Biological Imaging (2023), 3: e8
doi:10.1017/S2633903X23000089

RESEARCH ARTICLE

Three-dimensional alignment of density maps in cryo-electron microscopy

Yael Harpaz and Yoel Shkolnisky*

Department of Applied Mathematics, School of Mathematical Sciences, Tel-Aviv University, Tel-Aviv, Israel
*Corresponding author. E-mail: yoelsh@tauex.tau.ac.il

Received: 15 December 2022; **Revised:** 26 February 2023; **Accepted:** 28 February 2023

Keywords: Cryo-electron microscopy; global map alignment; synchronization; three-dimensional alignment

Abstract

A common task in cryo-electron microscopy data processing is to compare three-dimensional density maps of macromolecules. In this paper, we propose an algorithm for aligning three-dimensional density maps, which exploits common lines between projection images of the maps. The algorithm is fully automatic and handles rotations, reflections (handedness), and translations between the maps. In addition, the algorithm is applicable to any type of molecular symmetry without requiring any information regarding the symmetry of the maps. We evaluate our alignment algorithm on publicly available density maps, demonstrating its accuracy and efficiency. The algorithm is available at https://github.com/ShkolniskyLab/emalign.

Impact Statement
This paper describes a fast algorithm for the rotational and translational alignment of three-dimensional density maps. Such alignment is an essential step in cryo-electron microscopy data processing. The algorithm is available at https://github.com/ShkolniskyLab/emalign.

1. Introduction

Single particle cryo-electron microscopy (cryo-EM) is a method to determine the three-dimensional structure of biological macromolecules from their two-dimensional projection images acquired by an electron microscope[(1)]. In this method, a sample of identical copies of the investigated molecule is quickly frozen in a thin layer of ice, where each copy is frozen at an unknown random orientation. The frozen sample is imaged by an electron microscope, resulting in two-dimensional images, where each image is a tomographic projection of one of the randomly oriented copies in the ice layer. The goal of single particle cryo-EM is to determine the three-dimensional structure of the molecule from the acquired two-dimensional images. A common task in cryo-EM data processing is to compare two density maps of the same molecule. This is required, for example, to estimate the resolution of the maps, evaluating their Fourier shell correlation curve[(2)], or to analyze their different conformations. All these tasks require to first align two density maps, that is, to orient them in the same way in a common coordinate system. Due to the nature of the cryo-EM imaging process, the two density maps may differ not only in their three-dimensional orientation (i.e., their "rotation") but may also have different handedness (namely, reflected relative to each other) and may be centered differently with respect to a common coordinate system.

In this paper, we propose an algorithm for aligning two density maps, which is fully automatic and can handle rotations, translations, and reflections between the maps. The algorithm requires as an input only the two density maps. In particular, it does not assume knowledge of any other information such as the symmetry of the maps.

Formally, let $\phi_1 : \mathbb{R}^3 \to \mathbb{R}$ and $\phi_2 : \mathbb{R}^3 \to \mathbb{R}$ be two volumes such that

$$\phi_2(r) = \phi_1(Or - t), \tag{1}$$

where $r = (x,y,z)^T \in \mathbb{R}^3$, $O \in O(3)$ and $t = (\Delta x, \Delta y, \Delta z)^T \in \mathbb{R}^3$ ($O(3)$ is the group of all orthogonal transformations of the three-dimensional space, namely rotations and reflections). The alignment problem is to estimate O and t given ϕ_1 and ϕ_2. The matrix O is known as the orientation parameter, and the vector t as the translation parameter. In practice, we only get samples of ϕ_1 and ϕ_2, arranged as three-dimensional arrays of size $n \times n \times n$, where n is the resolution of sampling. In cryo-EM, ϕ_1 and ϕ_2 represent two reconstructions of the same underlying molecule that we would like to compare (such as two half maps from a refinement process). In principle, it is possible to approximate the solution to the alignment problem using exhaustive search, by generating a set of candidate pairs (O_i, t_i), where $O_i \in O(3)$ and $t_i \in \mathbb{R}^3$, and finding the pair which "best aligns" ϕ_1 to ϕ_2 in some chosen metric. The purpose of the alignment algorithm presented in this paper is to estimate the optimal alignment parameters in a fast and accurate way.

The paper is organized as follows. In Section 2, we review existing alignment algorithms. In Section 3, we give a high level simplified description of our algorithm. A detailed description is then given in Section 4. This description relies on a method for aligning a single projection image against a volume, a procedure which is described in Section 5. In Section 6, we discuss implementation considerations of the algorithm and analyze its complexity. An optional procedure for refining the estimated alignment parameters is described in Section 7. In Section 8, we demonstrate numerically the properties and performance of our algorithm. Finally, in Section 9, we discuss the properties and advantages of our algorithm.

2. Existing Methods

There exist several methods for three-dimensional alignment of molecular volumes. The Chimera software[3] offers a semi-automatic alignment method which requires the user to approximately align the volumes manually and then refines this alignment using an optimization procedure. This means that a sufficiently accurate initial approximation for the alignment is required. Achieving this initial approximate alignment manually naturally takes time and effort, yet it is crucial for the success of Chimera's alignment algorithm. The alignment procedure implemented by Chimera maximizes the correlation or overlap function between the two volumes by using a steepest descent optimization. The iterations of this optimization stop after reaching convergence or after 2,000 steps.

Another alignment method is the projection-based volume alignment (PBVA) algorithm[4]. This method aligns a target volume to a reference volume by aligning multiple projections of the reference volume to the target volume whose orientation is unknown. The PBVA algorithm is based on finding two identical projections, a projection P_1 from the reference volume and a projection P_2 from the target volume as follows. The reference volume is projected at some known Euler angles, resulting in a projection P_1, and the matching projection P_2 is found by maximizing the cross-correlation function between P_1 and a set of projections representing the possible projections of the target volume. The cross-correlation function is of five parameters—three Euler angles and two translation parameters in the plane of the projection P_2[5]. Finally, the rotation between the volumes is estimated from the relation between the Euler angles corresponding to the projections P_1 and P_2. After estimating the rotation between the volumes, the translation between them is found using projection images from the target volume. The translation between the volumes is estimated by least-squares regression using the two translation parameters of each projection from the target volume, where a minimum of two projections is required for calculating the three-dimensional translation vector. Using multiple projection images to estimate the translation between the volumes makes the alignment more robust.

https://doi.org/10.1017/S2633903X23000089 Published online by Cambridge University Press

The Xmipp software package[6] also offers a three-dimensional alignment algorithm. It is based on expanding the two volumes using spherical harmonics followed by computing the cross-correlation function between the two spherical harmonics expansions representing the volumes[7]. The process of expanding a volume into spherical harmonics is called the spherical Fourier transform (SFT) of the volume, where like the fast Fourier transform (FFT) algorithm, there exists an efficient algorithm for calculating the SFT[7]. The process of calculating the cross-correlation function between the two spherical harmonics expansions of the volumes and estimating the rotation between the two volumes is implemented by a fast rotational matching (FRM) algorithm[8]. After estimating the rotation between the two volumes, the translation between them is found by using the phase correlation algorithm[9].

Finally, the EMAN2 software package[10] offers 2 three-dimensional alignment algorithms. In the first algorithm (implemented by the program `e2align3d`, now mostly obsolete), the rotation between the volumes is estimated using an exhaustive search for the three Euler angles of the rotation. First, the algorithm generates a set of candidate Euler angles with large angular increments. Then, the algorithm iteratively decreases the angular increments in the set of candidates in order to refine the resolution of the angular search[11]. A much faster tree-based algorithm is implemented in the program `e2proc3d`. This method performs three-dimensional rotational and translational alignment using a hierarchical method with gradually decreasing downsampling in Fourier space. In Section 8, we compare our algorithm to this latter algorithm, as well as to the FRM algorithm implemented by Xmipp.

3. Outline of the Approach

We are given two volumes ϕ_1 and ϕ_2 satisfying (1). For simplicity, we assume for now that the volumes have no symmetry and are related by rotation only (no translation nor reflection). We generate a projection image from ϕ_2, denoted P, corresponding to an orientation given by a rotation matrix R. Since ϕ_1 and ϕ_2 are the same volume up to rotation, we can orient P relative to ϕ_1, that is, we can find the rotation $\tilde{R}$ such that projecting ϕ_1 in the orientation determined by $\tilde{R}$ results in the image P. As we show below, it holds that $\tilde{R} = OR$, where O is the transformation from (1). Since R and $\tilde{R}$ are known, we can estimate O as $O = \tilde{R}R^T$.

In practice, it may be that $\tilde{R}$ is not determined uniquely by P, as for example, a volume may have two very similar views even if it is not symmetric. Moreover, the volumes to align are discretized and sometimes noisy, which introduces inaccuracies into the estimation of O. Thus, to estimate O more robustly, instead of using a single image P, we generate from ϕ_2 multiple images $P_1, \ldots, P_N$ with orientations $R_1, \ldots, R_N$, align each P_i to ϕ_1 as above, resulting in estimates for O given by $O_i = \tilde{R}_i R_i^T$, and then estimate O from all O_i simultaneously by solving.

$$O = \underset{R}{\operatorname{argmin}} \sum_{i=1}^{N} \|O_i - R\|_F^2,$$

where $\|\cdot\|_F$ is the Frobenius matrix norm. In Section 4, we give an explicit solution for the latter optimization problem.

The key of the above procedure is estimating the orientation of a projection image P of ϕ_2 in the coordinate system of ϕ_1. This is done by inspecting a large enough set of candidate rotations and finding the rotation $\tilde{R}$ for which the induced common lines between P (when assuming its orientation is $\tilde{R}$) and a set of projections generated from ϕ_1 best agree. As inspecting each candidate rotation involves only one-dimensional operations (even if the input volumes are centered differently), it is very fast and highly parallelizable. Thus, this somewhat brute force approach is applicable to very large sets of candidate rotations (several thousands, for accurate alignment) and still results in a fast algorithm. We discuss below the complexity and advantages of this approach. We summarize the outline of our approach in Figure 1 and describe it in detail in Sections 4 and 5.

In the above approach, we assume that O is a rotation. However, ϕ_1 and ϕ_2 may have a different handedness, and so O may include a reflection. The above approach can obviously be used to resolve the handedness by aligning ϕ_2 to ϕ_1 and to a reflected copy of ϕ_1, and determining whether a reflection is

https://doi.org/10.1017/S2633903X23000089 Published online by Cambridge University Press

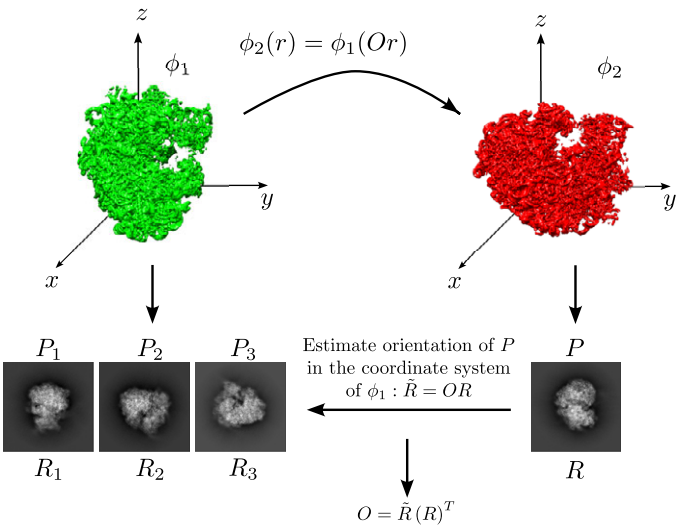


Figure 1. *Outline of the algorithm.*

needed using some quality score of the alignment parameters (e.g., the correlation between the aligned volumes). However, as we show below, in our method, there is no need to actually align ϕ_2 to a reflected copy of ϕ_1, saving roughly half of the computations (those required to actually align ϕ_2 to a reflected copy of ϕ_1), as explained in Section 4.

We next explain in detail the various steps of our algorithm, including handling translations, reflections, and symmetry in the volumes.

4. Estimating the Alignment Parameters

Consider two volumes ϕ_1 and ϕ_2, where one volume is a rotated copy of the other (assuming for now no reflection nor translation between the volumes), namely (see (1))

$$\phi_2(r) = \phi_1(Or), \quad r = (x,y,z)^T \in \mathbb{R}^3, \tag{2}$$

where O is an unknown rotation matrix. Our goal is to find an estimate for O.

In case where ϕ_1 and ϕ_2 exhibit symmetry, the solution for O is not unique. To be concrete, we denote by $SO(3)$ the group of all 3×3 rotation matrices. A group $G \subseteq SO(3)$ is a symmetry group of a volume ϕ, if for all $g \in G$ it holds that

$$\phi(r) = \phi(gr), \quad r = (x,y,z)^T \in \mathbb{R}^3. \tag{3}$$

In other words, a symmetry group of a volume is a group of rotations under which the volume is invariant (see[12] for more details). If we denote the symmetry group of ϕ_1 by $G_1 \subseteq SO(3)$ and define $r' = Or$, then, from (2) and (3), we get for any symmetry element $g \in G_1$

$$\phi_2(r) = \phi_1(Or) = \phi_1(r') = \phi_1(gr') = \phi_1(gOr). \tag{4}$$

Comparing the latter with (2), we conclude that the solution for O is not unique, and we thus replace the goal of finding O by finding any gO for some arbitrary element $g \in G_1$ of the symmetry group.

Note that we assume that O is a rotation, namely that ϕ_1 and ϕ_2 are related by rotation without reflection. The case where O is a reflection will be considered below. Let P be a projection image generated from ϕ_2 using a rotation R, that is,

$$P(x,y) = \int_{-\infty}^{\infty} \phi_2(Rr)dz = \int_{-\infty}^{\infty} \phi_2(xR^{(1)} + yR^{(2)} + zR^{(3)})dz, \tag{5}$$

where $R^{(1)}$, $R^{(2)}$, $R^{(3)}$ are the columns of the matrix R and $r = (x,y,z)^T$. From (2), we have that

https://doi.org/10.1017/S2633903X23000089 Published online by Cambridge University Press

$$\phi_2(Rr) = \phi_1(ORr). \tag{6}$$

Thus, using (5), we have

$$P(x,y) = \int_{-\infty}^{\infty} \phi_2(Rr)dz = \int_{-\infty}^{\infty} \phi_1(ORr)dz. \tag{7}$$

Equation (7) implies that if P has orientation R with respect to ϕ_2, then it has orientation OR with respect to ϕ_1. In Section 5, we describe how to estimate OR given P and ϕ_1, namely how to estimate a rotation $\tilde{R}$ that satisfies $\tilde{R} = OR$. If the volume ϕ_1 is symmetric with symmetry group G_1, then (as shown above) the rotation OR is equivalent to the rotation gOR for any $g \in G_1$, and moreover, the two rotations cannot be distinguished. Thus, we conclude that

$$\tilde{R} = gOR$$

for some unknown $g \in G_1$. Using the latter equation, we can estimate O as

$$O = g^T \tilde{R} R^T. \tag{8}$$

Note that in the latter equation R is known, $\tilde{R}$ can be estimated using the algorithm in Section 5 below, and g can be arbitrary. Thus, (8) provides a means for estimating O.

However, to estimate O more robustly, we use multiple projections generated from ϕ_2. Let $R_1, \ldots, R_N$ be random rotations, and let $P_1, \ldots, P_N$ be the corresponding projections generated from ϕ_2 according to (5). Using the procedure described above, we estimate for each P_i a rotation $\tilde{R}_i$ that satisfies $\tilde{R}_i = g_i O R_i$ for some unknown $g_i \in G_1$. Thus, as in (8), we can estimate O using any $i \in \{1, \ldots, N\}$ by

$$O = g_i^T \tilde{R}_i R_i^T. \tag{9}$$

Contrary to (8), if we want the right-hand side of (9) to result in the same O for all $i = 1, \ldots, N$, then g_i cannot be arbitrary. In order to estimate O, we therefore need to find g_i, $i = 1, \ldots, N$ and combine all estimates for O given in (9) into a single estimate.

To that end, define

$$X_i = R_i \tilde{R}_i^T, \quad i = 1, \ldots, N, \tag{10}$$

and look at the matrix H of size $3N \times 3N$ whose (i,j) block of size 3×3 is given by (see (9) and (10))

$$\begin{aligned} H_{ij} &= X_i{}^T X_j = \tilde{R}_i R_i^T R_j \tilde{R}_j^T \\ &= \tilde{R}_i \left(g_i^T \tilde{R}_i\right)^T O O^T \left(g_j^T \tilde{R}_j\right) \tilde{R}_j^T = g_i g_j^T. \end{aligned} \tag{11}$$

By a direct calculation, we get that the matrix of size $3N \times 3$

$$\tilde{g} = \begin{pmatrix} g_1 W \\ \vdots \\ g_N W \end{pmatrix}, \tag{12}$$

where W is an arbitrary 3×3 orthogonal matrix (i.e., $WW^T = W^T W = I$), satisfies

$$H\tilde{g} = N\tilde{g}. \tag{13}$$

Equation (11) also shows that the matrix H is of rank 3, which together with (13) implies that $\tilde{g}$ can be calculated by arranging the three leading eigenvectors v_1, v_2, v_3 of H in a matrix

$$V = \begin{pmatrix} | & | & | \\ v_1 & v_2 & v_3 \\ | & | & | \end{pmatrix}_{3N \times 3} = \begin{pmatrix} V_1 \\ \vdots \\ V_N \end{pmatrix}, \tag{14}$$

https://doi.org/10.1017/S2633903X23000089 Published online by Cambridge University Press

whose 3×3 blocks $V_1,\ldots,V_N$ are $g_1W,\ldots,g_NW$, for some unknown arbitrary W (see[13] for a detailed derivation). In practice, at this point, we replace each g_iW by its closest orthogonal transformation, as described in[14], to improve its accuracy in the presence of noise and discretization errors.

Next, in order to extract an estimate for $g_1,\ldots,g_N$ from (14) (i.e., to eliminate W from the estimates in $\tilde{g}$ given by (12)), we multiply each g_iW by $(g_1W)^T$, resulting in

$$g_{est}=\begin{pmatrix} g_{est_1}\\ \vdots\\ g_{est_N}\end{pmatrix}=\begin{pmatrix} g_1WW^Tg_1^T\\ \vdots\\ g_NWW^Tg_1^T\end{pmatrix}=\begin{pmatrix} g_1{g_1}^T\\ \vdots\\ g_N{g_1}^T\end{pmatrix}. \tag{15}$$

Thus, each g_{est_i} is a rotation, even if W is not. We define $O_i=g_{est_i}^T\tilde{R}_iR_i^T$, and using (9), we get for $i=1,\ldots,N$

$$O_i=g_{est_i}^T\tilde{R}_iR_i^T=g_1g_i^T\tilde{R}_iR_i^T=g_1O. \tag{16}$$

Thus, we have N estimates for g_1O. Equation (4) states that $\phi_2(r)=\phi_1(Or)=\phi_1(gOr)$ for any symmetry element $g\in G_1$. Therefore, estimating g_1O is equivalent to estimating O. In order to estimate g_1O simultaneously from all O_i, $i=1,\ldots,N$, we search for the rotation $O_{est}^{(1)}$ (the superscript will be explained shortly) that satisfies

$$O_{est}^{(1)}=\underset{R}{\operatorname{argmin}}\sum_{i=1}^{N}\|O_i-R\|_F^2. \tag{17}$$

In other words, $O_{est}^{(1)}$ is the "closest" to all the estimated rotations O_i in the least squares sense. To solve (17), let $\tilde{O}$ be the 3×3 matrix

$$\tilde{O}=\frac{1}{N}\sum_{i=1}^{N}O_i. \tag{18}$$

In[15], it is proven that the solution to the optimization problem in (17) is

$$O_{est}^{(1)}=\tilde{U}\tilde{V}^T, \tag{19}$$

where $\tilde{O}=\tilde{U}\tilde{\Sigma}\tilde{V}^T$ is the singular value decomposition (SVD) of $\tilde{O}$. The algorithm for estimating $O_{est}^{(1)}$ given ϕ_1 and ϕ_2, as described above, is presented in Algorithm 1.

Algorithm 1 Estimating $O_{est}^{(1)}$.

Input: Volumes ϕ_1,ϕ_2.

1: Generate random rotations $\{R_i\}_{i=1}^N$

2: Generate from ϕ_2 projections $\{P_i\}_{i=1}^N$ corresponding to the rotations $\{R_i\}_{i=1}^N$ ▷ Eq. (5)

3: Apply Algorithm 2 to each P_i and ϕ_1. Denote the resulting rotations by $\{\tilde{R}_i\}_{i=1}^N$

4: **for** $i=1$ to N **do**

5: Calculate $X_i=R_i\tilde{R}_i^T$ ▷Eq. (10)

6: Construct the matrix $3N\times 3N$

$$H=\begin{pmatrix} I_{3\times 3} & X_1^TX_2 & \cdots & X_1^TX_N\\ X_2^TX_1 & I_{3\times 3} & \cdots & X_2^TX_N\\ \vdots & \vdots & \ddots & \\ X_N^TX_1 & X_N^TX_2 & \cdots & I_{3\times 3}\end{pmatrix}$$

7: Find the three leading eigenvectors v_1,v_2,v_3 of H

8: Set

https://doi.org/10.1017/S2633903X23000089 Published online by Cambridge University Press

$$V = \begin{pmatrix} | & | & | \\ v_1 & v_2 & v_3 \\ | & | & | \end{pmatrix}_{3N\times 3} = \begin{pmatrix} V_1 \\ \vdots \\ V_N \end{pmatrix}$$

▷ Eq. (14)

9: Compute

$$g_{est} = \begin{pmatrix} g_{est_1} \\ \vdots \\ g_{est_N} \end{pmatrix} = \begin{pmatrix} V_1 V_1^T \\ \vdots \\ V_N V_1^T \end{pmatrix}$$

▷ Eq. (15)

10: **for** $i = 1$ to N **do**

11: Calculate $O_i = g_{est_i}^T \tilde{R}_i R_i^T$ ▷ Eq. (16)

12: Calculate $\tilde{O} = \frac{1}{N}\sum_{i=1}^{N} O_i$ ▷ Eq. (18)

13: Calculate $O_{est}^{(1)} = \tilde{U}\tilde{V}^T$, where $\tilde{O} = \tilde{U}\tilde{\Sigma}\tilde{V}^T$ is the SVD of $\tilde{O}$ ▷ Eq. (19)

Output: $O_{est}^{(1)}$ ▷ Estimated rotation

To handle the case where ϕ_1 and ϕ_2 have a different handedness (namely, related by reflection), we can of course apply Algorithm 1 to ϕ_2 and a reflected copy of ϕ_1. However, this would roughly double the runtime of the estimation process, as the most time-consuming step in Algorithm 1 is step 3, whose complexity is $O(n^3 \log n)$ operations for a volume of size $n \times n \times n$ voxels (see Section 5).

Alternatively, it is possible to augment the above algorithm to handle reflections without doubling its runtime. In the case where there is a reflection between ϕ_1 and ϕ_2, we need to replace the relation in (2) by the relation

$$\phi_2(r) = \phi_1(OJr), \quad J = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix}. \tag{20}$$

Note that J in (20) is a reflection and that O is a rotation. Repeating the above derivation starting from (20) shows that to estimate O in this case, we can use the same R_i used above and the same estimates $\tilde{R}_i$ obtained above (steps 1 and 3 of Algorithm 1), but this time we get that $O_i = g_i^T \tilde{R}_i J R_i^T J$ (compare with (9)). Then, we set $X_i = J R_i J \tilde{R}_i^T$ (compare with (10)) and proceed as above, resulting in an estimate $O_{est}^{(2)}$ (compare with (17)), which corresponds to the optimal alignment parameters if ϕ_1 and ϕ_2 have opposite handedness. Once we have the two estimates $O_{est}^{(1)}$ and $O_{est}^{(2)}$ for the alignment parameters between ϕ_1 and ϕ_2 (without and with reflection), we estimate the translation corresponding to each of $O_{est}^{(1)}$ and $O_{est}^{(2)}$ using phase correlation[9] (see Appendix C for details). This results in two sets of alignment parameters (rotation+translation). We then apply both sets of parameters to ϕ_2 to align it with ϕ_1 and pick the parameters for which ϕ_2 after alignment has higher correlation with ϕ_1. We denote the estimated parameters by (O_{est}, t_{est}).

5. Projection Alignment

It remains to show how to implement step 3 of Algorithm 1, that is, how to find the orientation of a projection P of ϕ_2 with respect to the coordinate system of ϕ_1. Mathematically, we would like to solve the equation

$$P(x,y) = \int_{-\infty}^{\infty} \phi_1(Rr)dz, \quad r = (x,y,z)^T \in \mathbb{R}^3 \tag{21}$$

https://doi.org/10.1017/S2633903X23000089 Published online by Cambridge University Press

for the unknown rotation R. A brute force approach of testing many candidate rotations in search for the R that (best) satisfies (21) is prohibitively expensive, as it requires to compute a projection of ϕ_1 for each candidate rotation (this is essentially projection matching). We therefore take a different approach, whose cost for inspecting each candidate rotation is much lower (in fact requires $O(n)$ operations to test each candidate rotation for a volume ϕ_1 discretized into an array of size $n \times n \times n$).

The idea is to generate several projection images from ϕ_1, and then, for each candidate rotation, to check the agreement of the common lines between P and the projections of ϕ_1, assuming the orientation of P is given by the candidate rotation. We estimate the rotation corresponding to P as the candidate rotation that results in the best agreement. We next formalize this method and then analyze its complexity.

We start by considering the case where there is no translation between P and ϕ_1, namely P and ϕ_1 satisfy (21), and our goal is to estimate R given P and ϕ_1. We generate N projection images from ϕ_1 (N is typically small, see Section 8), denoted $P_1^{(a)},\ldots,P_N^{(a)}$, with rotations $R_1^{(a)},\ldots,R_N^{(a)}$ chosen uniformly at random (note that we deliberately reuse the notation N used in Section 4, as explained below). We generate a set of candidate rotations S, over which we will search for the solution R of (21). The set S consists of a large number of approximately equally spaced rotations. See Appendix B for a detailed description of the construction of S.

We will assume for each candidate rotation $Q \in S$ that P was generated using the rotation Q (i.e., we assume that R in (21) is equal to Q), compute the mean correlation of the common lines between P and $P_1^{(a)},\ldots,P_N^{(a)}$, and choose as an estimate for R the rotation Q for which the mean correlation is highest. Specifically, for each $Q \in S$ and $R_i^{(a)}$, $i=1,\ldots,N$, we compute the direction of the common line between P and $P_i^{(a)}$, given by the angles α_i in P and β_i in $P_i^{(a)}$, as explained in Appendix A. The common line property[16] states that if $Q=R$ then

$$\widehat{P}(\xi\cos\alpha_i,\xi\sin\alpha_i)=\widehat{P}_i^{(a)}(\xi\cos\beta_i,\xi\sin\beta_i),\quad \xi\in\mathbb{R},$$

where $\widehat{P}$ and $\widehat{P}_i^{(a)}$ are the Fourier transforms of P and $P_i^{(a)}$, respectively (see Appendix A for a review of common lines and their properties). We thus define

$$f_i(Q,\xi)=\widehat{P}(\xi\cos\alpha_i,\xi\sin\alpha_i),$$
$$g_i(Q,\xi)=\widehat{P}_i^{(a)}(\xi\cos\beta_i,\xi\sin\beta_i),$$

and the cost function

$$\rho(Q)=\frac{1}{N}\Re\sum_{i=1}^{N}\frac{\int_0^\infty \bar{f}_i(Q,\xi)g_i(Q,\xi)d\xi}{\|f_i\|_{L^2}\|g_i\|_{L^2}},\tag{22}$$

where $\bar{f}_i$ denotes the complex conjugate of f_i. In other words, $\rho(Q)$ measures how well the common lines induced by Q between P and $P_1^{(a)},\ldots,P_N^{(a)}$ agree. We then set our estimate for R to be

$$R_{est}=\operatorname*{argmax}_{Q\in S}\rho(Q).$$

We explore the appropriate value for N in Section 8.

We now extend the above scheme to the case where P is not centered with respect to ϕ_1, namely P is given by

$$P(x-\Delta x,y-\Delta y)=\int_{-\infty}^{\infty}\phi_1(Rr)dz,\quad r=(x,y,z)^T\in\mathbb{R}^3,\tag{23}$$

for an unknown rotation R and an unknown translation $(\Delta x,\Delta y)$. The idea for estimating R is the same as before, except that the calculation of the common lines should take into account the unknown translation, as we describe next.

We denote the unshifted version of P by $\tilde{P}$, which is given by

https://doi.org/10.1017/S2633903X23000089 Published online by Cambridge University Press

$$\tilde{P}(x,y) = \int_{-\infty}^{\infty} \phi_1(Rr)dz, \quad r=(x,y,z)^T \in \mathbb{R}^3 \tag{24}$$

(this is exactly (21), but we repeat it to clearly set up the notation). Then,

$$P(x,y) = \tilde{P}(x+\Delta x, y+\Delta y).$$

Taking the Fourier transform of both sides of the latter equation, we get that[(1)]

$$\widehat{P}(\omega_x,\omega_y) = \widehat{\tilde{P}}(\omega_x,\omega_y)e^{i(\omega_x\Delta x+\omega_y\Delta y)}. \tag{25}$$

Suppose that the common line between $\tilde{P}$ and $P_i^{(a)}$ is given by the angles α_i in $\tilde{P}$ and β_i in $P_i^{(a)}$ (see Appendix A). By definition of the common line, it holds that

$$\widehat{\tilde{P}}(\xi\cos\alpha_i,\xi\sin\alpha_i) = \widehat{P}_i^{(a)}(\xi\cos\beta_i,\xi\sin\beta_i).$$

Using (25), we get that

$$\widehat{P}(\xi\cos\alpha_i,\xi\sin\alpha_i)e^{-i\xi\Delta\xi} = \widehat{P}_i^{(a)}(\xi\cos\beta_i,\xi\sin\beta_i),$$

where $\Delta\xi = \Delta x\cos\alpha_i + \Delta y\sin\alpha_i$ is the one-dimensional shift between the projections along their common line. We assume that this one-dimensional shift is bounded by some number d.

Thus, we need to modify our cost function (22) to take into account also the unknown (one-dimensional) phase $e^{-i\xi\Delta\xi}$. We therefore define (with a slight abuse of notation in reusing the previous notation for the cost function)

$$\begin{aligned} f_i(Q,\Delta\xi,\xi) &= \widehat{P}(\xi\cos\alpha_i,\xi\sin\alpha_i)e^{-i\xi\Delta\xi}, \\ g_i(Q,\xi) &= \widehat{P}_i^{(a)}(\xi\cos\beta_i,\xi\sin\beta_i) \end{aligned}$$

and the cost function

$$\rho(Q,\Delta\xi) = \frac{1}{N}\Re\sum_{i=1}^{N}\frac{\int_0^{\infty}\bar{f}_i(Q,\Delta\xi,\xi)g_i(Q,\xi)d\xi}{\|f_i\|_{L^2}\|g_i\|_{L^2}}, \tag{26}$$

and set our estimate for the solution R of (23) to be

$$R_{est} = \underset{Q\in S,\Delta\xi\in[-d,d]}{\operatorname{argmax}} \rho(Q,\Delta\xi). \tag{27}$$

The formula for the angles α_i and β_i of the common line between P and $P_i^{(a)}$ induced by the rotations $Q\in S$ and R_i is given in Appendix A. Note that at this point we are only interested in R_{est} and not in the translation $(\Delta x,\Delta y)$ in P, as the relative translation between ϕ_1 and ϕ_2 is efficiently determined using phase correlation (see[(9)] and Appendix C) once we have determined their relative rotation. The algorithm for solving equation (23) is summarized in Algorithm 2.

Algorithm 2 Projection alignment.

Input: Projection P and volume ϕ_1 satisfying (23).
1: Generate random rotations $R_1,\ldots,R_N$
2: Generate from ϕ_1 projections $P_1^{(a)},\ldots,P_N^{(a)}$ corresponding to the rotations $R_1,\ldots,R_N$
3: Generate candidate rotations S ▷ Appendix B
4: Compute

$$R_{est} = \underset{Q\in S,\Delta\xi\in[-d,d]}{\operatorname{argmax}} \rho(Q,\Delta\xi).$$

▷ Eqs. 26 and (27)
Output: R_{est} ▷ Estimated rotation

https://doi.org/10.1017/S2633903X23000089 Published online by Cambridge University Press

As mentioned above, we use the same N in Sections 4 and 5. While, in principle, the number of projections generated from ϕ_2 in Section 4 can be different from the number of projections generated from ϕ_1 in Section 5, due to the symmetric role of ϕ_1 and ϕ_2 in the alignment problem, there is no reason to consider different values.

6. Implementation and Complexity Analysis

Algorithms 1 and 2 are formulated in the continuous domain. Obviously, to implement them, we must explain how to apply them to volumes ϕ_1 and ϕ_2 given as three-dimensional arrays of size $n \times n \times n$. We now explain how to discretize each of the steps of Algorithms 4 and 5 and analyze their complexity. For simplicity, we use for the discrete quantities the same notation we have used for the continuous ones.

The only step in Algorithm 1 that needs to be discretized is step 2. This step is accurately discretized based on the Fourier projection slice theorem (A.1) using a non-equally spaced FFT[17,18], whose complexity is $O(n^3 \log n)$ (for a fixed prescribed accuracy). The result of this step is a discrete projection image P given as a two-dimensional array of size $n \times n$ pixels. The remaining steps of Algorithm 4 are already discrete, and since the value of N is small compared to n, their complexity is negligible.

We next analyze Algorithm 2. The input to this algorithm is a projection image P of size $n \times n$ pixels, and a volume ϕ_1 of size $n \times n \times n$ voxels. The algorithm also uses the parameter N, but since it is a small constant, we ignore it in our complexity analysis. Step 1 of Algorithm 2 requires a constant number of operations. Step 2 is accurately implemented using a non-equally spaced FFT[17,18], whose complexity is $O(n^3 \log n)$ (for a fixed prescribed accuracy). Step 3 is independent of the input volume, and moreover, the set S can be precomputed and stored. To implement step 4, we first discretize the interval of one-dimensional shifts $[-d, d]$ in fixed steps of Δd pixels (say, 1 pixel). Specifically, we use the following shift candidates for the optimization in step 4:

$$\Delta\xi \in \{-d + k\Delta d \,|\, k = 0, \ldots, \lfloor 2d/\Delta d \rfloor\}.$$

Then, for each $Q \in S$, we compute the angles α_i and β_i (see Appendix A) and evaluate (26) for the pair $(Q, \Delta\xi)$ by replacing the integral with a sum. If we store the polar Fourier transforms of all involved projection images P and $P_1^{(a)}, \ldots, P_N^{(a)}$ (computed using the non-equally spaced FFT[17,18]), each such evaluation amounts to accessing the rays in the polar Fourier transform corresponding to the angles α_i and β_i, namely $O(n)$ operations. Thus, the total number of operations required to implement step 4 of Algorithm 2 is $|S| \times (\lfloor 2d/\Delta d \rfloor + 1) \times n$ ($|S|$ is the number of elements in the set S). Of course, all $|S| \times (\lfloor 2d/\Delta d \rfloor + 1)$ evaluations are independent and can be computed in parallel. Thus, the total complexity of Algorithm 2 is $O(n^3 \log n)$ operations for step 2 and $O(n)$ operations for testing each pair $(Q, \Delta\xi)$ in step 4. Therefore, since the optimization in step 4 is very fast, it is practical to test even a very large set of candidate rotations S.

Finally, we note that in practice, to further speed up the algorithm, we first downsample the input volumes to size n_{ds}, align the two downsampled volumes, and apply the estimated alignment parameters to the original volumes. We demonstrate in Section 8 that this approach still results in a highly accurate alignment.

To understand the theoretical advantage of the above approach, we compare it to a brute force approach. In the brute force approach, we 1) scan over a large set of rotations and three-dimensional translations, 2) for each pair of a rotation and a translation, we transform one of the volumes according to this pair of parameters, and 3) choose the pair for which the correlation between the volumes after the transformation is maximal. Testing each pair of candidate parameters requires $O(n^3)$ operations (for rotating and translating one of the volumes, and for computing correlation), which amounts to a total of $O\left(n^3 \times |S| \times (2d/\Delta d)^3\right)$ operations. In other words, testing each candidate rotation and translation is way more expensive than in our proposed method. In our approach, the expensive operation of complexity $O(n^3 \log n)$ needs to be executed only once per each pair of inputs P and ϕ_1. Moreover, in our approach, the

https://doi.org/10.1017/S2633903X23000089 Published online by Cambridge University Press

search over shifts is one-dimensional as opposed to the three-dimensional search required in the brute force approach.

7. Parameters' Refinement

In this section, we describe an optional refinement procedure for improving the accuracy of the estimated parameters O_{est} and t_{est} obtained using the algorithm of Section 4.

We define the vector $\Theta = (\psi, \vartheta, \varphi, \Delta_x, \Delta_y, \Delta_z)$ consisting of the six parameters required to describe the transformation between two volumes—three Euler angles $(\psi, \vartheta, \varphi)$ describing their relative rotation and three parameters $(\Delta x, \Delta y, \Delta z)$ describing their relative translation. We define the operator $T_\Theta(\phi)$, which applies the transformation parameters Θ to the volume ϕ (i.e., T_Θ first rotates the volume and then translates it, according to the parameters in Θ). Next, for given volumes ϕ_1 and ϕ_2, we denote their correlation by $\rho(\phi_1, \phi_2)$. We are reusing the notation ρ from Section 5, since all occurrences of ρ in this paper correspond to a correlation coefficient whose evaluation formula is clear from its arguments. Finally, we define the objective function

$$c(\theta) = 1 - \rho(T_\Theta(\phi_1), \phi_2), \tag{28}$$

which vanishes for the parameters Θ that align ϕ_1 with ϕ_2.

To refine O_{est} and t_{est} of Section 4, we simply apply the BFGS algorithm[19] to the objective function (28), with an initialization given by O_{est} and t_{est}.

8. Results

The alignment algorithm (with and without the optional refinement described in Section 7) was implemented in Python and is available online[1], including the code that generates the figures of this section. A Matlab version of the algorithm is available as part of the ASPIRE software package[20].

As the algorithm uses two parameters—the downsampling n_{ds} (see Section 6) and the number of reference projections N (see Section 4)—we first examine how to appropriately set their values. Then, we examine the advantage of the refinement procedure proposed in Section 7. To show the benefits of our algorithm in practice, we then compare its performance to that of two other alignment algorithms—the alignment algorithm from the EMAN2 software package[10] (implemented in the program `e2proc3d`) and the FRM algorithm implemented in the Xmipp software package[6]. Finally, we examine the performance of the three algorithms using noisy input volumes.

We tested our algorithm on volumes from the electron microscopy data bank (EMDB)[21] with different types of symmetries, whose properties are described in Table 1. All tests were executed on a dual Intel Xeon E5-2683 CPU (32 cores in total), with 768 GB of RAM running Linux. The memory required by the algorithm is of the order of the size of the input volumes. We used 15,236 candidate rotations in Algorithm 2 (the size of the set S), generated as described in Appendix B. This set of candidates is roughly equally spaced in the set of rotations $SO(3)$. While it is difficult to characterize the resolution of this set in terms of the resolution of each of the Euler angles, a rough calculation suggests that the resolution in each of the Euler angles is smaller than 5 degrees. We do not use rotations generated by a regular grid of Euler angles, as such a grid is less efficient than our grid, due to the nonuniform rotations generated by a regular grid of Euler angles. For example, discretizing each of the Euler angles to 5 degrees would result in 186,624 rotations, more than an order of magnitude larger than the number of rotations we use.

For each test, we generate a pair of volumes ϕ_1 and ϕ_2 related by a rotation matrix O and a translation vector $t \in \mathbb{R}^3$. The translation is chosen at random with magnitude up to 10% of the size of the volume. We denote the alignment parameters estimated by our algorithm by O_{est} and t_{est}. We evaluate the accuracy of our algorithm by calculating the difference between the rotations O and O_{est}. To that end, we first note that

[1] https://github.com/ShkolniskyLab/emalign

https://doi.org/10.1017/S2633903X23000089 Published online by Cambridge University Press

Table 1. *Test volumes.*

EMDID	Sym	Size (n)
2660	C1	360
0667	C2	480
0731	C3	486
0882	C4	160
21376	C5	256
11516	C7	512
21143	C8	256
6458	C11	448
30913	D2	110
20016	D3	384
22462	D4	320
9233	D7	400
21140	D11	324
4179	T	200
24494	I	432

Note. Each volume is a three-dimensional array of size $n \times n \times n$, with n specified on the third column. The symmetry of each volume is given by the second column.

following (4), O_{est} is an estimate of gO for some arbitrary $g \in G_1$, where $G_1 \subseteq SO(3)$ is the symmetry group of ϕ_1. In order to calculate the difference between O and O_{est}, we have to find the symmetry element g. In our tests, the symmetry group G_1 is known (see Table 1), and so we find g by solving

$$\underset{g \in G_1}{\operatorname{argmin}} \|O_{est} - gO\|_F, \tag{29}$$

followed by defining $O'_{est} = g^T O_{est}$. Next, the error in the estimated rotation O'_{est} is calculated using the axis-angle representation of rotations as follows. The axis of the rotation O is defined to be the unit vector $v \in \mathbb{R}^3$ that satisfies $Ov = v$, that is, v is an eigenvector of O corresponding to eigenvalue 1. Similarly, we define the unit vector $v' \in \mathbb{R}^3$ to be the axis of the rotation O'_{est}. Then, we calculate the angle between the axes of the rotations as

$$e_1 = \cos^{-1}\left(v^T v'\right). \tag{30}$$

The angle of rotation of the matrix O around its axis v is given by $\theta_1 = \cos^{-1}(u \cdot Ou)$, where $u \in \mathbb{R}^3$ is a unit vector perpendicular to v. Similarly, we define θ_2 to be the angle of rotation of the matrix O'_{est} around its axis v'. The error in the rotation angle is then defined as

$$e_2 = |\theta_1 - \theta_2|. \tag{31}$$

We start by investigating the appropriate value for the downsampling parameter n_{ds} (see Section 6). To that end, for each of the volumes in Table 1, we create its rotated and shifted copy and apply our algorithm with the downsampling parameter equal to 16, 32, 64, and 128 (namely, we actually align downsampled copies of the volumes and then apply the estimated parameters to the original volumes). The results are shown in Figure 2. For each value of downsampling, we show a bar plot that summarizes the results for all test volumes. Note that these results are without the refinement procedure of Section 7. To provide a more detailed information on the chosen downsampling value, we show in Figure 3 only the results for downsampling to sizes 64 and 128. Based on these results, we use a downsampling value of 64 in all subsequent tests. In particular, this value of downsampling results in an accurate initialization of the refinement procedure of Section 7, as shown in Figure 4. As of timing, we show in Figure 5 the timing, without and with refinement, for downsampling to sizes 64 and 128.

https://doi.org/10.1017/S2633903X23000089 Published online by Cambridge University Press

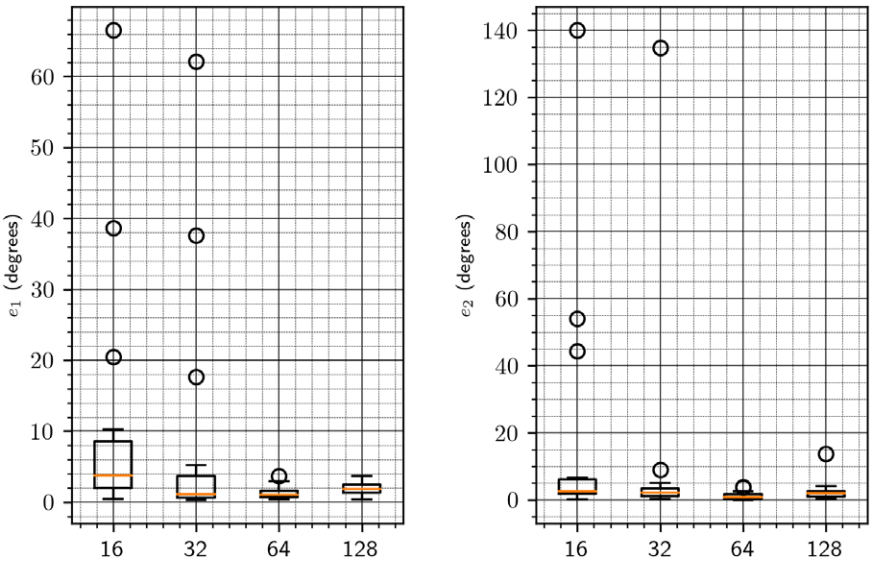


Figure 2. *Downsampling parameter versus accuracy of the algorithm. The left figure corresponds to the error e_1 in the rotation axis (see (30)). The right figure corresponds to the error e_2 in the rotation angle (see (31)).*

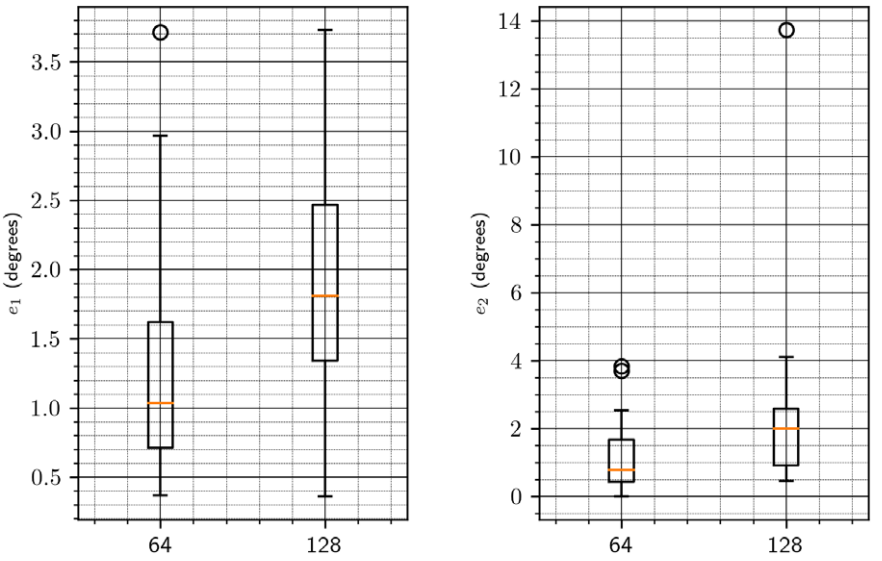


Figure 3. *Downsampling parameter versus accuracy of the algorithm, focused on 64 and 128. See Figure 2 for more details.*

Next, we wish to determine the number of reference projections N to use in Algorithms 1 and 2. We set the downsampling parameter to 64 and measure the estimation error for different numbers of reference projections. The results are summarized in Figure 6. We also show the timing for different numbers of reference projections, without and with refinement, in Figure 7. Based on these results, we choose the number of reference projections to be 30, as a good compromise between accuracy and speed.

https://doi.org/10.1017/S2633903X23000089 Published online by Cambridge University Press

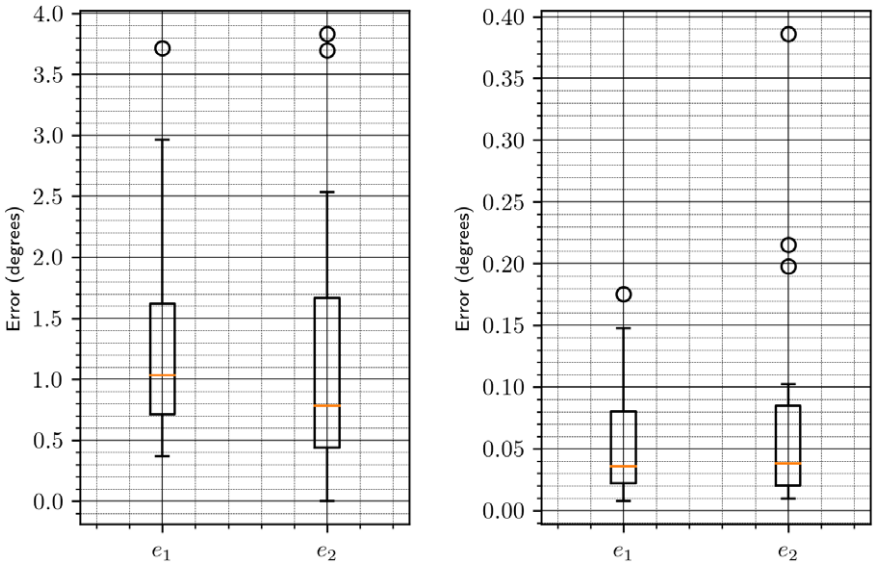


Figure 4. *Error without (left figure) and with (right figure) refinement for downsampling to size* $64 \times 64 \times 64$*. The error reported in the figure is either* e_1 *(30) or* e_2 *(31), as shown in the x-axis.*

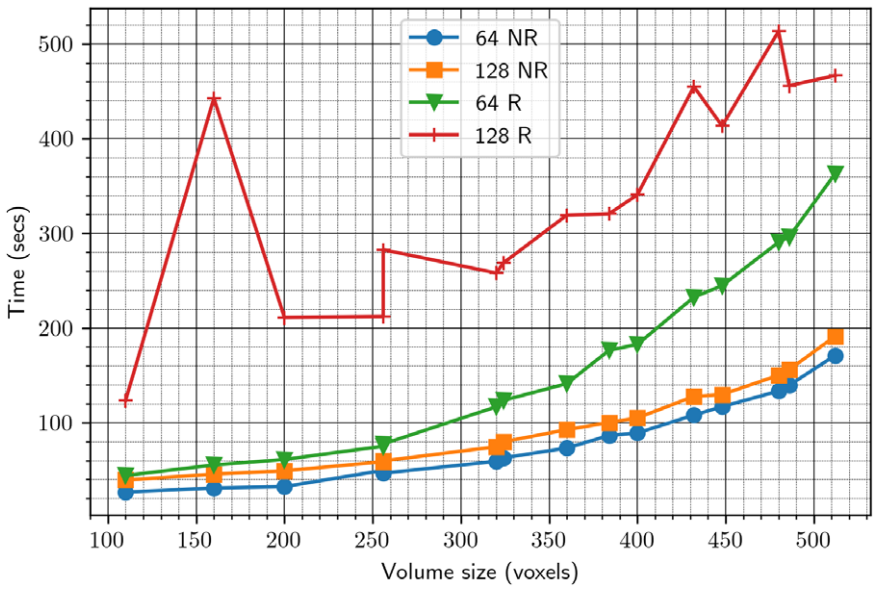


Figure 5. *Timing of the alignment algorithm with downsampling to sizes 64 and 128. NR stands for "without refinement"; R stands for "with refinement."*

Next, we compare the performance of our algorithm with that of EMAN2's and Xmipp's alignment algorithms. The accuracy and timing results are summarized in Tables 2 and 3, respectively. Finally, we demonstrate the performance of the different algorithms for noisy input volumes. To that end, we use as a reference volume EMD 2660[22] from EMDB (of size $360 \times 360 \times 360$ voxels) and create its rotated and translated copy. We add to the reference volume and its rotate/translated copy additive Gaussian noise with SNR (signal-to-noise ratio) ranging from 1 to 1/256. A central slice from the noisy reference volume

https://doi.org/10.1017/S2633903X23000089 Published online by Cambridge University Press

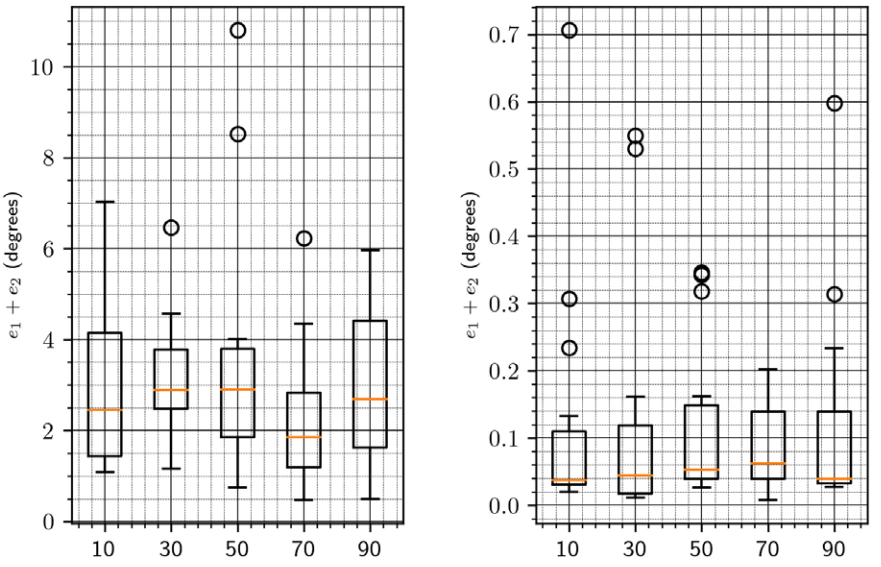


Figure 6. *Error versus the number of reference projections N. The left and right figures show the error without and with the refinement procedure of Section 7, respectively. The error reported in this figure is the sum $e_1 + e_2$ given in (30) and (31).*

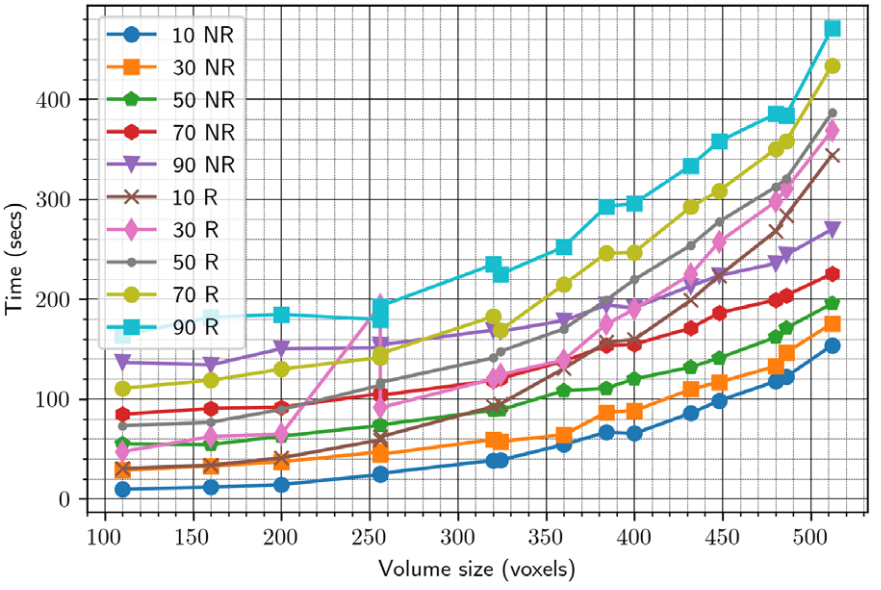


Figure 7. *Time versus the number of reference projections.*

at different levels of SNR is shown in Figure 8. The accuracy results of all algorithms for the various SNRs are shown in Table 4. The timings of the different algorithms are shown in Table 5.

9. Discussion and Conclusions

In this paper, we proposed a fully automatic method for aligning three-dimensional volumes with respect to rotation, translation, and reflection. While the parameters of the algorithm can be tuned whenever

https://doi.org/10.1017/S2633903X23000089 Published online by Cambridge University Press

Table 2. *Accuracy comparison with EMAN2 and Xmipp.*

Sym	EMDID	EMalign(NR)	EMalign(R)	EMAN	Xmipp
C1	2660	2.802	0.094	0.094	5.557
C2	667	3.747	0.223	0.121	7.181
C3	731	8.664	5.131	0.135	48.058
C4	882	1.952	0.041	0.408	0.317
C5	21376	2.949	0.358	0.507	15.445
C7	11516	3.961	0.397	0.153	5.745
C8	21143	1.502	0.536	0.455	2.778
C11	6458	2.825	0.116	0.046	0.314
D2	30913	6.273	0.035	0.425	0.141
D3	20016	3.499	0.075	0.033	1.564
D4	22462	6.016	0.126	0.095	0.251
D7	9233	4.034	0.063	0.029	5.866
D11	21140	3.183	0.042	0.247	0.127
T	4179	1.324	0.556	0.348	6.246
I	24494	3.268	0.030	0.114	0.028
Mean		3.733	0.522	0.214	6.641
Std		1.945	1.288	0.168	12.206

Note. The errors reported in this table are the sum $e_1 + e_2$ given in (30) and (31). Errors are given in degrees. For EMalign, (NR) corresponds to "without refinement" and (R) to "with refinement." The two bottom rows show the mean and standard deviation of the error (in degrees) over all experiments.

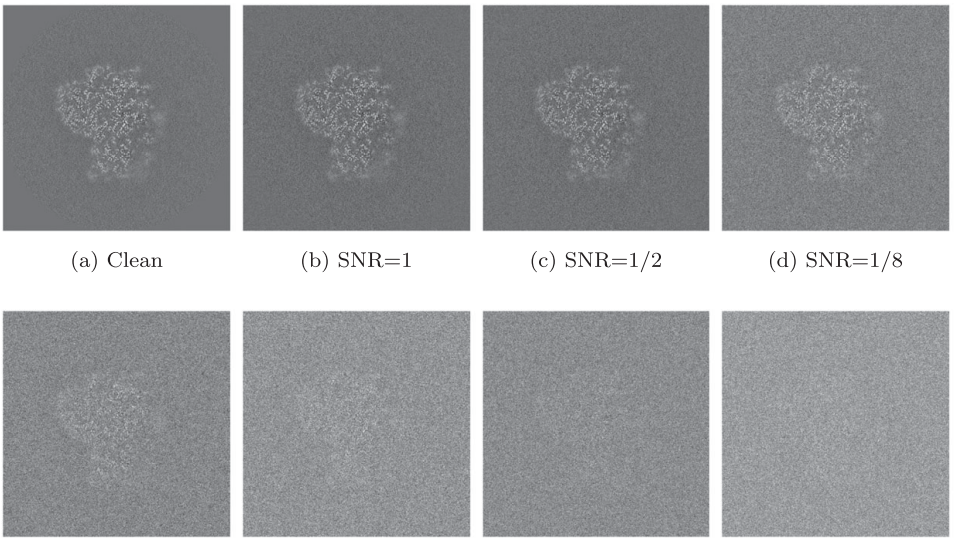
Table 3. *Timing comparison with EMAN2 and Xmipp (in seconds).*

Sym	EMDID	size	EMalign(NR)	EMalign(R)	EMAN	Xmipp
C1	2660	360	49	130	172	2106
C2	667	480	80	235	354	5812
C3	731	486	85	173	351	5582
C4	882	160	18	55	66	91
C5	21376	256	24	58	155	529
C7	11516	512	78	216	425	6854
C8	21143	256	33	55	120	698
C11	6458	448	54	151	276	3854
D2	30913	110	16	34	55	37
D3	20016	384	41	105	197	2214
D4	22462	320	29	87	201	1095
D7	9233	400	40	124	171	2970
D11	21140	324	35	79	197	1175
T	4179	200	21	59	80	246
I	24494	432	54	158	281	3313

Note. For EMalign, (NR) corresponds to "without refinement" and (R) to "with refinement". The column "size" is the side length of the input volumes.

needed, we showed that the default parameters work very well for a wide range of volumes of various symmetries. We also developed an auxiliary algorithm, which finds the orientation of a volume giving rise to a given projection image (Section 5). This algorithm may serve as a fast and highly accurate substitute to projection matching.

https://doi.org/10.1017/S2633903X23000089 Published online by Cambridge University Press



(a) Clean (b) SNR=1 (c) SNR=1/2 (d) SNR=1/8

(e) SNR=1/32 (f) SNR=1/64 (g) SNR=1/128 (h) SNR=1/256

Figure 8. *Central slice of the noisy reference volume at different SNRs.*

Table 4. *Accuracy comparison for noisy input volumes at different SNRs.*

SNR	EMalign(NR)	EMalign(R)	EMAN	Xmipp
Clean	4.066	0.143	0.072	0.968
1	3.715	0.145	0.072	0.898
1/2	1.827	0.150	0.072	0.851
1/8	5.733	0.291	0.072	0.728
1/32	5.014	3.318	0.095	0.811
1/64	4.283	0.598	0.105	1.124
1/128	2.727	0.691	0.202	1.177
1/256	4.449	25.089	0.124	1.598
1/512	92.569	97.549	0.288	1.662

Note. See Table 2 for more details.

Table 5. *Timing comparison for noisy input volumes at different SNRs.*

SNR	EMalign(NR)	EMalign(R)	EMAN	Xmipp
Clean	33	102	181	2097
1	35	122	176	2091
1/2	39	97	175	2177
1/8	33	105	170	2121
1/32	33	100	155	1991
1/64	38	97	157	2125
1/128	39	113	179	2297
1/256	36	101	163	2092
1/512	39	92	167	2365

Note. All timings are given in seconds.

https://doi.org/10.1017/S2633903X23000089 Published online by Cambridge University Press

The core difference between our approach and other existing approaches is that our approach is based on common lines between projection images generated from the volumes. The advantage of this approach is that inspecting each candidate rotation is very fast, as it is based on one-dimensional operations on the common lines ($O(n)$ operations for volumes of size $n \times n \times n$). We also note that our cost function (26) for identifying the optimal alignment is different than in other algorithms. While the typical cost function used by alignment algorithms is the correlation between the volumes, our cost function is the average correlation of the common lines between projection images of the volumes. These two cost functions are not equivalent, and while in our experiments we have not identified a scenario where one cost function is superior over the other, having tools that are based on different principles may prove beneficial in the future.

From the comparison of our algorithm with the alignment algorithms in EMAN2 and Xmipp, we conclude that our algorithm can be used in one of two modes. If we are interested in fast alignment with good accuracy (average error of 1.9 degrees of the rotation axis, and average error of 1.86 degrees of the in-plane rotation angle, with standard deviations of 1.25 degrees and 1.3 degrees, respectively), we can use our algorithm without the refinement procedure of Section 7. This is appropriate, for example, for visualization, as such an initial alignment is sufficient as an input for high resolution optimization-based alignment algorithms, such as the one in Chimera[3]. In such a case, our algorithm is more than 3 times faster than EMAN2's algorithm (even though our algorithm is implemented entirely in Python), and almost 40 times faster than Xmipp's algorithm. If we are interested in very low alignment errors, the refinement procedure of Section 7 brings the average errors down to 0.25 degrees for the rotation axis and 0.28 degrees for the in-plane rotation angle (with standard deviations of 0.66 degrees and 0.63 degrees, respectively). In such a case, our algorithm is 80% faster than EMAN2's and 15 times faster than Xmipp's.

As of noise robustness, we see from Table 4 that our algorithm performs well down to SNR = 1/128. The algorithms in EMAN2 and Xmipp give very accurate results at even lower SNRs, but at the cost of a much higher running time. As a future research direction, there are several ways to improve the robustness of our algorithm to noise. First, as our algorithm is based on generating projection images of the volumes, it is possible to apply image denoising methods to the projection images. Then, it is possible to incorporate denoising into the common lines matching step (step 4 in Algorithm 2), for example, by denoising the common lines as one-dimensional signals or by incorporating frequency-dependent weights into (26). This is expected to significantly improve the robustness of our algorithm to noise while increasing its running time only slightly.

Competing interest. The authors declare no competing interests exist.

Authorship contribution. Conceptualization: Y.S.; Funding acquisition: Y.S.; Methodology: Y.H. and Y.S.; Software: Y.H. and Y.S.; Supervision: Y.S.; Writing—original draft: Y.H. and Y.S. All authors approved the final submitted draft.

Funding statement. This research was supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement 723991—CRYOMATH) and by the NIH/NIGMS Award R01GM136780–01.

Data availability statement. Code and documentation of the algorithm are available on https://pypi.org/ and at https://github.com/ShkolniskyLab/emalign.

References

1. Singer A & Shkolnisky Y (2012) *Center of Mass Operators for Cryo-EM–Theory and Implementation*, pp. 147–177. Boston, MA: Springer US.
2. Van Heel M & Schatz M (2005) Fourier shell correlation threshold criteria. *J Struct Biol* **151**(3), 250–262.
3. Pettersen EF, Goddard TD, Huang CC, *et al.* (2004) UCSF chimera–a visualization system for exploratory research and analysis. *J Comput Chem* **25**(13), 1605–1612.
4. Yu L, Snapp RR, Ruiz T & Radermacher M (2013) Projection-based volume alignment. *J Struct Biol* **182**, 93–105.
5. Radermacher M (1994) Three-dimensional reconstruction from random projections: orientational alignment via Radon transforms. *Ultramicroscopy* **53**(2), 121–136.

https://doi.org/10.1017/S2633903X23000089 Published online by Cambridge University Press

6. de la Rosa-Trevín JM, Otón J, Marabini R, *et al.* (2013) Xmipp 3.0: an improved software suite for image processing in electron microscopy. *J Struct Biol* **184**(2), 321–328.
7. Chen Y, Pfeffer S, Hrabe T, Schuller JM & Förster F (2013) Fast and accurate reference-free alignment of subtomograms. *J Struct Biol* **182**(3), 235–245.
8. Kovacs JA & Wriggers W (2002) Fast rotational matching. *Acta Crystallogr Sect D* **58**(8), 1282–1286.
9. Kuglin C.D & Hines D.C. (1975) The phase correlation image alignment method. *Proc. Int. Conf. on Cybernetics and Society* 163–165.
10. Tang G, Peng L, Baldwin PR, *et al.* (2007) EMAN2: an extensible image processing suite for electron microscopy. *J Struct Biol* **157**, 38–46.
11. EMAN Manual page. Align3D. https://blake.bcm.edu/emanwiki/Align3D (accessed 20 March 2023).
12. van Heel M (1999) Pointgroup symmetry of oligomeric macromolecules. *Structure* **7**, 1575–1583.
13. Cucuringu M, Singer A & Cowburn D (2012) Eigenvector synchronization, graph rigidity and the molecule problem. *J IMA* **1** (1), 21–67.
14. Arun KS, Huang TS & Blostein SD (1987) Least-squares fitting of two 3-D point sets. *IEEE Trans Pattern Anal Mach Intell* **5**, 698–700.
15. Singer A & Shkolnisky Y (2011) Three-dimensional structure determination from common lines in cryo-EM by eigenvectors and semidefinite programming. *SIAM J Imaging Sci* **4**(2), 543–572.
16. Shkolnisky Y & Singer A (2012) Viewing direction estimation in cryo-EM using synchronization. *SIAM J Imaging Sci* **5**(3), 1088–1110.
17. Barnett AH, Magland JF & af Klinteberg L (2019) A parallel non-uniform fast Fourier transform library based on an "exponential of semicircle" kernel. *SIAM J Sci Comput* **41**(5), C479–C504.
18. Barnett AH (2021) Aliasing error of the $\exp\left(\beta\sqrt{1-z^2}\right)$ kernel in the nonuniform fast Fourier transform. *Appl Comput Harmon Anal* **51**, 1–16.
19. Press WH, Teukolsky SA, Vetterling WT & Flannery BP (2007) *Numerical Recipes*, 3rd ed. Cambridge, USA: Cambridge University Press.
20. ASPIRE - algorithms for single particle reconstruction. http://spr.math.princeton.edu/.
21. Lawson CL, Patwardhan A, Baker ML, et al. (2015) EMDataBank unified data resource for 3DEM. *Nucl Acid Res* **44**(D1), D396–D403.
22. Wong W, Bai X, Brown A, *et al.* (2014) Cryo-EM structure of the plasmodium falciparum 80s ribosome bound to the anti-protozoan drug emetine. *eLife* **3**, e03080.
23. Singer A, Zhao Z, Shkolnisky Y & Hadani R (2011) Viewing angle classification of cryo-electron microscopy images using eigenvectors. *SIAM J Imaging Sci* **4**, 723–759.
24. Reddy BS & Chatterji BN (1996) An FFT-based technique for translation, rotation, and scale-invariant image registration. *IEEE Trans Image Process* **5**(8), 1266–1271.

A. Appendix. Common lines

In this section, we review the Fourier projection slice theorem and its induced common line property. Given a volume ϕ and a rotation matrix R, the projection image of ϕ corresponding to orientation R is given by (5). We identify $R^{(3)}$ (the third column of R) as the viewing direction of ϕ (see[23]). The first two columns $R^{(1)}$ and $R^{(2)}$ of R form an orthonormal basis for a plane in $\mathbb{R}^3$ which is perpendicular to the viewing direction $R^{(3)}$. Therefore, if R_i and R_j are two rotations with the same viewing direction ($R_i^{(3)} = R_j^{(3)}$), then the two projection images P_i and P_j generated according to (5) look the same up to some in-plane rotation.

The Fourier projection slice theorem relates the two-dimensional Fourier transform of a projection image P to the three-dimensional Fourier transform of ϕ. Let

$$\widehat{P}(\omega_x,\omega_y) = \iint_{\mathbb{R}^2} P(x,y)e^{-i\left(x\omega_x+y\omega_y\right)}dxdy$$

be the two-dimensional Fourier transform of $P(x,y)$, and let

$$\widehat{\phi}(\omega_x,\omega_y,\omega_z) = \iiint_{\mathbb{R}^3} \phi(x,y,z)e^{-i\left(x\omega_x+y\omega_y+z\omega_z\right)}dxdydz$$

be the three-dimensional Fourier transform of $\phi(x,y,z)$. The Fourier projection slice theorem[16] states that

$$\widehat{P}(\omega_x,\omega_y) = \widehat{\phi}\left(\omega_x R^{(1)} + \omega_y R^{(2)}\right), \tag{A.1}$$

where P is defined in (5). (Equation A.1) states that the two-dimensional Fourier transform of each projection image P is equal to a planar slice of the three-dimensional Fourier transform of ϕ. Moreover, it states that this planar slice is the plane $\omega_x R^{(1)} + \omega_y R^{(2)}$. The Fourier projection slice theorem (A.1) holds, up to discretization errors, also for discrete volumes and their sampled projection images.

From (A.1), we get that any two Fourier transformed projection images $\widehat{P}_i$ and $\widehat{P}_j$ with different viewing directions ($R_i^{(3)} \neq R_j^{(3)}$) are equal to two different planar slices from $\widehat{\phi}$. Since there exists a line that is common to both planar slices, the two Fourier transformed images share a common line. We refer to that line as the common line between P_i and P_j. We denote the angle that this

https://doi.org/10.1017/S2633903X23000089 Published online by Cambridge University Press

line makes with the local x-axis of the (Fourier transformed) images $\widehat{P}_i$ and $\widehat{P}_j$ by α_{ij} and α_{ji}, respectively. Mathematically, the common line property is expressed as[16]

$$\widehat{P}_i\left(\xi\cos\alpha_{ij},\xi\sin\alpha_{ij}\right)=\widehat{P}_j\left(\xi\cos\alpha_{ji},\xi\sin\alpha_{ji}\right),\quad \xi\in\mathbb{R},$$

implying that the samples of the Fourier transformed images along the common line are equal.

To find an expression for the angles α_{ij} and α_{ji}, we consider the unit vector

$$q_{ij}=\frac{R_i^{(3)}\times R_j^{(3)}}{\|R_i^{(3)}\times R_j^{(3)}\|},$$

where $\times$ is the cross product between vectors. Define the unit vectors

$$c_{ij}=\left(\cos\alpha_{ij},\sin\alpha_{ij},0\right)^T,\quad c_{ji}=\left(\cos\alpha_{ji},\sin\alpha_{ji},0\right)^T.$$

It can be shown[16] that these vectors satisfy the equation

$$R_ic_{ij}=q_{ij}=R_jc_{ji},$$

which implies that c_{ij} and c_{ji} can be computed as

$$c_{ij}=R_i^Tq_{ij},\quad c_{ji}=R_i^Tq_{ij},$$

from which α_{ij} and α_{ji} can be easily extracted.

B. Appendix. Constructing the set S

We generate the set of candidate rotations S by using the Euler angles representation for rotations. Let L be a positive integer, and let τ,θ,φ be Euler angles. We construct S by sampling the Euler angles in equally spaced increments as follows. First, we sample $\tau\in\left\{0,\ldots,\frac{\pi}{2}\right\}$ at $\lfloor\frac{L}{4}\rfloor$ points. Then, for each τ, we sample $\theta\in\{0,\ldots,\pi\}$ at $\lfloor\frac{L}{2}\sin(\tau)\rfloor$ points. Finally, for each pair (τ,θ), we sample $\varphi\in\{0,\ldots,2\pi\}$ at $\lfloor\frac{L}{2}\sin(\tau)\sin(\theta)\rfloor$ points. For each (τ,θ,φ) on this grid, we compute a corresponding rotation matrix R by

$$R=R_z(\tau)R_y(\theta)R_x(\varphi),$$

where

$$R_z(\tau)=\begin{pmatrix}\cos\tau & -\sin\tau & 0\\ \sin\tau & \cos\tau & 0\\ 0 & 0 & 1\end{pmatrix},$$

$$R_y(\theta)=\begin{pmatrix}\cos\theta & 0 & \sin\theta\\ 0 & 1 & 0\\ -\sin\theta & 0 & \cos\theta\end{pmatrix},$$

$$R_x(\varphi)=\begin{pmatrix}1 & 0 & 0\\ 0 & \cos\varphi & -\sin\varphi\\ 0 & \sin\varphi & \cos\varphi\end{pmatrix}.$$

C. Appendix. Translation estimation

For completeness, we review the well-known phase correlation procedure for translation estimation[9]. Consider two volumes ϕ_1 and ϕ_2 shifted relative to one another, that is,

$$\phi_2(r)=\phi_1(r-t),\quad r=(x,y,z)^T\in\mathbb{R}^3,\tag{C.2}$$

where $t=\left(\Delta_x,\Delta_y,\Delta_z\right)^T\in\mathbb{R}^3$. Our goal is to estimate t.

First, by the Fourier shift property, the Fourier transforms of ϕ_1 and ϕ_2 satisfy

$$\widehat{\phi}_2\left(\omega_x,\omega_y,\omega_z\right)=\widehat{\phi}_1\left(\omega_x,\omega_y,\omega_z\right)e^{-i\left(\omega_x\Delta_x+\omega_y\Delta_y+\omega_z\Delta_z\right)}.\tag{C.3}$$

From (C.3), we get[24]

$$\begin{aligned}\widehat{\rho}\left(\omega_x,\omega_y,\omega_z\right)&=\frac{\widehat{\phi}_1\widehat{\phi}_2^{\,*}}{|\widehat{\phi}_1\widehat{\phi}_2^{\,*}|}=\frac{\widehat{\phi}_1\widehat{\phi}_1^{*}e^{i\left(\omega_x\Delta_x+\omega_y\Delta_y+\omega_z\Delta_z\right)}}{|\widehat{\phi}_1\widehat{\phi}_1^{*}|}\\&=e^{i\left(\omega_x\Delta_x+\omega_y\Delta_y+\omega_z\Delta_z\right)},\end{aligned}\tag{C.4}$$

since $|e^{i\left(\omega_x\Delta_x+\omega_y\Delta_y+\omega_z\Delta_z\right)}|=1$. Then, since the inverse Fourier transform of a complex exponential is a Dirac delta, we have

https://doi.org/10.1017/S2633903X23000089 Published online by Cambridge University Press

$$\rho(x,y,z) = \delta\left(x+\Delta_x, y+\Delta_y, z+\Delta_z\right). \tag{C.5}$$

Therefore, $t = \left(\Delta_x, \Delta_y, \Delta_z\right)^T$ is given by

$$\left(\Delta_x, \Delta_y, \Delta_z\right) = -\underset{(x,y,z)}{\operatorname{argmax}}\, \rho(x,y,z). \tag{C.6}$$

While this appendix is formulated in the continuous domain, the same holds if we replace ϕ_1 and ϕ_2 by their discrete versions sampled on a regular grid and replace the Fourier transform by the discrete Fourier transform.

Cite this article: Harpaz Y and Shkolnisky Y (2023). Three-dimensional alignment of density maps in cryo-electron microscopy. *Biological Imaging*, 3: e8. doi:https://doi.org/10.1017/S2633903X23000089