Published in final edited form as:
IEEE Trans Image Process. 2011 November ; 20(11): 3051–3062. doi:10.1109/TIP.2011.2147323.

Computing Steerable Principal Components of a Large Set of Images and Their Rotations

Colin Ponce and
Department of Computer Science, Cornell University, Ithaca, NY 14850 USA

Amit Singer
Department of Mathematics and PACM, Princeton University, Princeton NJ 08544-1000 USA

Colin Ponce: cponce@cs.cornell.edu; Amit Singer: amits@math.princeton.edu

Abstract

We present here an efficient algorithm to compute the Principal Component Analysis (PCA) of a large image set consisting of images and, for each image, the set of its uniform rotations in the plane. We do this by pointing out the block circulant structure of the covariance matrix and utilizing that structure to compute its eigenvectors. We also demonstrate the advantages of this algorithm over similar ones with numerical experiments. Although it is useful in many settings, we illustrate the specific application of the algorithm to the problem of cryo-electron microscopy.

Index Terms

EDICS Category; TEC-PRC image and video processing techniques

I. Introduction

IN image processing and computer vision applications, often one is not interested in the raw pixels of images used as input, but wishes to transform the input images into a representation that is meaningful to the application at hand. This usually comes with the added advantage of requiring less space to represent each image, effectively resulting in compression. Because less data are required to store each image, algorithms can often operate on images in this new representation more quickly.

One common transformation is to project each image onto a linear subspace of the image space. This is typically done using Principal Component Analysis (PCA), also known as the Karhunen–Loéve expansion. The PCA of a set of images produces the optimal linear approximation of these images in the sense that it minimizes the sum of squared reconstruction errors. As an added benefit, PCA often serves to reduce noise in a set of images.

PCA is typically computed as the singular value decomposition (SVD) of the data matrix X of image vectors or as the eigenvector decomposition of the covariance matrix $C = XX^T$. One then ranks eigenvectors according to their eigenvalues and projects each image onto the linear subspace spanned by the top n eigenvectors. One can then represent each image with only n values, instead of the original number of pixel values.

There are a number of applications in which each of an image's planar rotations are interesting as well as the original images. It is usually unreasonable due to both time and

space constraints to replicate each image many times at different rotations and to compute the resulting covariance matrix for eigenvector decomposition. To cope with this, Teague, [19] and then Khotanzad and Lu [10], developed a means of creating rotation-invariant image approximations based on Zernike polynomials. While Zernike polynomials are not adaptive to the data, PCA produces an optimal data adaptive basis (in the least squares sense). This idea was first utilized in the 1990s, when, in optics, Hilai and Rubinstein [8] developed a method of computing an invariant Karhunen–Loéve expansion, which they compared with the Zernike polynomials expansion. Independently, a similar method to produce an approximate set of steerable kernels was developed by Perona [14] in the context of computer vision and machine learning. Later, Uenohara and Kanade [20] produced, and Park [12] corrected, an algorithm to efficiently compute the PCA of an image and its set of uniform rotations. Jogan *et al.* [9] then developed an algorithm to compute the PCA of a set of images and their uniform rotations.

The natural way to represent images when rotating them is to sample them on a polar grid consisting of N_θ radial lines at evenly spaced angles and N_r samples along each radial line. The primary advantage of [9] is that the running time of the algorithm increases nearly linearly with both the size of the image (N_r below) and the number of rotations (N_θ below). The disadvantage, however, is that the running time experiences cubic growth with respect to the number of images P in the set. The result is an algorithm with a running time of $O(P^2 N_r N_\theta \log(N_r N_\theta) + N_\theta P^3)$, which is impractical for large sets of images.

We present here an alternate algorithm that grows cubically with the size of the image, but only linearly with respect to the number of rotations and the number of images. Due to this running time, our algorithm is appropriate for computing the PCA of a large set of images with a running time of $O(PN_\theta N_r(N_r+\log(N_\theta))+N_\theta N_r^3)$.

Although this algorithm is generally useful in many applications, we discuss its application to cryo-electron microscopy (cryo-EM) [3]. In cryo-EM, many copies of a molecule are embedded in a sheet of ice so thin that images of the molecule of interest are disjoint when viewing the sheet flat. The sheet is then imaged with an electron microscope, destroying the molecules in the process. The result is a set of projection images of the molecule taken at unknown random orientations with extremely low signal-to-noise ratio (SNR). The cryo-EM problem, then, is to reconstruct the 3-D structure of the molecule using thousands of these images as input.

Because the images are so noisy, preprocessing must be done to remove noise from the images. The first step is usually to compute the PCA of the image set [1], [21], a step which both reduces noise and compresses the image representation. This is where our algorithm comes in, because in cryo-EM every projection image is equally likely to appear in all possible in-plane rotations. After PCA, class averaging is usually performed to estimate which images are taken from similar viewing angles and to average them together in order to reduce the noise [3], [17], [22]. A popular method for obtaining class averages requires the computation of rotationally invariant distances between all $P(P-1)/2$ pairs of images [13]. The principal components we compute in this paper provide a way to accelerate this large-scale computation. This particular application will be discussed in a separate publication [18].

Other applications for computing the PCA of a large set of images with their uniform rotations include those discussed in [9]. A set of full-rotation panoramic images were taken by a robot at various points around a room. Those panoramic images and their uniform rotations were then processed using PCA. A new image taken at an unknown location in the room was then template matched against the set of images already taken to determine the

location in the room from which the new image was taken. Because these images are likely to be somewhat noisy, it is important to perform PCA on them to be able to effectively template match against the training set.

The remainder of this paper is organized as follows. In Section II, we derive the principles behind the algorithms. In Section III, we describe an efficient implementation of our algorithm and discuss its space and computational complexity. In Section IV, we present a method for applying this algorithm to images sampled in rectangular arrays. In Section V, we present numerical experiments comparing our algorithm to other existing algorithms. Finally, in Section VI, we present an example of its use in cryo-EM.

II. Derivation

Consider a set of P images represented in polar coordinates. Each image is conveniently stored as an $N_r \times N_\theta$ matrix, where the number of columns N_θ is the number of radial lines and the number of rows N_r is the number of samples along each radial line. Note that N_θ and N_r must be the same for all images. Then, each column represents a radial line beginning in the center of the image, and each row represents a circle at a given distance from the center. The first circle is at distance 0 from the center, and so the center value is repeated N_θ times as the first row of the matrix.

We can create a rotation of an image simply by rotating the order of the columns. Moving the leftmost column to the rightmost and cyclically shifting all other columns of the matrix, for example, represents a rotation of $2\pi/N_\theta$. In this way, we can generate all N_θ rotations of each image. Thus, let matrix $I^{i,l}$ represent the lth rotation of image i, $0 \le i \le P-1$, $0 \le l \le N_\theta-1$. Note that we may assume the average pixel value in each image is 0, otherwise we subtract the mean pixel value from all pixels.

Let $J^{i,l}$ be a vector created by row-major stacking of the entries of image $I^{i,l}$, that is, $J^{i,l}_{N_\theta m+n} = I^{i,l}_{m,n}$, $0 \le m \le N_r-1$, $0 \le n \le N_\theta-1$. Then construct the $N_rN_\theta \times PN_\theta$ data matrix X by stacking all $J^{i,l}$ side by side to create

$$X = \begin{bmatrix} | & & | \\ J^{0,0} & \cdots & J^{P-1,N_\theta-1} \\ | & & | \end{bmatrix} \quad (1)$$

and set the covariance matrix $C = XX^T$. Note that C is of size $N_rN_\theta \times N_rN_\theta$. The principal components that we wish to compute are the eigenvectors of C. This can be computed directly through the eigenvector decomposition of C, or through the SVD of X. We will now detail an alternative and more efficient approach for computing these eigenvectors exploiting the block circulant structure of C.

Consider C^i, the covariance matrix given by just image i and its rotations, $J^{i,0} \ldots J^{i,N_\theta-1}$ as follows:

$$C^i = \sum_{n=0}^{N_\theta-1} J^{i,n} J^{i,n^T}. \quad (2)$$

Note that

$$C=\sum_{i=0}^{P-1} C^i. \quad (3)$$

Now, let $C^{i,\alpha\beta}$ be an $N_\theta \times N_\theta$ block of C^i corresponding to the outer product of row α of $I^{i,0}$, …, $I^{i,N_\theta-1}$ in $J^{i,0}$, …, $J^{i,N_\theta-1}$ with row β of $I^{i,0}$, …, $I^{i,N_\theta-1}$.

We will now show that C^i is of the form

$$C^i=\begin{bmatrix} C^{i,0,0} & \cdots & C^{i,0,N_r-1} \\ \vdots & \ddots & \vdots \\ C^{i,N_r-1,0} & \cdots & C^{i,N_r-1,N_r-1} \end{bmatrix} \quad (4)$$

where each $C^{i,\alpha,\beta}$ is an $N_\theta \times N_\theta$ circulant matrix.

Because image i and its rotations comprise a full rotation of row α and a full rotation of row β, we have

$$C^{i,\alpha,\beta}_{n_1,n_2}=\sum_{n=0}^{N_\theta-1} I^{i,n}_{\alpha,n_1} \cdot I^{i,n}_{\beta,n_2}=\sum_{n=0}^{N_\theta-1} I^{i,0}_{\alpha,n_1+n} \cdot I^{i,0}_{\beta,n_2+n} \quad (5)$$

where each row of $I^{i,0}$ is considered a circular vector, so that all indexes are taken modulo N_θ. However, due to the circular property of the vector, for any integer c, we have

$$C^{i,\alpha,\beta}_{n_1+c,n_2+c}=\sum_{n=0}^{N_\theta-1} I^{i,0}_{\alpha,n_1+c+n} \cdot I^{i,0}_{\beta,n_2+c+n}=C^{i,\alpha,\beta}_{n_1,n_2}. \quad (6)$$

This means that $C^{i,\alpha,\beta}$ is a circulant matrix. However, this analysis applies to any rows α and β. Therefore, C^i consists of $N_r \times N_r$ tiles of $N_\theta \times N_\theta$ circulant matrices, as in (4).

Now, the sum of circulant matrices is also a circulant matrix. Define $C^{\alpha,\beta}$ as

$$C^{\alpha,\beta}=\sum_{i=0}^{P-1} C^{i,\alpha,\beta}. \quad (7)$$

Thus, $C^{\alpha,\beta}$ is an $N_\theta \times N_\theta$ circulant matrix. Then C is of the form

$$C=\begin{bmatrix} C^{0,0} & \cdots & C^{0,N_r-1} \\ \vdots & \ddots & \vdots \\ C^{N_r-1,0} & \cdots & C^{N_r-1,N_r-1} \end{bmatrix}. \quad (8)$$

A known fact is that the eigenvectors V^0, V^1, …, $V^{N\theta-1}$ of any $N_\theta \times N_\theta$ circulant matrix A are the columns of the discrete Fourier transform matrix [5], given by

$$V^k_n=e^{-2\pi\imath kn/N_\theta}, \quad 0 \le k, n \le N_\theta-1 \quad (9)$$

where $\iota = \sqrt{-1}$. The first coordinate of any V^k is 1, so the eigenvalues $\lambda_0, \lambda_1, \ldots, \lambda_{N_\theta - 1}$ of a circulant matrix A, for which, $AV^k = \lambda_k V^k$ are given by

$$\begin{aligned} \lambda_k = (AV^k)_0 &= \sum_{n=0}^{N_\theta - 1} A_{0,n} e^{-\iota 2\pi kn/N_\theta} \\ &= \widehat{A_0}(k), \quad 0 \le k \le N_\theta - 1 \end{aligned} \tag{10}$$

where A_0 is the first row in matrix A, and $\hat{A}_0$ is its discrete Fourier transform. These eigenvalues are, in general, complex.

It follows that, for the matrix C, there are N_θ eigenvalues associated with each of its N_r^2 circulant tiles. We denote these eigenvalues $\lambda_k^{\alpha,\beta}$ satisfying $C^{\alpha,\beta} V^k = \lambda_k^{\alpha,\beta} V^k$. Note that

$$\begin{aligned} \lambda_k^{\alpha,\beta} &= \sum_{n=0}^{N_\theta - 1} C_{0,n}^{\alpha,\beta} e^{-\iota 2\pi kn/N_\theta} \\ &= \sum_{n=0}^{N_\theta - 1} \left(\sum_{i=0}^{P-1} C_{0,n}^{i,\alpha,\beta} \right) e^{-\iota 2\pi kn/N_\theta} \\ &= \sum_{i=0}^{P-1} \lambda_k^{i,\alpha,\beta}. \end{aligned} \tag{11}$$

Now construct a vector $W^{k,q}$ of length $N_r N_\theta$ by concatenating multiples $c_\alpha^{k,q}$ of the vector V^k N_r times, that is,

$$W^{k,q} = \begin{bmatrix} c_0^{k,q} V^k \\ \vdots \\ c_{N_r-1}^{k,q} V^k \end{bmatrix}, \quad 0 \le k \le N_\theta - 1. \tag{12}$$

These $W^{k,q}$ vectors will be our eigenvectors for C. The exact values of $c_0^{k,q}, c_1^{k,q}, \ldots, c_{N_r-1}^{k,q}$, as well as the index set for q, will be determined below.

Consider the result of the matrix-vector multiplication $CW^{k,q}$. Along each tile-row of $C(\alpha = \alpha_0)$, each circulant submatrix $C^{\alpha_0,\beta}$ is multiplied by an eigenvector $c_\beta^{k,q} V^k$. Thus

$$CW^{k,q} = \begin{bmatrix} \left(\sum_{\beta=0}^{N_r-1} \lambda_k^{0,\beta} c_\beta^{k,q} \right) \cdot V^k \\ \vdots \\ \left(\sum_{\beta=0}^{N_r-1} \lambda_k^{N_r-1,\beta} c_\beta^{k,q} \right) \cdot V^k \end{bmatrix}. \tag{13}$$

So, requiring $W^{k,q}$ to be an eigenvector of C satisfying $CW^{k,q} = \mu^{k,q} W^{k,q}$ is equivalent to

$$\sum_{\beta=0}^{N_r-1} \lambda_k^{\alpha,\beta} c_\beta^{k,q} = \mu^{k,q} c_\alpha^{k,q}, \quad \text{for } \alpha = 0, 1, \ldots, N_r - 1. \tag{14}$$

However, this in turn is equivalent to the eigenvector decomposition problem for the matrix

$$\Lambda^k = \begin{bmatrix} \lambda_k^{0,0} & \cdots & \lambda_k^{0,N_r-1} \\ \vdots & \ddots & \vdots \\ \lambda_k^{N_r-1,0} & \cdots & \lambda_k^{N_r-1,N_r-1} \end{bmatrix}. \quad (15)$$

This means that the vector $W^{k,q}$ is an eigenvector of C with eigenvalue $\mu^{k,q}$ if the vector $c^{k,q}=(c_0^{k,q},\ldots,c_{N_r-1}^{k,q})^T$ is an eigenvector of the matrix Λ^k, satisfying $\Lambda^k c^{k,q} = \mu^{k,q} c^{k,q}$.

Next, we show that the matrix Λ^k is Hermitian. To that end, note that the covariance matrix C is by definition symmetric, so column γ of the $N_\theta \times N_\theta$ submatrix $C^{\alpha,\beta}$ is equal to row γ of submatrix $C^{\beta,\alpha}$. Now, set $\gamma = 0$. Then, by viewing the rows of the tile $C^{\alpha,\beta}$ as circular vectors, the circulant property of the tiles implies that

$$C_{0,n}^{\beta,\alpha} = C_{n,0}^{\alpha,\beta} = C_{0,-n}^{\alpha,\beta}, \quad \text{for } n=0,\ldots,N_\theta-1. \quad (16)$$

The eigenvalues of $C^{\beta,\alpha}$, then, are

$$\begin{aligned} \lambda_k^{\beta,\alpha} &= (C^{\beta,\alpha} V^k)_0 = \sum_{n=0}^{N_\theta-1} C_{0,n}^{\beta,\alpha} \cdot e^{-\iota 2\pi kn/N_\theta} \\ &= \sum_{n=0}^{N_\theta-1} C_{0,-n}^{\alpha,\beta} \cdot e^{-\iota 2\pi kn/N_\theta} \end{aligned} \quad (17)$$

which, by the change of variables $n' = -n$, becomes

$$\lambda_k^{\beta,\alpha} = \sum_{n'=0}^{N_\theta-1} C_{0,n'}^{\alpha,\beta} \cdot e^{\iota 2\pi kn'/N_\theta} = \overline{\lambda_k^{\alpha,\beta}} \quad (18)$$

where $\overline{\lambda_k^{\alpha,\beta}}$ refers to the complex conjugate of $\lambda_k^{\alpha,\beta}$. Note that this property is true only because $C_{n_1,n_2}^{\alpha,\beta}$ is real for all n_1, n_2.

Thus, Λ^k is a Hermitian matrix. As a result, all of Λ^k's eigenvalues are real. Furthermore, these eigenvalues are the $\mu^{k,q}$ so $q \in 0, 1, \ldots, N_r-1$ and the $c^{k,q}$ are complex-valued.

Thus, for some fixed k_0, Λ^{k0} has N_r linearly independent eigenvectors that in turn give N_r linearly independent eigenvectors $W^{k,q}$ of C, through (12). Furthermore, the vectors V^0, V^1, …, $V^{N\theta-1}$ are linearly independent. From this, it follows that the constructed eigenvectors $W^{k,q}$ are linearly independent.

Now, consider an eigenvector $W^{k,q}$, $0 \le k \le N_\theta-1$, $0 \le q \le N_r-1$ as an eigenimage, that is, consider the eigenimage as an $N_r \times N_\theta$ matrix

$$R^{k,q} = \begin{bmatrix} c_0^{k,q} V^{k^T} \\ \vdots \\ c_{N_r-1}^{k,q} V^{k^T} \end{bmatrix}. \quad (19)$$

This is of the same form as the original images, in which each row is a circle and each column is a radial line, with the first row being the center of the image. The coefficients $c_r^{k,q}$ are complex, so we may write $c_r^{k,q}=f^{k,q}(r)e^{\iota\phi^{k,q}(r)}$ for $0 \le r \le N_r-1$. Therefore, $R^{k,q}$ is of the form

$$\begin{aligned} R^{k,q}(r,\theta) &= f^{k,q}(r)e^{\iota\phi^{k,q}(r)}e^{-\iota k\theta} \\ &= f^{k,q}(r)e^{\iota(-k\theta+\phi^{k,q}(r))} \end{aligned} \quad (20)$$

for $\theta = 0, 2\pi/N_\theta, \ldots, 2\pi(N_\theta-1)/N_\theta$, $r = 0, 1, \ldots, N_r-1$.

The real and imaginary parts of $R^{k,q}$ are eigenimages, given by

$$\begin{aligned} R_c^{k,q}(r,\theta) &= \Re(R^{k,q}(r,\theta)) \\ &= f^{k,q}(r)\cos(-k\theta+\phi^{k,q}(r)) \end{aligned} \quad (21)$$

$$\begin{aligned} R_s^{k,q}(r,\theta) &= \Im(R^{k,q}(r,\theta)) \\ &= f^{k,q}(r)\sin(-k\theta+\phi^{k,q}(r)) \end{aligned} \quad (22)$$

for $r = 0,1, \ldots, N_r - 1$, $\theta = 0,2\pi/N_\theta, \ldots, 2\pi(N_\theta-1)/N_\theta$.

Thus, each eigenimage $R^{k,q}$ produces two eigenimages $R_c^{k,q}$ and $R_s^{k,q}$. It may appear as though this gives us $2N_rN_\theta$ eigenimages, which is too many. However, consider (11) for $V^{N-1-k} = V^{-k}$:

$$\begin{aligned} \lambda_{-k}^{\alpha,\beta} &= \sum_{n=0}^{N_\theta-1} C_{0,n}^{\alpha,\beta} e^{-\iota 2\pi(-k)n/N_\theta} \\ &= \sum_{n=0}^{N_\theta-1} \left(\sum_{i=0}^{P-1} C_{0,n}^{i,\alpha,\beta} \right) e^{\iota 2\pi kn/N_\theta} \\ &= \sum_{i=0}^{P-1} \overline{\lambda_k^{i,\alpha,\beta}} \\ &= \overline{\lambda_k^{\alpha,\beta}}. \end{aligned} \quad (23)$$

Therefore, $\Lambda^{-k}=\overline{\Lambda^k}$, Thus

$$\Lambda^{-k}\overline{c^{k,q}} = \overline{\Lambda^k c^{k,q}} = \overline{\mu^{k,q} c^{k,q}}. \quad (24)$$

However, because Λ^k is Hermitian, $\mu^{k,q}$ is real, so $\mu^{k,q}=\overline{\mu^{k,q}}$. Therefore, the eigenvalues of Λ^k are the same as those for Λ^{-k}, and the eigenvectors of Λ^k are the complex conjugates of the eigenvectors of Λ^{-k}. Note also that $k = -k$ if $k = 0$ or $k = N_\theta/2$, which are also the values of k such that Λ^k has real eigenvectors. Therefore, each eigenvector from $k = 0$ or $k = N_\theta/2$ contributes one real eigenimage, while eigenvectors from every other value of k each contribute two. Thus, linear combinations of the eigenvectors of the first $N_\theta/2 + 1$ Λ^k matrices results in $N_r \times N_\theta$ eigenimages.

Note also that, when N_θ is odd, eigenvectors for $k = 0$ each contribute one real eigenimage, while eigenvectors for every other k each contribute two. Thus, counting eigenimages for $k = 0, 1, \ldots, N_\theta-1/2$ produces $N_r \times N_\theta$ eigenimages, as required.

III. Implementation and Computational Complexity

The first step in this algorithm is to compute the covariance matrix C^i of each image I^i and its rotations. However, it is not necessary to compute the actual covariance matrix. Instead, we need only compute the eigenvalues $\lambda_k^{i,\alpha,\beta}$ of each circulant tile $C^{i,\alpha\beta}$ of the covariance matrix. From (10) and (11), it follows that the eigenvalues are equal to the discrete Fourier transform of the top row of a tile $C^{i,\alpha\beta}$.

Note that, by (5), we have

$$C_{0,m}^{i,\alpha,\beta} = \sum_{n=0}^{N_\theta - 1} I_{\alpha,n}^{i,0} I_{\beta,m+n}^{i,0} \quad (25)$$

where $m = 0, \ldots, N_\theta - 1$. Now, the convolution theorem for cross correlations states that, for two sequences f and g, if $f \star g$ indicates the cross correlation of f and g, then

$$\widehat{f \star g} = \overline{\hat{f}} \cdot \hat{g}. \quad (26)$$

Therefore, the eigenvalues $\lambda_k^{i,\alpha,\beta}$ can be computed efficiently as

$$\begin{aligned} \lambda_k^{i,\alpha,\beta} &= \overline{\left(\sum_{n=0}^{N_\theta - 1} I_{\alpha,n}^{i,0} e^{-\iota 2\pi k n / N_\theta} \right)} \cdot \left(\sum_{n=0}^{N_\theta - 1} I_{\beta,n}^{i,0} e^{-\iota 2\pi k n / N_\theta} \right) \quad (27) \\ &= \overline{\hat{I}_\alpha^{i,0}}(k) \cdot \hat{I}_\beta^{i,0}(k). \end{aligned}$$

This can be computed efficiently with the dast Fourier transform (FFT). In addition, because the covariance matrix C^i is symmetric, $\lambda_k^{i,\alpha,\beta} = \overline{\lambda_k^{i,\beta,\alpha}}$. We can in this way compute N_θ different $N_r \times N_r$ matrices $\Lambda^{i,k}$ and sum

$$\Lambda^k = \sum_{i=0}^{P-1} \Lambda^{i,k} \quad (28)$$

so that Λ^k is as defined in (15).

Note that, in some implementations, such as in Matlab, complex eigenvectors of unit norm are always returned. As a result, real eigenvectors, which occur when $k = 0$ or $k = N_\theta/2$, will have twice the norm of the other eigenvectors. One must therefore normalize these eigenvectors by dividing them by $\sqrt{2}$.

Computation of the Fourier transforms $\hat{I}_\alpha^{i,0}$ can be performed in time $O(N_\theta \log(N_\theta))$. Because we do this for every α, computation of the necessary Fourier transforms takes total time $O(N_r N_\theta \log(N_\theta))$. In addition, it uses space $O(N_\theta N_r)$. Computation of a matrix Λ^k has computational complexity $O(N_r^2)$, but, because we compute N_θ of them, the overall time of $O(N_\theta N_r^2)$. In addition, the space complexity of this step is $O(N_\theta N_r^2)$. Each of these steps must be repeated for each image i, and then the P matrices $\Lambda^{i,k}$, $(i = 0, \ldots, P-1)$must be summed together for each image. Thus, the computational complexity up to this point is $O(P N_\theta N_r (N_r + \log(N_\theta)))$and the space complexity is $O(N_\theta N_r^2)$.

Note that the computation of Λ^k for each image i is completely independent of each other image. Thus, these computations can be performed in parallel to further improve performance. Such a parallel implementation requires space $O(PN_\theta N_r^2)$.

At this point, we need to perform eigenvector decomposition on each Λ^k to determine the eigenvectors of C. The computational complexity of this step is $O(N_\theta N_r^3)$ and the space complexity is $O(P N_\theta N_r)$. Thus, the computational complexity of the entire algorithm is

$$O(PN_\theta N_r(N_r+\log(N_\theta))+N_\theta N_r^3) \quad (29)$$

and the total space complexity is

$$\max(N_\theta N_r^2, PN_\theta N_r). \quad (30)$$

The algorithm is summarized in Algorithm 1, and a comparison of the computational complexity of this algorithm with others is shown in Table I.[1]

Algorithm 1

PCA of a Set of Images and Their Rotations

Require: P image matrices $I^{0,0}$, $I^{1,0}$,…, $I^{P-1,0}$ of size $N_r \times N_\theta$ in polar form.

1: Set $\lambda_k^{\alpha,\beta}=0$, $\alpha, \beta = 0,\ldots, N_r-1$, $k = 0,\ldots, N_\theta-1$

2: **for** $i = 0,\ldots, P-1$ **do**

3: Compute the Fourier transforms $\widehat{I}_\alpha^{i,0}(k)$ for $\alpha = 0,\ldots, N_r-1$, $k=0,\ldots, N_\theta-1$.

4: **for** $\alpha\beta = 0,\ldots, N_r-1$ **do**

5: Compute the eigenvalues $\lambda_k^{i,\alpha,\beta}$ of $C^{i,\alpha\beta}$ using (27).

6: **end for**

7: Update $\Lambda^k \leftarrow \Lambda^k + \Lambda^{i,k}$ for $k = 0,\ldots, N_\theta-1$ as in (28).

8: **end for**

9: **for** $k = 0,\ldots, N_\theta/2$ **do**

10: Compute the eigenvector decomposition of matrix Λ^k to generate eigenvectors $c^{k,q}$ and eigenvalues $\mu^{k,q}$ for $q = 0, \ldots, N_r-1$

11: **for** $q = 0,\ldots,N_r-1$ **do**

12: Construct eigenvector $W^{k,q}$ as in (12), using eigenvector $c^{k,q}$ of Λ^k.

13: **end for**

14: **end for**

15: **for** $k = 0,\ldots, N_\theta/2$, $q = 0,\ldots,N_r-1$ **do**

16: Rearrange $W^{k,q}$ as an image matrix as in (19) and compute its real and imaginary parts $R_c^{k,q}$ and $R_s^{k,q}$.

17: If $k = 0$ or $k = N_\theta/2$, normalize eigenimages if necessary by dividing by $\sqrt{2}$.

---

[1] A Matlab implementation of this algorithm can be found at http://www.cs.cornell.edu/cponce/SteerablePCA/

18: **end for**

19: **return** $R_c^{k,q}$ and $R_s^{k,q}$ as in (21) and (22).

IV. Application to Rectangular Images

Although this algorithm is based on a polar representation of images, it can be implemented efficiently on images represented in the standard Cartesian manner efficiently and without loss of precision using the 2-D polar Fourier transform and the Radon transform.

The 2-D Radon transform of a function $I(x, y)$ along a line $\tau_\theta^\perp$ through the origin is given by the projection of $I(x, y)$ along the direction τ_θ [11]. The Radon transform is defined on the space of lines $\tau_\theta^\perp$ in $\mathbb{R}^2$, so we may write

$$\begin{aligned} R(t, \tau_\theta^\perp) &= \int_{\tau_\theta} I(x, y)\, ds \\ &= \int\int_{-\infty}^{\infty} I(x, y) \times \delta(x\cos(\theta) + y\sin(\theta) - t)\, dxdy. \end{aligned} \quad (31)$$

The Fourier projection-slice theorem states that the 1-D Fourier transform of the projection of an image along τ_{θ_0} is equal to the slice of the 2-D Fourier transform of the image taken along $\tau_{\theta_0}^\perp$ [11]. To see this, let $r \in \mathbb{R}^2$ be any point on the plane, and let

$$r = u + s, \quad u \in \tau_{\theta_0}^\perp, \quad s \in \tau_{\theta_0}. \quad (32)$$

Also, take $w \in \tau_{\theta_0}^\perp$. Note that $r \cdot w = u \cdot w$, since $s \cdot w = 0$. Then

$$\begin{aligned} \widehat{R}_{\theta_0}(w) &= \int_{\tau_{\theta_0}^\perp} \int_{\tau_{\theta_0}} I(u+s)\, ds\, e^{-\imath u \cdot w} du \\ &= \int_{\tau_{\theta_0}^\perp} \int_{\tau_{\theta_0}} I(u+s) e^{-\imath u \cdot w} ds du \\ &= \int_{\mathbb{R}^2} I(r) e^{-\imath r \cdot w} dr \\ &= \widehat{I}(w). \end{aligned} \quad (33)$$

The Fourier projection-slice theorem provides an accurate way for computing the Radon transform by sampling the image's 2-D Fourier transform on a polar grid and then taking inverse 1-D Fourier transforms along lines through the origin. Because the 2-D Fourier transform is discrete, performing the normal 2-D FFT and interpolating it to a polar grid leads to approximation errors. Instead, we use a nonequally spaced FFT described in [2] and [6]. This step has a computational complexity of $O((mL)^2 \log(L) + N_r N_\theta \log(N_r N_\theta) \log(1/\varepsilon))$, where L is the number of pixels on the side of a square true image, N_r is the number of pixels in a radial line in polar form, N_θ is the number of radial lines, m is an oversampling factor used in the nonequally spaced FFT, and ε is the required precision.[2]

The Fourier transform is a unitary transformation by Parseval's theorem. Therefore, the principal components of the normal 2-D FFT'ed images are simply the 2-D FFT of the principal component images. The 2-D polar Fourier transform, however, is not a unitary

[2]We would like to thank Yoel Shkolnisky for sharing with us his code for the 2-D polar Fourier transform. In his code, $m = 2$ and ε is single precision accuracy.

transformation, because the frequencies are nonequally spaced: they are sampled on concentric circles of N_r equally spaced radii, such that every circle has a constant number N_θ of samples instead of being proportional to its circumference, or equivalently, to its radius. In order to make it into a unitary transformation, we have to multiply the 2-D Fourier transform $\hat{I}(|w|\cos\theta, |w|\sin\theta)$ of image I by $\sqrt{|w|}$. In other words, all images are filtered with the radially symmetric "wedge" filter $\sqrt{|w|}$ or, equivalently, are convolved with the inverse 2-D Fourier transform of this filter.

After multiplying the images' 2-D polar Fourier transforms by $\sqrt{|w|}$, we take the 1-D inverse Fourier transform of every line that passes through the origin. From the Fourier projection-slice theorem, these 1-D inverse Fourier transforms are equal to the line projections of the convolved images, that is, the collection of 1-D inverse Fourier transforms of a given "$\sqrt{|w|}$-filtered" image, is the 2-D Radon transform of the filtered image.

These 2-D Radon transformed filtered images are real valued and are given in polar form that can be organized into matrices as described at the beginning of Section II. We can then utilize the algorithm described above to compute the principal components of such images. Moreover, since the transformation from the original images to the Radon transformed filtered images is unitary, we are guaranteed that the computed principal components are simply the Radon transform of the filtered principal components images.

After computing all of the polar Radon principal components we desire, we must convert them back to rectangular coordinates. We cannot simply invert the Radon transform because the pixels of each eigenimage are not uniformly sampled, making the problem ill-formed. However, for the inversion we can utilize the special form of the Radon eigenimages $R_c^{k,q}$ and $R_s^{k,q}$ from (21) and (22), given by

$$R_c^{k,q}(r,\theta) = f^{k,q}(r)\cos(-k\theta + \phi^{k,q}(r)) \quad (34)$$

$$R_s^{k,q}(r,\theta) = f^{k,q}(r)\sin(-k\theta + \phi^{k,q}(r)). \quad (35)$$

To compute the inverse Radon transform of such functions we utilize the projection-slice theorem. In principle, this is done by first computing the Fourier transform of each radial line, dividing by $\sqrt{|w|}$ and then taking the 2-D inverse Fourier transform of that polar Fourier image to produce the true eigenimage.

Let us first compute the 2-D polar Fourier transform by taking 1-D discrete Fourier transform along each radial line of the function in (34). Equation (34) is only defined for $r \geq 0$, but we must compute the discrete Fourier transform (DFT) along an entire line through the origin. We must therefore expand this definition to allow for any real r. Let

$$g_c^{k,\theta_0}(x) = \begin{cases} R_c^{k,q}(x,\theta_0), & \text{if } x \geq 0 \\ R_c^{k,q}(-x,\theta_0+\pi), & \text{if } x<0 \end{cases} \quad (36)$$

for some fixed θ_0. Note that we should really denote this function g_c^{k,q,θ_0}, but since q is fixed throughout this section, we drop it for convenience of notation. Then

$$g_c^{k,\theta_0}(x)=\begin{cases} f^{k,q}(x)\cos(-k\theta_0+\phi^{k,q}(x)), & \text{if } x \geq 0 \\ f^{k,q}(-x)\cos(-k(\theta_0+\pi)+\phi^{k,q}(x)), & \text{if } x<0. \end{cases} \quad (37)$$

Define g_s^{k,θ_0} similarly for $R_s^{k,q}$. Note that g_c^{k,θ_0} and g_s^{k,θ_0} are even functions if k is even, and odd functions if k is odd. The DFT can then be defined as

$$F_c^{k,\theta_0}(r)=\sum_{n=-(N_r-1)}^{N_r-1} g_c^{k,\theta_0}(n)e^{-\iota 2\pi/2N_r-1rn} \quad (38)$$

where $r=|w|$. Note that

$$F_c^{k,\theta_0}(-r)=\sum_{n=-(N_r-1)}^{N_r-1} g_c^{k,\theta_0}(n)e^{\iota 2\pi/2N_r-1rn} = \overline{F_c^{k,\theta_0}(r)} \quad (39)$$

because $g_c^{k,\theta_0}(n)$ is real. Furthermore, with the change of variables $n'=-n$, we find that, if k is even, then

$$F_c^{k,\theta_0}(-r)=\sum_{n'=-(N_r-1)}^{N_r-1} g_c^{k,\theta_0}(-n')e^{-i2\pi/2N_r-1rn'} = F_c^{k,\theta_0}(r) \quad (40)$$

and, if k is odd, then

$$F_c^{k,\theta_0}(-r)=\sum_{n'=-(N_r-1)}^{N_r-1} g_c^{k,\theta_0}(-n')e^{-i2\pi/2N_r-1rn'} = -F_c^{k,\theta_0}(r). \quad (41)$$

Therefore, if k is even, then $F_c^{k,\theta_0}(r)$ is real and even, and, if k is odd, then $F_c^{k,\theta_0}(r)$ is pure imaginary and odd.

Then, the 2-D Fourier transform of the true image can be defined as (a proof of this formula can be found in Appendix A)

$$F_c^k(r,\theta)=F_c^{k,\theta}(r) = \cos(-k\theta)F_c^{k,0}(r)-\sin(-k\theta)F_s^{k,0}(r) \quad (42)$$

and

$$F_s^k(r,\theta)=F_s^{k,\theta}(r) = \sin(-k\theta)F_c^{k,0}(r)+\cos(-k\theta)F_s^{k,0}(r). \quad (43)$$

As shown above, these functions are either real or pure imaginary, depending on k.

Next we will take the 2-D inverse Fourier transform of $1/\sqrt{r}F_c^k(r,\theta)$ (where $r=|w|$ and $1/\sqrt{r}$ is the filter) to obtain true eigenimages I_c^k and I_s^k. Define H_c^k as the order k Hankel transform of $1/\sqrt{r}F_c^{k,0}(r)$

$$H_c^k(s)=\int_0^\infty r\frac{1}{\sqrt{r}}F_c^{k,0}(r)J_k(rs)\,dr \quad (44)$$

and define H_s^k as the order k Hankel transform of $1/\sqrt{r}F_s^{k,0}(r)$:

$$H_s^k(s)=\int_0^\infty r\frac{1}{\sqrt{r}}F_s^{k,0}(r)J_k(rs)\,dr, \quad (45)$$

where J_k is the Bessel function.

Then, the true real eigenimages can be described (in polar form) with the two equations (a proof of this formula can be found in Appendix B)

$$I_1^k(s,\alpha)=\beta(k)\left(\cos\left(k\left(\alpha+\frac{\pi}{2}\right)\right)H_c^k(s)+\sin\left(k\left(\alpha+\frac{\pi}{2}\right)\right)H_s^k(s)\right) \quad (46)$$

$$I_2^k(s,\alpha)=\beta(k)\left(\sin\left(k\left(\alpha+\frac{\pi}{2}\right)\right)H_c^k(s)-\cos\left(k\left(\alpha+\frac{\pi}{2}\right)\right)H_s^k(s)\right) \quad (47)$$

where

$$\beta(k)=\frac{1}{2}(1+(-1)^k+\imath\,(1-(-1)^k)) \quad (48)$$

so that $\beta(k)=1$ when k is even, and $\beta(k)=\imath$ when k is odd. In practice, H_c^k and H_s^k are computed using standard numerical procedures [4], [7], [15] and from the polar form (46)–(47), it is a straightforward manner to obtain the images' values on the Cartesian grid.[3]

V. Numerical Experiments

We performed numerical experiments to test the speed and accuracy of our algorithm against similar algorithms. We performed this test on a UNIX environment with four processors, each of which was a Dual Core AMD Opteron Processor 275 running at 2.2 GHz. These experiments were performed in Matlab with randomly generated pixel data in each image. We did this because the actual output of the algorithms are not important in measuring runtime, and runtime is not affected by the content of the images. The images are assumed to be on a polar grid as described at the beginning of Section II.

We compared our algorithm against three other algorithms: 1) the algorithm described in [9], which we will refer to as JZL; 2) SVD of the data matrix; and 3) computation of the covariance matrix C and then eigenvector decomposition of C. Note that our algorithm as well as JZL compute the PCA of each image and its planar rotations, while SVD and

[3]MATLAB code to convert polar Radon eigenimages into rectangular eigenimages can be found at http://www.cs.cornell.edu/~cponce/SteerablePCA/

eigenvector decomposition do not take rotations into account and are therefore not as accurate.

In Table II, one can see a comparison of the running times of various algorithms. In this experiment, images of size $N_r = 100$ and $N_\theta = 72$ were used. For image counts P of at least 64 and less than 12 288, our algorithm is fastest, at which point eigenvector decomposition becomes faster. However, the goal of this paper is to work with large numbers of images, so our algorithm is not designed to compete for very small P. In addition, eigenvector decomposition does not take planar rotations into account. Thus, eigenvector decomposition only computes the principal components of P images, whereas our algorithm actually computes the principal components of PN_θ images. In addition, the calculation of the covariance matrix was made faster because Matlab makes use of parallel processors to compute matrix products.

In Table III, one can see a comparison of the running times of various algorithms as $P = 500$ and $N_\theta = 72$ are held constant and N_r is increased. Even though the running time of our algorithm shows cubic growth here, it is still far faster than either JZL or eigenvector decomposition. SVD performs more quickly than our algorithm here; however, it is not considering image rotations. Note that the large jump in running time for eigenvector decomposition with $N_r = 384$ is due to the fact that, for large N_r, the $N_rN_\theta \times N_rN_\theta$ covariance matrix became too large to store in memory, causing the virtual memory to come into use, causing large delays.

In Table IV, one can see a comparison of the running times of various algorithms as $P = 500$ and $N_r = 100$ are held constant, and N_θ is increased. Again, our algorithm is significantly faster than any of the others.

We now consider the numerical accuracy of our algorithm. We compare the resulting eigenvectors of SVD against those of our algorithm and eigenvector decomposition of the covariance matrix. However, these eigenvectors often occur in pairs, and any two orthonormal eigenvectors that span the correct eigensubspace are valid eigenvectors. Therefore, we cannot simply compute the distance between eigenvectors of different methods, because the paired eigenvectors produced may be different.

To measure the accuracy of the algorithms, then, we compute the Grassmannian distance between two eigensubspaces: if a subspace V is spanned by orthonormal vectors $u_1, \ldots, u_k$ then the projection matrix onto that subspace is defined by $P_v = \sum_{i=1}^{k} v_i v_i^T$. The Grassmannian distance between two subspaces V and W is defined as the largest eigenvalue magnitude of the matrix $(P_V - P_W)$.

In our first experiment, we artificially generated 50 projection images of the *E. Coli* 50-s ribosomal subunit, as described in the next section, and convert them into "$\sqrt{w}$-filtered" radon images as described in Section IV with $N_r = 100$ and $N_\theta = 36$. We then perform PCA on these Radon images and each of their 36 rotations using the four different algorithms. We then pair off principal components with identical eigenvalues, and compute the Grassmannian distance between each set of principal components and those computed by SVD.

In our second experiment, we do the same thing but using images consisting entirely of Gaussian noise with mean 0 and variance 1. The results of both experiments can be seen in Table V.

Finally, in our third experiment, we compare the true eigenimages that are produced by our algorithm using the procedure described in Section IV with those produced by performing SVD on a full set of images and rotations. To do this, we produce artificially generated 75 projection images of the *E. Coli* 50-s ribosomal subunit and produced 36 uniform rotations of each one.[4] We then performed SVD on this set of 2700 images and compared the results with running our algorithm on the original set of 75 images. The results can be seen in Fig. 1. Eigenimages 2 and 3 are rotated differently in our algorithm than in SVD. This is because the two eigenimages have the same eigenvalue, and different rotations correspond to different eigenvectors in the 2-D subspace spanned by the eigenvectors. Eigenimage 4 is also rotated differently, which is again due to paired eigenvalues; the fifth eigenimage is not shown.

VI. Cryo-EM

Cryo-EM is a technique used to image and determine the three dimensional structure of molecules. Many copies of some molecule of interest are frozen in a sheet of ice that is sufficiently thin so that, when viewing the sheet from its normal direction, the molecules do not typically overlap. This sheet is then imaged using a transmission electron microscope. The result is a set of noisy projection images of the molecule of interest, taken at random orientations of the molecule. This process usually destroys the molecules of interest, preventing one from turning the sheet of ice to image a second time at a known orientation. The goal is to then to take a large set of such images, often 10 000 or more, and from that set deduce the three dimensional structure of the molecule.

Several sample projection images are shown in Fig. 2(a). These images were artificially generated by taking projections from random orientations of the *E. Coli* 50-s ribosomal subunit, which has a known structure. Gaussian noise was then added to create an SNR of Var(signal)/Var(noise) = 1/5. $P = 250$ such images of size 129 × 129 were generated, and then transformed into radon images of size $N_\theta = 72$ and $N_r = 100$. Note that, in a real application of cryo-EM, images often have much lower SNRs, and many more images are used.

We then applied to this set of images the algorithm described in Sections II–IV. The first 10 "$\sqrt{|w|}$-filtered" radon eigenimages are shown in Fig. 2(b). Note that eigenimages 2 and 3 are an example of paired eigenimages. Paired eigenimages have the same eigenvalue, and occur as eigenvectors in paired Hermitian matrices Λ^k and Λ^{-k}, as in (21) and (22). Note that eigenimages 1 and 6 have a frequencies of $k = 0$, which results in only a real eigenimage and so is not paired with another.

Fig. 2(c) shows the conversion of the Radon eigenimages in Fig. 2(b) to true eigenimages. This was done with the procedure described in Section IV. Eigenimages 2 and 3 are paired with frequency $k = 1$, and eigenimages 4 and 5 are paired with frequency $k = 2$. They are described by (46) and (47). Eigenimage 1 is described by (46) only. These images can be compared in their general shape to the eigenimages shown in [19, p. 117], where the SVD was used to compute the eigenimages for all possible rotations of KLH projection images. We remark that the true eigenimages are shown just for illustrative purposes: as will be shown in [18], the principal components of the 2-D $\sqrt{|w|}$-multiplied polar Fourier transformed images are sufficient to accelerate the computation of the rotationally invariant distances [13], but the true eigenimages are not required.

[4]We would like to thank Y. Shkolnisky for providing us with his code for rotating images. It produces a rotated image by upsampling using FFT, then rotating using bilinear interpolation, and finally downsampling again to the original pixels.

Acknowledgments

This work was supported by the National Institute of General Medical Sciences under Award R01GM090200. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Hsueh-Ming Hang.

The authors would like to thank Y. Shkolnisky for providing us his code for computing the 2-D Fourier transform on a polar grid, as well as his code for rotating images. They would also like to thank Z. Zhao for helpful discussions and for proofreading the paper.

References

1. Borland L, van Heel M. Classification of image data in conjugate representation spaces. J Opt Soc Amer A. 1990; 7(4):601–610.
2. Dutt A, Rokhlin V. Fast Fourier transorms for nonequispaced data. SIAM J Sci Comput. 1993; 14(6):1368–1393.
3. Frank, J. Three-Dimensional Electron Microscopy of Macromolecular Assemblies: Visualization of Biological Molecules in Their Native State. 2. New York: Oxford Univ. Press; 2006.
4. Gardner DG, Gardner JC, Lausch G, Meinke WW. Method for the analysis of multi-component exponential decays. J Chem Phys. 1959; 31:987.
5. Golub, GH.; Van Loan, CF. Matrix Computations. Baltimore, MD: Johns Hopkins Univ; 1996.
6. Greengard L, Lee JY. Accelerating the nonuniform fast Fourier transform. SIAM Rev. 2004; 46(3): 443–454.
7. Guizar-Sicairos M, Gutierrez-Vega JC. Computation of quasi-discrete Hankel transforms of integer order for propagating optical wave fields. J Opt Soc Amer A. 2004; 21:53–58.
8. Hilai R, Rubinstein J. Recognition of rotated images by invariant Karhunen-Loeve expansion. J Opt Soc Amer A. 1994; 11:1610–1618.
9. Jogan M, Zagar E, Leonardis A. Karhunen–Loéve expansion of a set of rotated templates. IEEE Trans Image Process. Jul; 2003 12(7):817–825. [PubMed: 18237956]
10. Khotanzad A, Lu JH. Object recognition using a neural network and invariant Zernike features. Proc IEEE Conf Vis Pattern Recognit. 1989:200–205.
11. Natterer, F. The Mathematics of Computerized Tomography. Philadelphia, PA: SIAM; 2001.
12. Park R. Correspondence. IEEE Trans Image Process. Mar; 2002 11(3):332–334. [PubMed: 18244635]
13. Penczek PA, Zhu J, Frank J. A common-lines based method for determining orientations for $N > 3$ particle projections simultaneously. Ultramicroscopy. 1996; 63:205–218. [PubMed: 8921628]
14. Perona P. Deformable kernels for early vision. IEEE Trans Pattern Anal Mach Intell. May; 1995 17(5):488–499.
15. Siegmann AE. Quasi fast Hankel transform. Opt Lett. 1977; 1:13–15. [PubMed: 19680315]
16. Sigworth FJ. Classical detection theory and the Cryo-EM particle selection problem. J Structural Biol. 2004; 145:111–122.
17. Singer A, Zhao Z, Shkolnisky Y, Hadani R. Viewing angle classification of cryo-electron microscopy images using eigenvectors. SIAM J Imag Sci. 2011; 4(2):723–759.
18. Singer A, Zhao Z. In Preparation.
19. Teague MR. Image analysis via the general theory of moments. J Opt Soc Amer. 1980; 70:920–930.
20. Uenohara M, Kanade T. Optimal approximation of uniformly rotated images: Relationship between Karhunen–Loeve expansion and discrete cosine transform. IEEE Trans Image Process. Jan; 1998 7(1):116–119. [PubMed: 18267385]
21. van Heel M, Frank J. Use of multivariate statistics in analysing the images of biological macromolecules. Ultramicroscopy. 1981; 6:187–194. [PubMed: 7268930]
22. van Heel M, Gowen B, Matadeen R, Orlova EV, Finn R, Pape T, Cohen D, Stark H, Schmidt R, Schatz M, Patwardhan A. Single-particle electron cryo-microscopy: Towards atomic resolution. Quarterly Rev Biophys. 2000; 33(4):307–369.

Biographies

Colin Ponce received the B.S.E. degree in computer science and a certificate in applied and computational mathematics from Princeton University, Princeton, NJ, in 2010. He is currently working toward the Ph.D. degree in computer science from Cornell University, Ithaca, NY.

His current research focuses on machine learning and scientific computing.

Mr. Ponce is a member of Tau Beta Pi and Sigma Xi.

Amit Singer received the B.Sc. degree in physics and mathematics and Ph.D. degree in applied mathematics from Tel Aviv University, Tel Aviv, Israel, in 1997 and 2005, respectively.

He is an Associate Professor of Mathematics and a member of the Executive Committee of the Program in Applied and Computational Mathematics (PACM) at Princeton University, Princeton, NJ. He joined Princeton University as an Assistant Professor in 2008. From 2005 to 2008, he was a Gibbs Assistant Professor in Applied Mathematics with the Department of Mathematics, Yale University, New Haven, CT. He served in the Israeli Defense Forces during 1997–2003. His current research in applied mathematics focuses on problems of massive data analysis and structural biology.

Prof. Singer was the recipient of the Alfred P. Sloan Research Fellowship (2010) and the Haim Nessyahu Prize for Best PhD in Mathematics in Israel (2007).

Appendix A. Proof of 2-D Fourier Transform Formula

Here, we prove formulas (42) and (43). To that end, consider the operation in (38):

$$
\begin{aligned}
F_c^{k,\theta_0}(r) &= \sum_{n=-(N_r-1)}^{N_r-1} g_c^{k,\theta_0}(n) e^{-\iota 2\pi/2N_r - 1rn} \\
&= \sum_{n=-(N_r-1)}^{-1} R_c^{k,q}(-n, \theta_0+\pi) e^{-\iota 2\pi/2N_r - 1rn} + \sum_{n=0}^{N_r-1} R_c^{k,q}(n, \theta_0) e^{-\iota 2\pi/2N_r - 1rn} \\
&= \sum_{n=-(N_r-1)}^{-1} f(-n) \\
&\quad \times \cos(-k(\theta_0+\pi)+\phi(-n)) e^{-\iota 2\pi/2N_r - 1rn} \\
&\quad + \sum_{n=0}^{N_r-1} f(n) \cos(-k\theta_0+\phi(n)) e^{-\iota 2\pi/2N_r - 1rn} \\
&= \sum_{n=-(N_r-1)}^{-1} f(-n) [\cos(-k(\theta_0+\pi)) \cos(\phi(-n)) \\
&\quad - \sin(-k(\theta_0+\pi)) \sin(\phi(-n))] e^{-\iota 2\pi/2N_r - 1rn} \\
&\quad + \sum_{n=0}^{N_r-1} f(n) [\cos(-k\theta_0) \cos(\phi(n)) \\
&\quad - \sin(-k\theta_0) \sin(\phi(n))] e^{-\iota 2\pi/2N_r - 1rn} \\
&= \cos(-k(\theta_0+\pi)) \\
&\quad \times \sum_{n=-(N_r-1)}^{-1} f(-n) \cos(\phi(-n)) e^{-\iota 2\pi/2N_r - 1rn} \\
&\quad - \sin(-k(\theta_0+\pi)) \\
&\quad \times \sum_{n=-(N_r-1)}^{-1} f(-n) \sin(\phi(-n)) e^{-\iota 2\pi/2N_r - 1rn} \\
&\quad + \cos(-k\theta_0) \sum_{n=0}^{N_r-1} f(n) \cos(\phi(n)) e^{-\iota 2\pi/2N_r - 1rn} \\
&\quad - \sin(-k\theta_0) \sum_{n=0}^{N_r-1} f(n) \sin(\phi(n)) e^{-\iota 2\pi/2N_r - 1rn}.
\end{aligned} \tag{49}
$$

Note that

$$
\begin{aligned}
\cos(-k(\theta_0+\pi)) \sum_{n=-(N_r-1)}^{-1} f(-n) \cos(\phi(-n)) e^{-\iota 2\pi/2N_r - 1rn} &= \cos(-k\theta_0)(-1)^k \sum_{n=-(N_r-1)}^{-1} f(-n) \cos(\phi(-n)) e^{-\iota 2\pi/2N_r - 1rn} \\
&= \cos(-k\theta_0) \sum_{n=-(N_r-1)}^{-1} f(-n)(-1)^k \cos(\phi(-n)) e^{-\iota 2\pi/2N_r - 1rn} \\
&= \cos(-k\theta_0) \sum_{n=-(N_r-1)}^{-1} f(-n) \cos(-k\pi+\phi(-n)) e^{-\iota 2\pi/2N_r - 1rn}
\end{aligned} \tag{50}
$$

and similarly

$$
\sin(-k(\theta_0+\pi)) \sum_{n=-(N_r-1)}^{-1} f(-n) \sin(\phi(-n)) e^{-\iota 2\pi/2N_r - 1rn} = \sin(-k\theta_0) \sum_{n=-(N_r-1)}^{-1} f(-n) \sin(-k\pi+\phi(-n)) e^{-\iota 2\pi/2N_r - 1rn}. \tag{51}
$$

Therefore, it follows that (49) becomes

$$
\begin{aligned}
&F_c^{k,\theta_0}(r)\\
&=\cos(-k\theta_0)\sum_{n=-(N_r-1)}^{-1} f(-n)\cos(-k\pi+\phi(-n))e^{-\iota 2\pi/2N_r-1rn}\\
&\quad-\sin(-k\theta_0)\sum_{n=-(N_r-1)}^{-1} f(-n)\\
&\quad\times\sin(-k\pi+\phi(-n))e^{-\iota 2\pi/2N_r-1rn}\\
&\quad+\cos(-k\theta_0)\sum_{n=0}^{N_r-1} f(n)\cos(\phi(n))e^{-\iota 2\pi/2N_r-1rn}\\
&\quad-\sin(-k\theta_0)\sum_{n=0}^{N_r-1} f(n)\sin(\phi(n))e^{-\iota 2\pi/2N_r-1rn}\\
&=\cos(-k\theta_0)\left(\sum_{n=-(N_r-1)}^{-1} R_c^{k,q}(-n,\pi)e^{-\iota 2\pi/2N_r-1rn}+\sum_{n=0}^{N_r-1} R_c^{k,q}(n,0)e^{-\iota 2\pi/2N_r-1rn}\right)\\
&\quad-\sin(-k\theta_0)\left(\sum_{n=-(N_r-1)}^{-1} R_s^{k,q}(-n,\pi)e^{-\iota 2\pi/2N_r-1rn}+\sum_{n=0}^{N_r-1} R_s^{k,q}(n,0)e^{-\iota 2\pi/2N_r-1rn}\right)\\
&=\cos(-k\theta_0)\sum_{n=-(N_r-1)}^{N_r-1} g_c^{k,0}(n)e^{-\iota 2\pi/2N_r-1rn}\\
&\quad-\sin(-k\theta_0)\sum_{n=-(N_r-1)}^{N_r-1} g_s^{k,0}(n)e^{-\iota 2\pi/2N_r-1rn}\\
&=\cos(-k\theta_0)F_c^{k,0}(r)-\sin(k\theta_0)F_s^{k,0}(r).
\end{aligned}
\tag{52}
$$

Therefore, define the 2-D Fourier transform of the true image as

$$
\begin{aligned}
&F_c^k(r,\theta)=F_c^{k,\theta}(r)\\
&=\cos(-k\theta)F_c^{k,0}(r)-\sin(-k\theta)F_s^{k,0}(r).
\end{aligned}
\tag{53}
$$

A similar manipulation shows that

$$
\begin{aligned}
&F_s^k(r,\theta)=F_s^{k,\theta}(r)\\
&=\sin(-k\theta)F_c^{k,0}(r)+\cos(-k\theta)F_s^{k,0}(r).
\end{aligned}
\tag{54}
$$

As shown above, these functions are either real or pure imaginary, depending on *k*.

Appendix B. Proof of the Radon Inversion Formula

Here, we show that the true eigenimages I_1^k and I_2^k are given by (46) and (47). To do this, we take the 2-D inverse Fourier transform of $F_c^{k,0}(r,\theta)$. First, let $\tilde{F}_c^{k,0}(w_1,w_2)=1/\sqrt{w_1^2+w_2^2}F_c^k(\sqrt{w_1^2+w_2^2},\arctan w_2/w_2)$. Then

$$
I_c^k(x_1,x_2)=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\tilde{F}_c^{k,0}(w_1,w_2)e^{\iota(w_1,w_2)\cdot(x_1,x_2)}dw_1dw_2. \tag{55}
$$

With the change of variables $x_1 = s\cos\alpha$, $x_2 = s\sin\alpha$ and $w_1 = r\cos$ &*thetas;*, $w_2 = r\sin\theta$, we obtain

$$\begin{aligned}
&I_c^k(s,\alpha)\\
&=\int_0^{2\pi}\int_0^{\infty}\tfrac{1}{\sqrt{r}}F_c^{k,0}(r,\theta)e^{\imath(r\cos\theta,r\sin\theta)\cdot(s\cos\alpha,s\sin\alpha)}r\,dr d\theta\\
&=\int_0^{2\pi}\int_0^{\infty}\tfrac{1}{\sqrt{r}}(\cos(-k\theta)F_c^{k,0}(r)\\
&-\sin(-k\theta)F_s^{k,0}(r))\,e^{\imath(r\cos\theta,r\sin\theta)\cdot(s\cos\alpha,s\sin\alpha)}r\,dr d\theta\\
&=\int_0^{2\pi}\int_0^{\infty}\tfrac{1}{\sqrt{r}}(\cos(-k\theta)F_c^{k,0}(r)\\
&-\sin(-k\theta)F_s^{k,0}(r))\,e^{\imath rs\cos(\theta-\alpha)}r\,dr d\theta\\
&=\int_0^{2\pi}\int_0^{\infty}\left(\tfrac{e^{\imath k\theta}+e^{-\imath k\theta}}{2}\right)\tfrac{1}{\sqrt{r}}F_c^{k,0}(r)e^{\imath rs\cos(\theta-\alpha)}r\,dr d\theta\\
&-\int_0^{2\pi}\int_0^{\infty}\left(\tfrac{e^{-\imath k\theta}-e^{-\imath k\theta}}{2\imath}\right)\tfrac{1}{\sqrt{r}}F_s^{k,0}(r)e^{\imath rs\cos(\theta-\alpha)}r\,dr d\theta\\
&=\tfrac{1}{2}\int_0^{\infty}r\tfrac{1}{\sqrt{r}}F_c^{k,0}(r)\left(\int_0^{2\pi}e^{\imath(k\theta+rs\cos(\theta-\alpha))}d\theta+\int_0^{2\pi}e^{-\imath(k\theta-rs\cos(\theta-\alpha))}d\theta\right)dr\\
&-\tfrac{1}{2\imath}\int_0^{\infty}r\tfrac{1}{\sqrt{r}}F_s^{k,0}(r)\times\left(\int_0^{2\pi}-e^{\imath(k\theta+rs\cos(\theta-\alpha))}d\theta+\int_0^{2\pi}e^{-\imath(k\theta-rs\cos(\theta-\alpha))}d\theta\right)dr.
\end{aligned}\tag{56}$$

Now, with the change of variables $\theta' = \alpha + \pi/2 - \theta$, we have

$$\begin{aligned}
\int_0^{2\pi}e^{\imath(k\theta+rs\cos(\theta-\alpha))}d\theta&=e^{\imath k(\alpha+\pi/2)}\int_{\alpha+\pi/2-2\pi}^{\alpha+\pi/2}e^{-\imath(k\theta'-rs\sin(\theta'))}d\theta'\\
&=2\pi e^{\imath k(\alpha+\pi/2)}J_k(rs)
\end{aligned}\tag{57}$$

and, with the change of variables $\theta'' = \theta - \alpha + \pi/2$, we have

$$\begin{aligned}
\int_0^{2\pi}e^{-\imath(k\theta-rs\cos(\theta-\alpha))}d\theta&=e^{-\imath k(\alpha-\pi/2)}\int_{-\alpha+\pi/2}^{-\alpha+\pi/2+2\pi}e^{-\imath(k\theta''-rs\sin(\theta''))}d\theta''\\
&=2\pi e^{-\imath k(\alpha-\pi/2)}J_k(rs)
\end{aligned}\tag{58}$$

where J_k is the Bessel function. Thus, I_c^k becomes

$$\begin{aligned}
I_c^k(s,\alpha)&=\tfrac{2\pi}{2}\left(e^{\imath k(\alpha+\pi/2)}+e^{-\imath k(\alpha-\pi/2)}\right)\int_0^{\infty}r\tfrac{1}{\sqrt{r}}F_c^{k,0}(r)J_k(rs)\,dr-\tfrac{2\pi}{2\imath}\left(-e^{\imath k(\alpha+\pi/2)}+e^{-\imath k(\alpha-\pi/2)}\right)\int_0^{\infty}r\tfrac{1}{\sqrt{r}}F_s^{k,0}(r)J_k(rs)\,dr\\
&=\tfrac{2\pi}{2}\left(e^{\imath k(\alpha+\pi/2)}+e^{-\imath k(\alpha-\pi/2)}\right)H_c^k(s)+\tfrac{2\pi}{2\imath}\left(e^{\imath k(\alpha+\pi/2)}-e^{-\imath k(\alpha-\pi/2)}\right)H_s^k(s)
\end{aligned}\tag{59}$$

where $H_c^k(s)$ is the order k Hankel transform of $1/\sqrt{r}F_c^{k,0}(r)$, and $H_s^k(s)$ is the order k Hankel transform of $1/\sqrt{r}F_s^{k,0}(r)$.

Now note that

$$e^{-\imath k(\alpha-\pi/2)}=e^{-\imath k(\alpha+\pi/2)+\imath k\pi}=e^{\imath k\pi}e^{-\imath k(\alpha+\pi/2)}.\tag{60}$$

As a result,

$$I_c^k(s,\alpha)=\frac{2\pi}{2}\left(e^{\imath k(\alpha+\pi/2)}+e^{\imath k\pi}e^{-\imath k(\alpha+\pi/2)}\right)H_c^k(s)+\frac{2\pi}{2\imath}\left(e^{\imath k(\alpha+\pi/2)}-e^{\imath k\pi}e^{-\imath k(\alpha+\pi/2)}\right)H_s^k(s).\tag{61}$$

If k is even, then $e^{\imath k\pi} = -1$, so

$$\begin{aligned} I_c^k(s,\alpha) &= \tfrac{2\pi}{2}\left(e^{\imath k(\alpha+\pi/2)}+e^{-\imath k(\alpha+\pi/2)}\right) H_c^k(s)+\tfrac{2\pi}{2\imath}\left(e^{\imath k(\alpha+\pi/2)}-e^{-\imath k(\alpha+\pi/2)}\right) H_s^k(s) \\ &= 2\pi\cos\left(k\left(\alpha+\tfrac{\pi}{2}\right)\right) H_c^k(s)+2\pi\sin\left(k\left(\alpha+\tfrac{\pi}{2}\right)\right) H_s^k(s). \end{aligned} \quad (62)$$

This is a real-valued function because $F_c^{k,0}$ and $F_s^{k,0}$ are real when k is even.

If k is odd, then $e^{\imath k\pi} = -1$, so

$$\begin{aligned} I_c^k(s,\alpha) &= \tfrac{2\pi}{2}\left(e^{\imath k(\alpha+\pi/2)}-e^{-\imath k(\alpha+\pi/2)}\right) H_c^k(s)+\tfrac{2\pi}{2\imath}\left(e^{\imath k(\alpha+\pi/2)}+e^{-\imath k(\alpha+\pi/2)}\right) H_s^k(s) \\ &= 2\pi\imath\sin\left(k\left(\alpha+\tfrac{\pi}{2}\right)\right) H_c^k(s)-2\pi\imath\cos\left(k\left(\alpha+\tfrac{\pi}{2}\right)\right) H_s^k(s). \end{aligned} \quad (63)$$

While this appears imaginary, it is also real because $F_c^{k,0}$ and $F_s^{k,0}$, and therefore H_c^k and H_s^k, are imaginary when k is odd. A similar manipulation shows that, when k is even, then

$$I_s^k(s,\alpha) = -2\pi\sin\left(k\left(\alpha+\frac{\pi}{2}\right)\right) H_c^k(s)+2\pi\cos\left(k\left(\alpha+\frac{\pi}{2}\right)\right) H_s^k(s) \quad (64)$$

and, when k is odd, then

$$I_s^k(s,\alpha) = 2\pi\imath\cos\left(k\left(\alpha+\frac{\pi}{2}\right)\right) H_c^k(s)+2\pi\imath\sin\left(k\left(\alpha+\frac{\pi}{2}\right)\right) H_s^k(s). \quad (65)$$

Again, both (64) and (65) are real. However, note that (62) is a complex scalar multiple of (65). Scalar multiples of real values are unimportant, so we may combine these two into the single formula

$$I_1^k(s,\alpha) = \beta(k)\left(\cos\left(k\left(\alpha+\frac{\pi}{2}\right)\right) H_c^k(s)+\sin\left(k\left(\alpha+\frac{\pi}{2}\right)\right) H_s^k(s)\right) \quad (66)$$

where

$$\beta(k) = \frac{1}{2}(1+(-1)^k+\imath\,(1-(-1)^k)), \quad (67)$$

so that $\beta(k) = 1$ when k is even, and $\beta(k) = \imath$ when is odd. Similarly, (63) is a complex scalar multiple of (64), so we may combine these two into the single formula

$$I_2^k(s,\alpha) = \beta(k)\left(\sin\left(k\left(\alpha+\frac{\pi}{2}\right)\right) H_c^k(s)-\cos\left(k\left(\alpha+\frac{\pi}{2}\right)\right) H_s^k(s)\right). \quad (68)$$

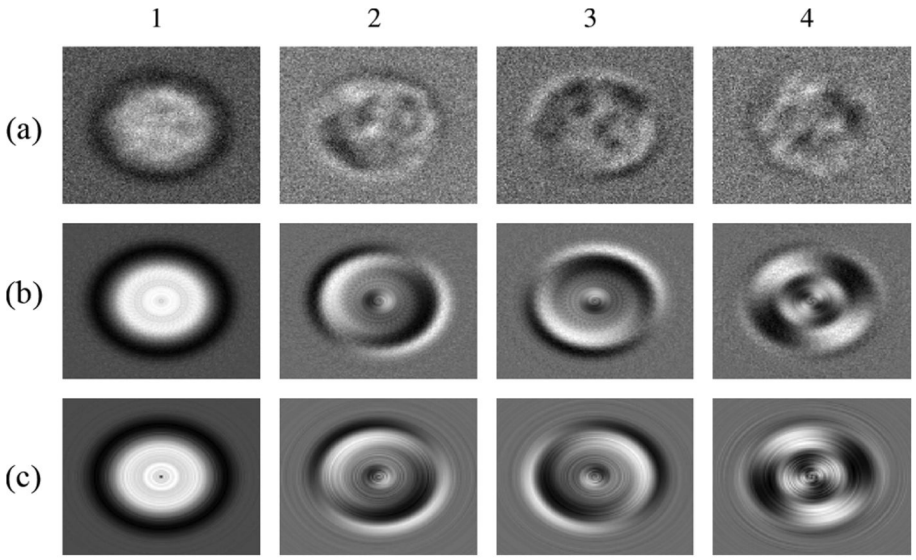

Fig. 1.
Number above each column represents which eigenimages is shown in that column. (a) Eigenimages produced by running SVD on the original set of 75 images. (b) Eigenimages produced by running SVD on the set of 2700 rotated images. (c) Eigenimages produced by our algorithm.

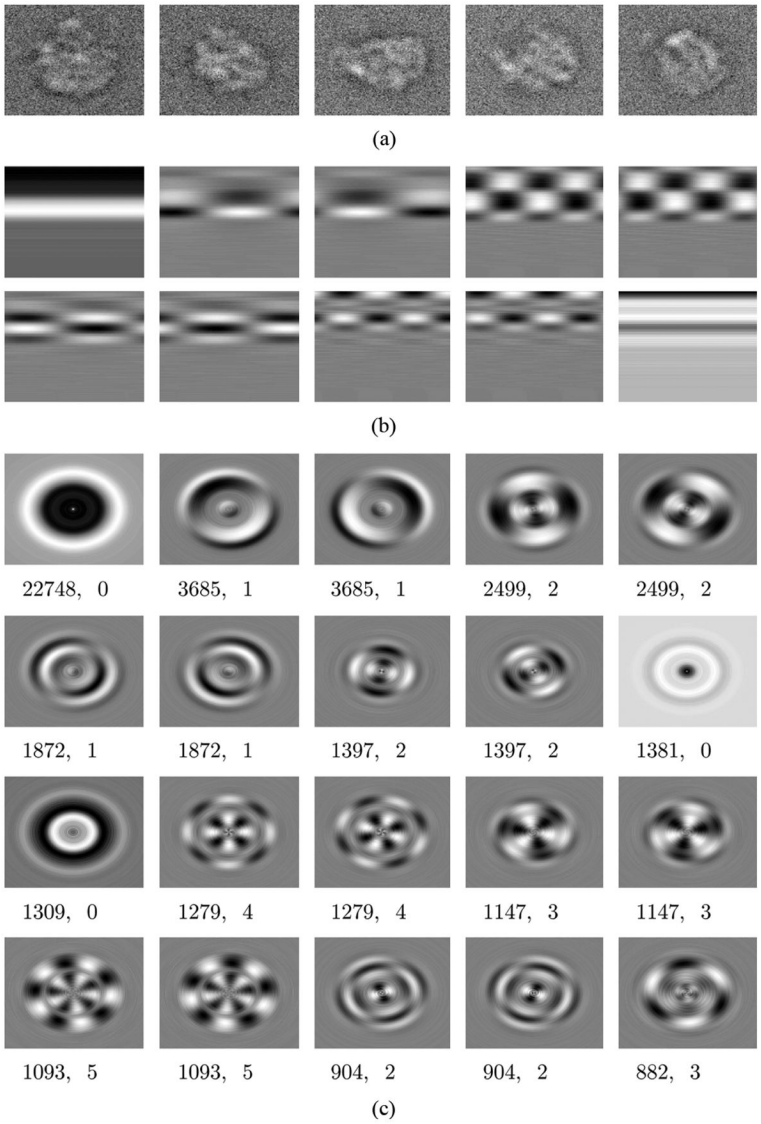

Fig. 2.

(a) Artificially generated projections of the *E. Coli* 50-s ribosomal subunit, with Gaussian noise added to create an SNR of 1/5. (b) The first ten" $\sqrt{|w|}$-filtered" radon eigenimages. (c) The first 20 true eigenimages. Below each true eigenimage is its eigenvalue μ and frequency k, respectively.

TABLE I

Comparison of Computational Complexities of Different PCA Algorithms

Algorithm	Computational Complexity
Ours	$O(PN_\theta N_r(N_r+\log(N_\theta))+N_\theta N_r^3)$
JZL	$O(P^2N_rN_\theta\log(N_rN_\theta)+N_\theta P^3)$
SVD	$\min\{N_rP^2N_\theta^3, PN_r^2N_\theta^3\}$
Eig. Decomp.	$PN_rN_\theta^2+(N_rN_\theta)^3$

TABLE II

Comparison of Running Times Under Increasing Number of Images P, in Seconds $N_r = 100$ and $N_\theta = 72$

P	This Paper	JZL	SVD	Eig. Decomp.
24	1.74	0.90	0.48	12.0
32	1.90	1.6	0.76	13.4
48	2.13	3.5	1.48	19.0
64	2.47	6.3	2.50	19.1
96	3.00	14.9	4.55	30.0
128	3.53	26.3	6.74	33.2
192	4.44	70.0	9.91	45.0
256	5.34	139	13.2	43.1
384	7.45	378	20.3	56.1
512	9.25	902	27.3	63.3
768	13.1	2796	43.7	73.0
1024	16.8	7926	63.1	80.2
1536	24.3	N/A	93.3	79.2
2048	31.8	N/A	137	96.8
3022	46.0	N/A	219	97.9
4096	61.9	N/A	317	122
6144	92.0	N/A	529	136
8192	122	N/A	703	151
12288	182	N/A	1116	160
16384	242	N/A	1739	197

TABLE III

Comparison of Running TIMES Under Increasing Number of Pixels Per Radial Line N_r, in Seconds. $P = 500$ and $N_\theta = 72$

N_r	This Paper	JZL	SVD	Eig. Decomp.
8	0.25	477	1.89	0.43
12	0.36	482	3.05	0.85
16	0.52	491	3.69	1.29
24	0.96	506	5.87	2.94
32	1.41	524	8.05	5.65
48	2.67	552	12.48	13.1
64	4.12	628	19.0	22.6
96	8.61	804	25.8	52.5
128	14.9	1027	35.1	87.8
192	34.7	1190	56.0	192
256	66.9	1520	68.9	417
384	185	2127	107	17916
512	445	3389	156	N/A
768	1751	6882	281	N/A
1024	5060	12149	386	N/A

TABLE IV

Comparison of Running Times Under Increasing Number of Rotations N_θ, in Seconds. $P = 500$ and $N_r = 100$

N_θ	This Paper	JZL	SVD	Eig. Decomp.
8	4.84	299	3.43	0.91
12	5.21	325	4.76	2.01
16	5.64	351	5.48	3.11
24	6.27	500	8.51	6.71
32	6.77	459	11.2	12.9
48	7.78	585	18.5	26.0
64	8.70	708	23.7	47.3
96	10.7	988	40.4	104
128	12.4	1302	51.5	178
192	16.4	1859	77.4	479
256	20.1	2705	108	4022
384	27.6	3873	170	N/A
512	37.3	6976	257	N/A
768	58.9	11133	400	N/A
1024	69.6	14706	560	N/A

TABLE V

Comparison of Grassmannian Distances Between SVD Eigensubspaces and Other Methods. (A)Images of Artificially Generated Projections of the *E.Coli* 50-s Ribosomal Subunit. (B) Images of White Noise

(A)				(B)			
Princ. Comps	**This Paper**	**JZL**	**Eig. Decomp.**	**Princ. Comps**	**This Paper**	**JZL**	**Eig. Decomp.**
1	$2.25 \cdot 10^{-15}$	$2.09 \cdot 10^{-15}$	$3.11 \cdot 10^{-15}$	1 & 2	$3.73 \cdot 10^{-13}$	$3.71 \cdot 10^{-13}$	$3.41 \cdot 10^{-13}$
2 & 3	$5.26 \cdot 10^{-15}$	$8.03 \cdot 10^{-15}$	$9.45 \cdot {}^{-15}$	3 & 4	$3.75 \cdot 10^{-13}$	$3.71 \cdot 10^{-13}$	$3.44 \cdot 10^{-13}$
4 &5	$6.94 \cdot 10^{-15}$	$7.17 \cdot 10^{-15}$	$7.83 \cdot 10^{-15}$	5 & 6	$9.25 \cdot 10^{-13}$	$9.27 \cdot 10^{-13}$	$1.05 \cdot 10^{-12}$
6 &7	$9.72 \cdot 10^{-15}$	$1.35 \cdot 10^{-14}$	$7.40 \cdot 10^{-15}$	7	$1.63 \cdot 10^{-12}$	$1.63 \cdot 10^{-12}$	$2.76 \cdot 10^{-12}$
8 &9	$9.45 \cdot 10^{-14}$	$9.36 \cdot 10^{-14}$	$1.87 \cdot 10^{-13}$	8 & 9	$1.60 \cdot 10^{-12}$	$1.60 \cdot 10^{-12}$	$2.74 \cdot 10^{-12}$

IEEE Trans Image Process. Author manuscript; available in PMC 2013 July 23.