymbol{k})\Big) \hat{A}(\boldsymbol{k})^* R_{-\gamma} \hat{B}(\boldsymbol{k})\, d\boldsymbol{k} \right|^2 d\boldsymbol{\delta}\, d\gamma \\
&\leq E_2^2 \int_0^{2\pi} \iint_{\Omega_K} \left|\hat{A}(\boldsymbol{k})^* R_{-\gamma} \hat{B}(\boldsymbol{k})\right|^2 d\boldsymbol{k}\, d\gamma \\
&= \Sigma_{H+1}^2 \int_0^{2\pi} \|\hat{A}(\cdot)^* R_{-\gamma} \hat{B}(\cdot)\|_2^2\, d\gamma\,. \qquad (78)
\end{aligned}$$

Thus, the relative root mean square error in the inner product function is controlled by Σ_{H+1}, which, if H is chosen as in Theorem 1, cannot exceed the desired ϵ. The last factor in (78) depends only on the overlap of the radial power spectra of A and B and may be rewritten as $2\pi \int_0^K |a(k,0)b(k,0)|^2\, dk$. Note that the bound (78) is tight, and is achieved if $\hat{A}(\boldsymbol{k})^* R_{-\gamma} \hat{B}(\boldsymbol{k})$ equals the first omitted right singular function $V_{H+1}(\boldsymbol{k})$.

The above argument provides a bound on the error induced by using the truncated translation kernel $F^{\mathrm{SVD}}(\boldsymbol{\delta},\boldsymbol{k})$ in $\mathcal{X}(\boldsymbol{\delta},\gamma)$. The discretized inner products $X(\boldsymbol{\delta},\gamma)$ and $X^{\mathrm{SVD}}(\boldsymbol{\delta},\gamma)$ differ from $\mathcal{X}(\boldsymbol{\delta},\gamma)$ and $\mathcal{X}^{\mathrm{SVD}}(\boldsymbol{\delta},\gamma)$ only by quadrature and Fourier truncation errors, which can be made arbitrarily small (see Section 3.1). Consequently, the error $\|X(\boldsymbol{\delta},\gamma) - X^{\mathrm{SVD}}(\boldsymbol{\delta},\gamma)\|_2$ is dominated by Σ_{H+1}, and hence ϵ.

6. Numerical results

In this section we compare the proposed FTK method of Section 4.5 with two existing fast methods for evaluating the inner product function $X(\boldsymbol{\delta},\gamma)$: the BFR and BFT methods (see Section 1). The methods are compared for the task of rigidly aligning $N_I \gg 1$ images to another set of N_I images, referred to as "templates" for clarity. We first define the BFR and BFT methods and describe their implementations.

6.1. Brute force rotations (BFR)

In short, for each of the N_I^2 image–template pairs, and each of the $\mathcal{O}(n)$ rotations of one of the images, we perform a standard fast 2D convolution of A with $R_{-\gamma}B$ using 2D FFTs. This relies on writing (17) as follows, using the 2D convolution theorem [4]:

$$\mathcal{X}(\boldsymbol{\delta},\gamma) = \iint_{\mathbb{R}^2} A(\boldsymbol{x}-\boldsymbol{\delta})R_{-\gamma}B(\boldsymbol{x})^* d\boldsymbol{x} = \frac{1}{(2\pi)^2}\iint_{\Omega_K} \hat{A}(\boldsymbol{k})\, R_{-\gamma}\hat{B}(\boldsymbol{k})^*\, \mathrm{e}^{-\mathrm{i}\boldsymbol{k}\cdot\boldsymbol{\delta}} d\boldsymbol{k}\ . \quad (79)$$

Discretizing the right-hand side on a Cartesian grid, we may now recover $\mathcal{X}(\boldsymbol{\delta},\gamma)$ on an entire $n\times n$ Cartesian $\boldsymbol{\delta}$ grid with a single 2D FFT. Since a 2D FFT is needed for each of $\mathcal{O}(n)$ rotations and N_I^2 image–template pairs, the cost per pair is $\mathcal{O}(n^3\log n)$, which is one power of n slower than the proposed SVD method; see Table 1. A finer sampling in $\boldsymbol{\delta}$ is achieved by zero-padding each FFT. Note that the cost is independent of W or N.

For each of the N_I images A and for each of the $\mathcal{O}(N_I n)$ rotated templates $R_{-\gamma}B$, a precomputation is needed to evaluate $\hat{A}(\boldsymbol{k})$ and $R_{-\gamma}\hat{B}(\boldsymbol{k})$ for each γ on an $n\times n$ Cartesian grid in $\boldsymbol{k}$. A common approach for computing $R_{-\gamma}\hat{B}(\boldsymbol{k})$ on such a grid is to resample by interpolation in real or Fourier space. We instead use the 2D type 2 NUFFT [2] as in Section 3.2 to get Fourier transform samples on polar grids with spectral accuracy; this preparation step is not included in our timings. Then for each γ we rotate on the polar grid and use a 2D type 1 NUFFT to resample to a Cartesian Fourier grid. We do include the latter in timings, but since this precomputation scales only as N_I, it is practically negligible compared to the main computation.

6.2. Brute force translations (BFT)

For each of the N_I templates B we precompute their Fourier–Bessel coefficients $b(k;q)$ on the rings $\{k_m\}_{m=1}^M$, $|q|\le Q$ (see Section 3.2). For each of the N_I images A and for each of their N desired translations, we use the same method for the translated Fourier–Bessel coefficients $T_{\boldsymbol{\delta}}a(k;q)$, except we multiply $\hat{A}$ by the translation phase (16) before the 1D FFTs in (34). This precomputation requires $\mathcal{O}(N_I N n^2\log n)$ operations.

For each of the $N_I^2 N$ pairs of templates and $\boldsymbol{\delta}$-translated images, we then apply (36) with $L=0$ to calculate $\hat{X}(\boldsymbol{\delta},q)$ and use (37) as in Section 4.1. This returns all inner products associated with $\boldsymbol{\delta}$ for the grid of $\gamma\in[0,2\pi)$. The total cost of precomputation is $\mathcal{O}(N_I^2 N(n^2+n\log n))$ effort, i.e. $\mathcal{O}(Nn^2)$ per image–template pair. We remark that a similar method was used for efficiently computing inner products across multiple rotations in [1]. Note that Joyeux *et al.* [17] instead use angular convolutions in real space, but that the expected cost is similar.

6.3. Performance comparison for cryo-EM template matching

To conduct our numerical experiments, we start with the GroEL–GroES 3D structure [42]. We compute $N_I = 10$ random tomographic projections (see Figure 5) to use as templates. The size of each template is 128×128 pixels (i.e., $n=128$), and the templates have unit L^2-norm in $[-1,1]^2$. The maximum frequency K is chosen as the Nyquist

frequency (see Remark 1). To generate images, we rotate the templates by random real angles in $[0, 2\pi)$ and translate them by random shifts within Ω_D, where D is varied in a manner described below. All such projections and transformations are computed using Fourier methods which perform spectrally accurate off-grid translations; see [1].

The timing results below are carried out with an efficient vectorized MATLAB implementation. Specifically, for BFT and FTK, computations are dominated by inner products, which are blocked together as matrix-matrix multiplications. Since these exploit level 3 BLAS, they are essentially as efficient as possible on the architecture. For BFR, the cost is dominated by FFT calls, which use MATLAB's version of the FFTW library, and are thus also very efficient. We run MATLAB 2016b on a Linux workstation with two six-core Intel Xeon E5-2643 CPUs at 3.4 GHz and 128 GB of memory.

We perform experiments over a wide range of D. For a given D, which in turn fixes $W = nD/4$, we cover Ω_D with an approximately uniform grid of translations $\boldsymbol{\delta}$ of two different realistic mean densities:

(a) a coarse grid of 4 translations per square pixel, i.e. mean spacing $\Delta x/2 \approx 0.0078$, yielding $N \approx \pi n^2 D^2$ translations, and

(b) a finer grid of 16 translations per square pixel, i.e. mean spacing $\Delta x/4 \approx 0.0039$, yielding $N \approx 4\pi n^2 D^2$ translations.

The former is commonly used in applications, while in our setting the latter is sufficient for robust alignment to the desired accuracy. We also search over a rotation grid of $2\pi K = n\pi^2 \approx 1300$ equispaced γ values; this grid does not depend on D. We now loop through image–template pairs, computing the array of inner products $X(\boldsymbol{\delta}, \gamma)$ on the translation and rotation grids using the three methods described above: BFR, BFT, and the proposed FTK method. For each experiment, we record the full time taken to compute these inner product arrays. The full computation time includes the precomputation scaling as N_I, which involves preparing the images and templates for inner product calculations, plus the actual computation of the inner products scaling as N_I^2. The latter dominates the cost as N_I grows. We exclude from our timing results the one-time FTK planning stage that is independent of the images, such as filling the translation kernel matrices and computing their SVD; this is negligible in any realistic application. In FTK, we set the (relative) accuracy to $\epsilon = 10^{-2}$, which, together with W, determines the number of SVD terms H.

As a measure of consistency, we ensure that—after the calculation—the maximal inner product for each image corresponds to that of the true template and the nearest possible on-grid shift and angle used to generate that image.

Figure 6 compares the running time of the BFR and BFT methods with our proposed FTK method across a range of N corresponding to D between 0.04 and 0.4. The largest D is equivalent to a 25-pixel shift. The comparison shows that, for a wide range of N, FTK outperforms BFT by a factor of 3 for the $\Delta x/2$ spacing. For the $\Delta x/4$ spacing grid, we observe a speedup factor of 8–10. At large N, the precomputation stage of BFT takes about one third of the overall running time, while for FTK, roughly one tenth of

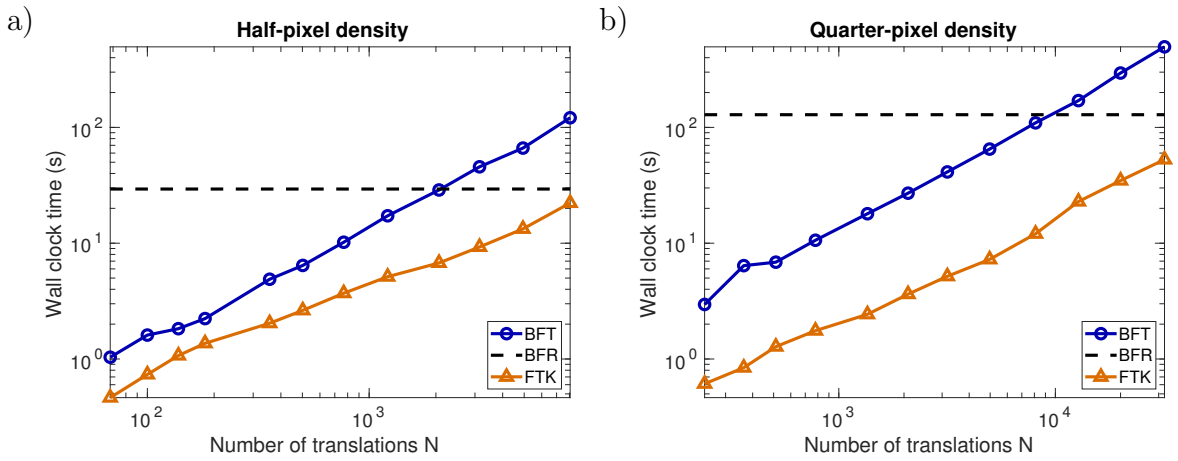

Figure 6. Timing results for the calculation of all inner products between rigid transformations of $N_I = 10$ images and N_I templates, of size $n \times n$ pixels for $n = 128$. Three methods are compared: BFR (brute force rotations, as implemented in Section 6.1), BFT (brute force translations, as in Section 6.2), and our proposed FTK method (Section 4.5). The horizontal axis shows N; this scales as $\mathcal{O}(D^2)$ because we choose a constant density of translations of in a disk of radius D. The mean translation spacings are (a) half-pixel $\Delta x/2$ and (b) quarter-pixel $\Delta x/4$. In both plots, the range of D is $[0.04, 0.4]$, i.e., between 1/50 and 1/5 of the image size.

the time is spent in precomputation. The figure also shows that BFR takes about 30 s for $\Delta x/2$ and 100 s for $\Delta x/4$, independent of N. In both cases, for all $D \leq 0.4$ (shifts up to 25 pixels), FTK is faster than BFR.

Note that, in order to maintain the same user-defined error $\epsilon = 10^{-2}$, the number of terms H increases with W. This relationship is exhibited more clearly in Figure 7, which shows the dependence between ϵ and H for various W. This figure also shows the corresponding relationship for linear interpolation (Section 4.2). For $\epsilon = 10^{-2}$, linear interpolation requires ~10 times as many nodes as FTK; this ratio is quite similar to the speedup shown in Figure 6. Furthermore, we note that H for the FTK method roughly follows the $\mathcal{O}((W + \log(1/\epsilon))^2)$ scaling given in Theorem 1.

Finally, we emphasize that, as Table 1 makes clear, in order for FTK to be more efficient than BFT, the number of required terms H must be smaller than the requested number of translations N, which requires a sufficient density of translations. What is not explicitly stated in the table is that a coarsely sampled grid of translations may not be sufficient to reconstruct the landscape of inner products. Indeed, while BFT produces accurate inner products for the translations on the grid, we must interpolate for off-grid translations. As shown in Figure 8, a popular scheme such as linear interpolation does not accurately reproduce these intermediate inner products. By contrast, FTK maintains nearly uniform error across the space of translations $\delta \leq D$ for the accuracy determined by H, as explained in Section 5.

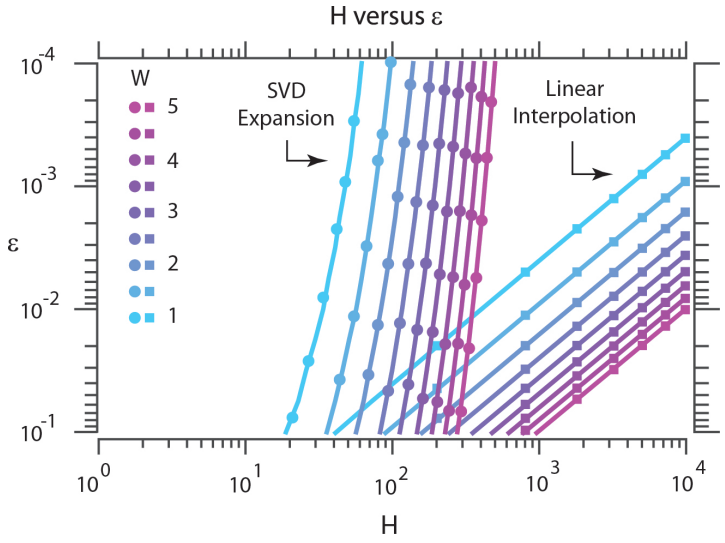

Figure 7. The accuracy of our SVD expansion (solid dots) as a function of the number of SVD terms H, on logarithmic scales. We measure accuracy in terms of $\varepsilon = E_{\text{HS}}$ of (64). By (66) and Section 5.2, this provides a bound on ϵ. The value $\log_{10}(1/\varepsilon)$ can be interpreted as the number of digits of accuracy of our expansion. Results for different values of W are shown in different colors (see legend at the left). Results for linear interpolation are also shown (solid squares).

7. Extensions

7.1. Other 2D kernels

Taking a step back, we can describe our approach in broad terms. Our original goal was to calculate inner products across a variety of translations. The first step was to notice that the translation operator $T_{\boldsymbol{\delta}}$ corresponds to pointwise multiplication by $F(\boldsymbol{\delta}, \boldsymbol{k})$ over the Fourier domain $\boldsymbol{k}$. Our method boils down to approximating $F(\boldsymbol{\delta}, \boldsymbol{k})$ with a sum of separable terms, each of the form $Y_\zeta(\boldsymbol{\delta}) \cdot G_\zeta(\boldsymbol{k})$.

The same approach can also be applied to other convolutional operators. For example, we might consider the inner product between one image and another that has been translated by some $\boldsymbol{\delta}$, rotated by some γ, and convolved with a filter which depends on some parameter vector $\boldsymbol{\tau}$. This is the case in cryo-EM, where images are filtered by some contrast transfer function (CTF), which is parametrized by a set of defocus parameters [22]. In addition to aligning the image with a CTF-filtered template, we may want to find the defocus parameters that give the highest correlation, which can yield improved reconstruction accuracy [3].

In the Fourier domain, translating by $\boldsymbol{\delta}$ and filtering by a CTF with defocus parameters $\boldsymbol{\tau}$ corresponds to pointwise multiplication by a function dependent on $\boldsymbol{\delta}$ and $\boldsymbol{\tau}$. We can extend our approach to tackle this scenario by factorizing this function into terms of the form $Y_\zeta(\boldsymbol{\delta}, \boldsymbol{\tau}) \cdot G_\zeta(\boldsymbol{k})$. Plugging this factorization into FTK then yields a fast algorithm for computing inner products as a function of $\boldsymbol{\delta}$, γ, and $\boldsymbol{\tau}$.

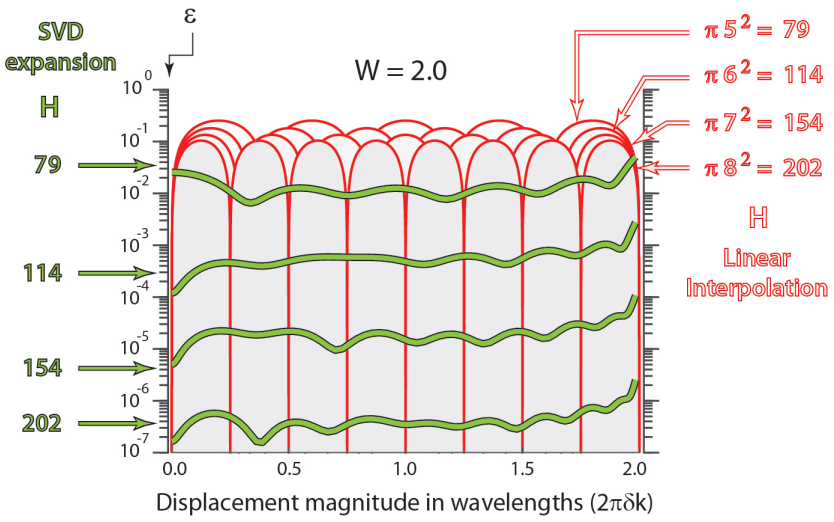

Figure 8. The accuracy of our SVD expansion (solid green lines) as a function of the shift in wavelengths $2\pi\delta K$ for $W = 2$ using a semi-logarithmic scale. We measure accuracy in terms of $\varepsilon(\delta) = \|F(\boldsymbol{\delta}, \cdot) - F^{\mathrm{SVD}}(\boldsymbol{\delta}, \cdot)\|_2$ (i.e., the 2-norm on Ω_K), where the displacement direction ω is fixed at zero. The value $\log_{10}(1/\varepsilon)$ can be interpreted as the number of digits of accuracy of our expansion. Each solid green line corresponds to one particular choice of H for our SVD expansion (values listed to the left). Results for linear interpolation are also shown (red lines, number of nodes listed to the right).

7.2. 3D volume-to-volume registration

We have so far discussed the calculation of inner products between 2D images. A closely related problem is the calculation of inner products between 3D volumes for the purposes of volume-to-volume alignment. In this context, rotations can be described by their Euler angles α, β, and γ, and the translations $\boldsymbol{\delta}$ are in $\mathbb{R}^3$. At first glance, it appears as though our methodology should generalize naturally from 2D to 3D. Unfortunately, there are several important differences which make such a generalization difficult.

To understand this impasse, first recall that our overall strategy was to (a) choose a basis in which rotations can be applied cheaply, (b) factorize the action of the translation operator in that basis using the SVD, and finally (c) use this factorization to calculate the inner products $\mathcal{X}(\boldsymbol{\delta}, \gamma)$ over a set of translations $\boldsymbol{\delta}$ and rotations γ.

The first problem arises in step (a). In 2D, rotations commute, and as a result, there exists a basis in which rotations act diagonally: the Fourier–Bessel basis. However, rotations do not commute in 3D, so there is no such basis for volumes. A compromise is the spherical Fourier–Bessel basis, where the angular component is given by spherical harmonic functions. In this basis, rotations are block diagonal, allowing us to compute multiple rotations quickly using 2D FFTs.

Step (b) proceeds in the same way for 3D, but step (c) incurs an additional cost. Indeed, once we have factorized the translation operator in the spherical Fourier–Bessel basis, we multiply each right singular function $V_\zeta(\boldsymbol{k})$ by the coefficients of the volumes in the basis. The resulting expression corresponds to one inner product in 2D, but not

in 3D. Thus, if we use the spherical Fourier–Bessel basis, applying each term in the SVD expansion requires significantly more work to calculate than a single inner product.

Another option is the 3D generalization of the least-squares interpolant described in Section 4.3. The normal equations are similar to (42), with the associated matrix

$$\mathcal{M}^{\mathrm{LS}}(\boldsymbol{\delta}', \boldsymbol{\delta}) = \frac{\sin(\tilde{\delta}K) - \tilde{\delta}K\cos(\tilde{\delta}K)}{(\tilde{\delta}K)^3} \ , \tag{80}$$

which has a derivative

$$\partial_{\tilde{\delta}K}\mathcal{M}^{\mathrm{LS}}(\boldsymbol{\delta}', \boldsymbol{\delta}) = \frac{((\tilde{\delta}K)^2 - 3)\sin(\tilde{\delta}K) + 3(\tilde{\delta}K)\cos(\tilde{\delta}K)}{(\tilde{\delta}K)^4} \ . \tag{81}$$

As before, we may optimize the node locations using gradient descent, but we encounter the same instability issues at high accuracies.

Alternatively, if one of the volumes is known beforehand (e.g., when aligning a given volume against a fixed set of volumes), then we can use a functional SVD to precompute a factorized representation of

$$X(\boldsymbol{\delta}, \alpha, \beta, \gamma) = \iiint_{\mathbb{R}^3} (T_{\boldsymbol{\delta}} R_{\alpha\beta\gamma} \hat{A}(\boldsymbol{k}))^* \hat{B}(\boldsymbol{k}) \, d\boldsymbol{k} \tag{82}$$

such that:

$$X(\boldsymbol{\delta}, \alpha, \beta, \gamma) \approx \sum_{\zeta=1}^{H} Y_\zeta(\boldsymbol{\delta}) \iiint_{\mathbb{R}^3} G_\zeta(\boldsymbol{k}; \alpha, \beta, \gamma) \hat{B}(\boldsymbol{k}) \, d\boldsymbol{k} \tag{83}$$

where $R_{\alpha\beta\gamma}$ represents 3D rotation by the Euler angles α, β, γ. Here, $Y_1, \ldots, Y_H$ and $G_1, \ldots, G_H$ depend on the specifics of volume A, but not on B.

8. Conclusion

We introduced a new efficient method—named FTK—for calculating multiple inner products between two images of $n \times n$ pixels, one fixed and another rotated and translated with respect to the first. The method is based on decomposing the images in a Fourier–Bessel basis, allowing us to compute the inner products quickly for multiple rotation angles via 1D FFTs. Furthermore, the truncated SVD of the translation operator enables us to compute approximate inner products efficiently for a large number N of arbitrary translations lying in a disk of radius $2W$ pixels. The method has computational complexity $\mathcal{O}(Hn(n+N))$, where we prove (Theorem 1) that the rank $H = \mathcal{O}((W + \log(1/\epsilon))^2)$, where ϵ is the relative root mean square error. The method is evaluated on a set of synthetic cryo-EM projections with sub-pixel translation grid spacings, where it is shown to significantly outperform the competing BFR and BFT methods. At mean spacing $\Delta x/4$ we show an acceleration over other known inner-product-forming methods of around $10\times$ over a wide range of W. The method is $\mathcal{O}(n \log n)$ times faster than the popular FFT-based cross-correlation alignment method (BFR), making it favorable for large images. Finally we present possible extensions of the method to other 2D kernels and 3D volume alignment, noting that the latter case poses additional difficulties due to the non-commutativity of rotations in 3D.

Acknowledgments

The authors thank Alex Townsend for showing us the expansion (69) crucial to Lemma 1, and Amit Singer, Leslie Greengard, and Zydrunas Gimbutas for fruitful discussions and helpful suggestions. The Flatiron Institute is a division of the Simons Foundation.

References

[1] A. Barnett, L. Greengard, A. Pataki, and M. Spivak. Rapid solution of the cryo-EM reconstruction problem by frequency marching. *SIAM J. Imaging Sci.*, 10(3):1170–1195, 2017.

[2] A. Barnett, J. Magland, and L. af Klinteberg. A parallel nonuniform fast Fourier transform library based on an "exponential of semicircle" kernel. *SIAM J. Sci. Comput.*, 41(5):C479–C504, 2019.

[3] A. Bartesaghi et al. Atomic resolution cryo-EM structure of beta-galactosidase. *Structure*, 26(6):848–856.e3, 2018.

[4] R. Bracewell. *The Fourier Transform and Its Applications*. McGraw-Hill, 3rd edition, 1999.

[5] A. Buades, B. Coll, and J.-M. Morel. A non-local algorithm for image denoising. In *Proc. CVPR*, volume 2, pages 60–65 vol. 2, June 2005.

[6] Y. Cheng, N. Grigorieff, P. A. Penczek, and T. Walz. A primer to single-particle cryo-electron microscopy. *Cell*, 161:439–449, 2015.

[7] A. Dutt and V. Rokhlin. Fast Fourier transforms for nonequispaced data. *SIAM J. Sci. Comput.*, 14:1369–1393, 1993.

[8] D. Elmlund and H. Elmlund. Cryogenic electron microscopy and single-particle analysis. *Annu. Rev. Biochem.*, 84:499–517, 2015.

[9] I. C. Gohberg and M. G. Krein. *Introduction to the theory of linear nonselfadjoint operators*. American Mathematical Society, Providence, 1969.

[10] G. H. Golub and C. F. Van Loan. *Matrix computations*. JHU Press, 4th edition, 2013.

[11] A. B. Goncharov. Methods of integral geometry and finding the relative orientation of identical particles arbitrarily arranged in a plane from their projections onto a straight line. *Dokl. Phys.*, 32:173, 1987.

[12] L. Greengard and J.-Y. Lee. Accelerating the nonuniform fast Fourier transform. *SIAM Review*, 46(3):443–454, 2004.

[13] N. Grigorieff. FREALIGN: High-resolution refinement of single particle structures. *J. Struct. Biol.*, 157(1):117–125, 2007.

[14] N. Grigorieff. Frealign: An exploratory tool for single-particle cryo-EM. In R. A. Crowther, editor, *The Resolution Revolution: Recent Advances In cryoEM*, volume 579 of *Methods Enzymol.*, pages 191–226. Academic Press, 2016.

[15] G. Harauz, E. Boekema, and M. van Heel. Statistical image analysis of electron micrographs of ribosomal subunits. *Methods Enzymol.*, 164:35–49, 1988.

[16] S. Jonic, C. Sorzano, P. Thevenaz, C. El-Bez, S. De Carlo, and M. Unser. Spline-based image-to-volume registration for three-dimensional electron microscopy. *Ultramicroscopy*, 103(4):303–317, 2005.

[17] L. Joyeux and P. A. Penczek. Efficiency of 2D alignment methods. *Ultramicroscopy*, 92(2):33–46, 2002.

[18] J. Keiner, S. Kunis, and D. Potts. Using NFFT 3 — a software library for various nonequispaced fast Fourier transforms. *ACM Trans. Math. Software*, 36(4), 2009.

[19] C. Kervrann and J. Boulanger. Optimal spatial adaptation for patch-based image denoising. *IEEE Trans. Image Process.*, 15(10):2866–2878, Oct 2006.

[20] R. R. Lederman, J. Andén, and A. Singer. Hyper-Molecules: on the Representation and Recovery of Dynamical Structures, with Application to Flexible Macro-Molecular Structures in Cryo-EM. *Inverse problems, accepted.* 2019.

[21] D. Lyumkis, A. F. Brilot, D. L. Theobald, and N. Grigorieff. Likelihood-based classification of cryo-EM images using FREALIGN. *J. Struct. Biol.*, 183(3):377–388, 2013.

[22] J. A. Mindell and N. Grigorieff. Accurate determination of local defocus and specimen tilt in electron microscopy. *J. Struct. Biol.*, 142(3):334–347, 2003.

[23] K. Murata and M. Wolf. Cryo-electron microscopy for structural analysis of dynamic biological macromolecules. *Biochim. Biophys. Acta Gen. Subj.*, 1862(2):324–334, 2017.

[24] E. Nogales and S. Scheres. Cryo-EM: a unique tool for the visualization of macromolecular complexity. *Mol. Cell.*, 58:677–689, 2015.

[25] F. W. J. Olver, D. W. Lozier, R. F. Boisvert, and C. W. Clark, editors. *NIST Handbook of Mathematical Functions*. Cambridge University Press, 2010. `http://dlmf.nist.gov`.

[26] W. Park, D. R. Madden, D. N. Rockmore, and G. S. Chirikjian. Deblurring of class-averaged images in single-particle electron microscopy. *Inverse Prob.*, 26(3):035002, 2010.

[27] P. Penczek, M. Radermacher, and J. Frank. Three-dimensional reconstruction of single particles embedded in ice. *Ultramicroscopy*, 40:33–53, 1992.

[28] A. Punjani, J. Rubinstein, D. Fleet, and M. Brubaker. cryoSPARC: algorithms for rapid unsupervised cryo-EM structure determination. *Nat. Methods*, 14:290–296, 2017.

[29] S. H. W. Scheres. A Bayesian view on cryo-EM structure determination. *J. Mol. Biol.*, 415:406–418, 2012.

[30] E. Schmidt. Zur Theorie der linearen und nichtlinearen Integralgleichungen. I Teil. Entwicklung willkürlichen Funktionen nach System vorgeschriebener. *Math. Ann.*, 63:433–476, 1907.

[31] Y. Shkolnisky and A. Singer. Viewing direction estimation in cryo-EM using synchronization. *SIAM J. Imaging Sci.*, 5(3):1088–1110, 2012.

[32] F. Sigworth. Principles of cryo-EM single-partice image processing. *Microscopy*, 65(1):57–67, 2016.

[33] A. Singer, R. R. Coifman, F. J. Sigworth, D. W. Chester, and Y. Shkolnisky. Detecting consistent common lines in cryo-EM by voting. *J. Struct. Biol.*, 169:312–322, 2009.

[34] S. Sreehari, S. V. Venkatakrishnan, L. Drummy, J. Simmons, and C. A. Bouman. Rotationally-invariant non-local means for image denoising and tomography. In *Proc. IEEE ICIP*, 2015.

[35] G. W. Stewart. Fredholm, Hilbert, Schmidt. Three fundamental papers on integral equations, 2011. Translation with commentary. `http://users.umiacs.umd.edu/~stewart/FHS.pdf`.

[36] R. Szeliski. Image alignment and stitching: A tutorial. *Found. Trends Comput. Graph. Vis.*, 4(1):1–104, 2006.

[37] G. Tang, L. Peng, P. Baldwin, D. Mann, W. Jiang, I. Rees, and S. Ludtke. EMAN2: An extensible image processing suite for electron microscopy. *J. Struct. Biol.*, 157:38–46, 2007.

[38] L. N. Trefethen and D. Bau III. *Numerical Linear Algebra*. SIAM, 1997.

[39] M. van Heel. Angular reconstitution: A posteriori assignment of projection directions for 3D reconstruction. *Ultramicroscopy*, 21:111–123, 1987.

[40] L. Wang, A. Singer, and Z. Wen. Orientation determination from cryo-EM images using least unsquared deviations. *SIAM J. Imaging Sci.*, 6(4):2450–83, 2013.

[41] J. Wimp. Polynomial expansions of Bessel functions and some associated functions. *Math. Comp.*, 16(80):446–458, 1962.

[42] Z. Xu, A. Horwich, and P. Sigler. The crystal structure of the asymmetric GroEL–GroES–(ADP)7 chaperonin complex. *Nature*, 388:741–750, 1997.

[43] Z. Yang and P. Penczek. Cryo-EM image alignment based on nonuniform fast Fourier transform. *Ultramicroscopy*, 108:959–969, 2008.

[44] Z. Zhao, Y. Shkolnisky, and A. Singer. Fast steerable principal component analysis. *IEEE Trans. Comput. Imaging*, 2(1):1–12, 2016.

[45] Z. Zhao and A. Singer. Rotationally invariant image representation for viewing direction classification in cryo-EM. *J. Struct. Biol.*, 186:153–166, 2014.

[46] S. Zimmer, S. Didas, and J. Weickert. A rotationally invariant block matching strategy improving image denoising with non-local means. In *Proc. LNLA*, 2008.