Supporting Information

Quantitative Characterization of Domain Motions in Molecular Machines

Suvrajit Maji[1], Rezvan Shahoei[2,3.4], Klaus Schulten[2,3,4*] and Joachim Frank[1,5,6*]

[1]Department of Biochemistry and Molecular Biophysics, Columbia University

[2]Department of Physics, University of Illinois at Urbana-Champaign

[3]Center for the Physics of Living Cells, Department of Physics, University of Illinois at Urbana-Champaign

[4]Beckman Institute for Advanced Science and Technology, University of Illinois at Urbana-Champaign

[5]Howard Hughes Medical Institute, Columbia University

[6]Department of Biological Sciences, Columbia University

*corresponding authors

*Joachim Frank (Tel: (212) 305 9510. Email: **jf2192@cumc.columbia.edu)**

*Klaus Schulten (Tel: 217-244-1604. Email**: kschulte@ks.uiuc.edu)**

Text S1. Volume Segmentation

Distance Transform. The Distance Transform[1-6] of a volumetric image (or density map) is defined as the distance of every voxel X from its nearest non-zero valued voxel, and is given by

$$D_T(X) = \min\left\{ d_p(X,Y) \,\middle|\, Y \in O,\ X \in O \cup O^c \right\} \qquad S1.1$$

where $d_p(X,Y)$ is a distance function (of order p or p-norm as shown in equation $S1.3$), O is the set of object voxels, and O^c is the set of background voxels. Alternatively, we can also define the distance transform as the distance of each voxel from its nearest zero-valued voxel (in that case, $Y \in O^c$ in equation $S1.1$). Since we calculate the Distance Transform (Section 1.2 step (d)) on the complementary binary map, $\sim M_{bin}$, we will show how we define the object voxels for $\sim M_{bin}$. Given the Gaussian filtered map M_f, the binary map M_{bin} is defined as:

$$M_{bin}(k) = \begin{cases} 1\ , & M_f(k) > threshold \\ 0\ , & otherwise \end{cases}$$

where *k* is the number of the voxel

$S1.2$

Then, for the complementary binary map, the values of voxels are the reverse of those in $M_{bin}(k)$. So, the object voxels for $\sim M_{bin}$ are defined as the non-zero voxels, i.e., voxels *k* for which $\sim M_{bin}(k) = 1$.

A commonly used distance function[7-8] d_p for calculating the Distance Transform (equation $S1.1$) is defined as

$$d_p(X,Y) = \|X - Y\|_{l_p} = \left(\sum_{k=1}^{n} |x_k - y_k|^p\right)^{1/p}$$

$S1.3$

$where\ X = (x_1, x_2, \dots, x_n),\ Y = (y_1, y_2, \dots, y_n)$ such that $X, Y \in \mathbb{R}^n$. Then X, Y and d_p define the metric space $(\mathbb{R}^{\boldsymbol{n}}, d_p)$ with d_p as the distance metric. For $p = 1$ we obtain the l_1-norm, also known as the Manhattan norm; for $p = 2$ we obtain the l_2-norm, or the Euclidean norm; and for the case where $p \to \infty$, we obtain the l_∞-norm or maximum norm.

In our case the image dataset is three-dimensional ($n = 3$) and we have used the l_1-norm as the distance metric since this choice seemed to generate better segmentation for our 3D volume images, compared to the typically used l_2-norm

$$d_1(X,Y) = \sum_{k=1}^{3} |x_k - y_k| .$$

$S1.4$

Segmentation Steps. The segmentation steps are summarized below.

a. Convert pdb structure into the corresponding 3D density map : M_{vol}
b. Apply a Gaussian filter to M_{vol} to obtain a smooth map M_f

c. Obtain a binary map: $M_{bin}(k) = \begin{cases} 1\ , & M_f(k) > threshold \\ 0\ , & otherwise, \end{cases}$

where *k* is the number of the voxel.

For simulated maps (derived directly from a pdb structure as in our case), with no background noise, we use $threshold = 0$, otherwise we can choose a higher value for experimental density maps with noise. We can inspect the binary map for proper thresholding (Figure 1A). Alternatively, we can also use an automatic thresholding method[9] based on density values.

d. Generate the Distance Transform of the complementary binary map: $\sim M_{bin} \rightarrow D_T$ where the elements of D_T are obtained from equation 1.1 using the distance function d (equation 1.2).

e. Invert the Distance Transform matrix to convert the high-density regions (potential object location) into catchment basins: $D_T \coloneqq -D_T$

f. Make the background or non-object voxels ($\sim object\_voxels$) for M_{bin} in D_T (from step (e)) as $-inf$; i.e., force the background to be its own catchment basin.

g. Apply the Watershed Algorithm (Section 1.1) on D_T (from step (f)) to obtain segments S_r.

h. Apply the region merging step as described above to merge over-segmented regions S_r into S_m ($m < r$). It may be possible to obtain a pre-specified number of segments through iterative application of the merging criteria (maximum merging size and merging level) but at present we have chosen to use the merging criteria already tested on the first input dataset and were able to get the expected segmentation for the subsequent datasets.

i. Once we have the segmentation parameters from one input dataset, we perform the segmentation (Figure 2) on the fly for multiple input pdb structures by using the residue list from the first input pdb structure so as to obtain consistent segmented domains on all pdb structures, assuming the subsequent structures are obtained from the same source such as a simulation trajectory. However, if the structures originate from mixed sources then we can use the initial segments derived from the first pdb structure as masks for each of the subsequent structures, potentially with missing residues, to extract the enclosed residues.

Text S2. Inertia Tensors. Tensors[10] are compact mathematical constructs that describe the linear mapping defined on a set of vectors or scalars**.** By definition, scalars are treated as zero-order tensors, vectors are first-order tensors, and 2D vectors or matrices are second-order tensors. In general, tensors have N^R elements where N is the dimension and R is the rank of the tensor. inertia tensors[10] are second-order tensors and they are represented by N^2 elements. For a 3D object, the inertia tensor $\boldsymbol{I}$ is a 3×3 matrix that describes the distribution of mass and, when the object is in motion, it provides the relationship between the angular velocity ($\vec{\omega}$) and angular momentum ($\vec{L}$):

$$\vec{L} = \boldsymbol{I}\vec{\omega} \qquad S2.1$$

Suppose we have an arbitrary rigid body (Figure S2A) and let $\vec{\boldsymbol{r}} = x\,\hat{\imath} + y\,\hat{\jmath} + z\,\hat{k}$ be the position vector of an infinitesimal mass element dm. Then the inertia tensor $\boldsymbol{I}$ for the rigid body is expressed as below:

$$\boldsymbol{I} = \begin{bmatrix} \int (y^2 + z^2) dm & -\int xy\, dm & -\int xz\, dm \\ -\int xy\, dm & \int (z^2 + x^2) dm & -\int yz\, dm \\ -\int zx\, dm & -\int yz\, dm & \int (x^2 + y^2) dm \end{bmatrix} \qquad S2.2$$

or, element-wise, the tensor matrix can be represented as

$$\boldsymbol{I} = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix} \qquad S2.3$$

The diagonal terms of ***I*** are called *moments of inertia* whereas the off-diagonal terms are called *products of inertia*. In general, the angular momentum $\vec{L}$ and angular velocity $\vec{\omega}$ are not in the same direction (as we can see from equation $S2.1$), producing a torque on the rigid body, but in a special case both the vectors are in the same direction, making the description of rotational motion simpler. Such a rotation occurs around the axis known as the principal axis. Therefore, the inertia tensor matrix is transformed into the principal axes form where all off-diagonal terms are zero. The diagonal terms are called *principal moments of inertia.*

The principal axes of inertia tensor ***I*** are coordinate axes such that ***I*** is a diagonal matrix. Thus determining these axes is equivalent to solving the eigenvalue problem:

$$\boldsymbol{I}\vec{\omega} = \lambda\vec{\omega} \qquad S2.4$$

On solving equation $S2.4$, we obtain the three eigenvalues $\lambda = \{\mathrm{I_{kk}}\}_{k=1\ldots3}$ which are the moments of inertia . Then we can obtain the diagonalized matrix ***I*** as:

$$\boldsymbol{I} = \begin{bmatrix} I_{11} & 0 & 0 \\ 0 & I_{22} & 0 \\ 0 & 0 & I_{33} \end{bmatrix} \qquad S2.5$$

Now we can use the eigenvalues $\{\mathrm{I_{kk}}\}_{k=1\ldots3}$ to compute the corresponding eigenvectors $\{\hat{\boldsymbol{e}}_{\boldsymbol{k}}\}_{k=1\ldots3}$, which are the principal axes introduced above.

For a symmetric object, the axis of symmetry is always one of the three orthogonal principal axes. For example if we have a circular disk (Figure S2B), then one of the axis of symmetry ($z-axis$) passing through the center is a principal axis. The other two orthogonal axes are arbitrarily chosen on the $x-y$ plane since two of the eigenvalues or principal moments of inertia will have the same value, making the disk axisymmetric. For a perfect sphere, there is an infinite number of ways to select the principal axes as there are infinite axes of symmetry passing through the center of the sphere. For an arbitrarily shaped, asymmetric object, the three principal axes define the three major direction of mass distribution.

There are a few additional aspects and features worth mentioning about the ordering of the principal axes based on the eigenvalues or principal moment of inertia and the implication for rotational motion. For a rigid body undergoing a rotational motion, the largest eigenvalue or the largest principal moment (e.g. I_{11}) corresponds to the rotation about the principal axis (axis 1) perpendicular to the largest cross-section and the smallest moment (e.g. I_{33}) is for the rotation about the principal axis (axis 3) perpendicular to the smallest cross-section (axis 3). This can also be interpreted in terms of the shape of the rigid body (Figure S2C). The mass is most elongated along the principal axis (axis 3) about which the moment is the smallest (I_{33}) and least elongated along the principal axis (axis 1) about which the moment is the largest (I_{11}). Also, in the context of a rigid body motion and following the conservation of angular momentum, we can say that the

angular velocity around axis 3 has the largest magnitude since the moment of inertia (I_{33}) is the least, and the angular velocity around axis 1 has the least magnitude, since the moment of inertia (I_{11}) is the largest.

Text S3. Definition and Properties of Quaternions. Quaternions[11] have been widely applied in computer vision and robotics [12], animations [13] and also in spacecraft orientation [14]. Quaternion (Hamilton, 1843) is a 4-dimensional extension to complex numbers and can be thought of as a vector with 4 components which can describe rotations and orientation in three dimensions. Quaternions are generally represented as

$$\mathring{q} = q_w + q_x\, i + q_y\, j + q_z\, k \;\; , where \;\; q_w, q_x, q_y, q_z \in \mathbb{R}$$

Here $i^2 = j^2 = k^2 = -1$ and $ij = k = -ji, jk = i = -kj, ki = j = -ik$

$S3.1$

It can also be expressed as $\mathring{q} = (q_w, \boldsymbol{q_v})$, where q_w is the scalar part and $\boldsymbol{q_v} = (q_x, q_y, q_z)$ is the vector part**. .** The norm of a quaternion is defined as

$$||\mathring{q}|| = \sqrt{\mathring{q}\mathring{q}^*} = \sqrt{\mathring{q} \cdot \mathring{q}} = \sqrt{{q_w}^2 + {q_x}^2 + {q_y}^2 + {q_z}^2} \qquad S3.2$$

where $\mathring{q}^*$ is the conjugate of $\mathring{q}$ and defined as

$$\mathring{q}^* = q_w - q_x\, i - q_y\, j - q_z\, k. \qquad S3.3$$

Note that $\mathring{q}^*$ can also be represented as $\mathring{q}^* = (q_w, -\boldsymbol{q_v})$.

Rotation of any vector can be represented by a unit quaternion $\mathring{q}$, where $\mathring{q}\mathring{q}^* = 1$ and also $\mathring{q} \cdot \mathring{q} = 1$ and such rotation preserves the length of the vector. Also $\mathring{q}$ and -$\mathring{q}$ gives the same rotation. Except for this ambiguity, every rotation is uniquely represented by a unit quaternion, which can also be described using an angle-axis representation as:

$$\mathring{q} = (\cos(\theta/2),\ \sin(\theta/2)\ \hat{\boldsymbol{e}}) \qquad S3.4$$

where θ is the angle of rotation and $\hat{\boldsymbol{e}} = e_x\hat{\imath} + e_y\,\hat{\jmath} + e_z\hat{k}$ is a unit vector representing the axis of rotation. We can obtain $S3.2$ from $S3.1$ as :

$$\theta = 2\cos^{-1}(q_w),\quad \hat{\boldsymbol{e}} = \left(e_x,\ e_y,\ e_z\right)^T = \frac{1}{\sqrt{1-{q_w}^2}}\left(q_x,\ q_y,\ q_z\right)^T \qquad S3.3$$

We can now see that the unit quaternion has a geometrically intuitive representation for any 3D rotation (Figure 4D).

The rotation of any vector can be represented by a unit quaternion $\mathring{q}$. Let a 3D vector be defined as $\boldsymbol{r} = x\,\hat{\imath} + y\,\hat{\jmath} + z\,\hat{k}$, where $\hat{\imath}, \hat{\jmath}, \hat{k}$ are the unit vectors in x,y,z directions. It can be represented as a purely imaginary quaternion with a zero scalar part ($r_w = 0$) as

$$\mathring{r} = 0 + r_x\,i + r_y\,j + r_z\,k \qquad S3.4$$

The product of quaternions can also be expressed using the product of a 4×4 orthogonal matrix and a 4×1 vector as:

$$\mathring{r}\mathring{q} = \begin{bmatrix} r_w & -r_x & -r_y & -r_z \\ r_x & r_w & -r_z & r_y \\ r_y & r_z & r_w & -r_x \\ r_z & -r_y & r_x & r_w \end{bmatrix} \mathring{q} = M\mathring{q} \qquad S3.5$$

$$\mathring{q}\mathring{r} = \begin{bmatrix} r_w & -r_x & -r_y & -r_z \\ r_x & r_w & r_z & -r_y \\ r_y & -r_z & r_w & r_x \\ r_z & r_y & -r_x & r_w \end{bmatrix} \mathring{q} = \bar{M}\mathring{q} \qquad S3.6$$

M and $\bar{M}$ are orthogonal and for a purely imaginary quaternion they are also skew-symmetric i.e. we have $M^T = -M,\ \bar{M}^T = -\bar{M}$.

If we apply a unit quaternion $\mathring{q}$ on a vector $\mathring{r}$, we obtain a rotated vector $\mathring{r}'$ given by

$$\mathring{r}' = \mathring{q}\mathring{r}\mathring{q}^*$$

$$= (-\mathring{q})\mathring{r}(-\mathring{q}^*) \qquad S3.6$$

We can see that $\|\mathring{r}'\| = \|\mathring{r}\|$ as° $\mathring{q} \cdot \mathring{q} = 1$; therefore rotation preserves the length of the vector $\mathring{r}$.

We should point out some features of rotation using unit quaternions compared to using rotation matrices. We need fewer arithmetic operations to multiply two unit quaternions compared to multiplying two rotation matrices, in order to obtain the same rotation. Moreover, due to numerical precision issues, the length of the vector may change or the norm of unit quaternion may not be exactly equal to one after the rotation operation. It is easier to normalize a quaternion to obtain a unit quaternion than to obtain the nearest orthonormal matrix. However, rotation matrices are recommended when a large number of points are involved in the rotation.

Text S4. Closed-form solution to the least-squares problem of Absolute Orientation. The original paper by Horn[15] provides more details than what we are using, hence we will present only the relevant mathematical background here for an easier understanding of the method.

Horn's method finds a closed-form solution to the least-square problem of absolute orientation using unit quaternions, for three or more points, although, for matching two sets of points when the data have been corrupted considerably, the algorithm proposed by Umeyama[16] may be more appropriate as the other method produces reflections instead of rotation in such cases. Umeyama's algorithm gives us rotation matrices and is similar in essence to Horn's method, except for a correction step for taking account of the reflection, and this could be incorporated later in our toolset. At present, however, we use Horn's method as it produces a compact solution using unit quaternion and we are dealing with principal axes coordinates which are consistent across a set of structures.

Let us assume that we have the transformation from coordinate system 1 to coordinate system 2 and let the corresponding set of points be $\{\boldsymbol{r_{1,i}}\}_{i=1\ldots n}$ and $\{\boldsymbol{r_{2,i}}\}_{i=1\ldots n}$, respectively. If $\boldsymbol{r_1}$ and $\boldsymbol{r_2}$ are vectors in the two systems with $R(\boldsymbol{r_1})$ as the vector obtained after rotating $\boldsymbol{r_1}$, then the transformation from one coordinate system to the other can be represented as

$$\boldsymbol{r_2} = s\, R(\boldsymbol{r_1}) + \boldsymbol{r_0} \qquad S4.1$$

Here s is the scale factor and $\boldsymbol{r_0}$ is the translation offset. As rotation preserves the length of a vector we have

$$\|R(\boldsymbol{r_1})\|^2 = \|\boldsymbol{r_1}\|^2 \qquad S4.2$$

The residual error for the transformation of each point can be calculated as

$$\boldsymbol{e_i} = \boldsymbol{r_{2,i}} - s\, R\left(\boldsymbol{r_{1,i}}\right) - \boldsymbol{r_0} \qquad S4.3$$

We want the calculations to be made relative to the centroid of the measured points for simplification. The centroids in the systems 1 and 2 are given by

$$\bar{\boldsymbol{r}}_{\boldsymbol{1}} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{r_{1,i}}\ , \qquad \bar{\boldsymbol{r}}_{\boldsymbol{2}} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{r_{2,i}} \qquad S4.4$$

Let $\boldsymbol{r'_{1,i}}$ and $\boldsymbol{r'_{2,i}}$ be the centroid-subtracted coordinates in systems 1 and 2, given by:

$$\boldsymbol{r'_{1,i}} = \boldsymbol{r_{1,i}} - \bar{\boldsymbol{r}}_{\boldsymbol{1}}\ , \quad \boldsymbol{r'_{2,i}} = \boldsymbol{r_{2,i}} - \bar{\boldsymbol{r}}_{\boldsymbol{2}} \qquad S4.5$$

Then the modified translational offset $\boldsymbol{r'_0}$ is given by

$$\boldsymbol{r'_0} = \boldsymbol{r_0} - \bar{\boldsymbol{r}}_{\boldsymbol{2}} + \boldsymbol{s}R(\bar{\boldsymbol{r}}_{\boldsymbol{1}}) \qquad S4.6$$

It follows from equations $S4.4$ and $S4.5$ that

$$\sum_{i=1}^{n} \boldsymbol{r'_{1,i}} = 0\ , \qquad \sum_{i=1}^{n} \boldsymbol{r'_{2,i}} = 0\ , \quad \sum_{i=1}^{n} R\left(\boldsymbol{r'_{1,i}}\right) = 0 \qquad S4.7$$

Then the residual error for each point can be written as

$$\boldsymbol{e_i} = \boldsymbol{r'_{2,i}} - s\, R\left(\boldsymbol{r'_{1,i}}\right) - \boldsymbol{r'_0} \qquad S4.8$$

and the objective is to minimize the sum of the squared residual errors

$$\boldsymbol{E} = \sum_{i=1}^{n} \|\boldsymbol{e_i}\|^2 = \sum_{i=1}^{n} \left\|\boldsymbol{r'_{2,i}} - s\, R\left(\boldsymbol{r'_{1,i}}\right) - \boldsymbol{r'_0}\right\|^2$$

$S4.9$

$$= \sum_{i=1}^{n} \left\|\boldsymbol{r'_{2,i}} - s\, R\left(\boldsymbol{r'_{1,i}}\right)\right\|^2 - 2\,\boldsymbol{r'_0} \cdot \sum_{i=1}^{n} \left(\boldsymbol{r'_{2,i}} - s\, R\left(\boldsymbol{r'_{1,i}}\right)\right) + n\, \|\boldsymbol{r'_0}\|^2.$$

$S4.10$

Now from equation $S4.7$ it follows that the middle term is zero, so we have

$$\boldsymbol{E} = \sum_{i=1}^{n} \left\|\boldsymbol{r'_{2,i}} - s\, R\left(\boldsymbol{r'_{1,i}}\right)\right\|^2 + n\, \|\boldsymbol{r'_0}\|^2.$$

$S4.11$

First we aim to find $\boldsymbol{r_0}'$ such that it minimizes $\boldsymbol{E}$. Since the first term is independent of $\boldsymbol{r_0}'$ and $\|\boldsymbol{r_0}'\| \geq 0$. Therefore $\boldsymbol{E}$ can only be minimized with respect to $\boldsymbol{r'_0}$ when $\boldsymbol{r'_0} = 0$. So, from equation 4.6 we have

$\boldsymbol{r_0} - \bar{\boldsymbol{r}}_{\boldsymbol{2}} + \boldsymbol{s}R(\bar{\boldsymbol{r}}_{\boldsymbol{1}}) = 0$ or

$$\boldsymbol{r_0} = \bar{\boldsymbol{r}}_{\boldsymbol{2}} - \boldsymbol{s}R(\bar{\boldsymbol{r}}_{\boldsymbol{1}}) \qquad S4.12$$

Then from equation 4.10 we have

$$\boldsymbol{E} = \sum_{i=1}^{n} \left(\left\|\boldsymbol{r'_{2,i}}\right\|^2 - 2s\, \boldsymbol{r'_{2,i}} \cdot R\left(\boldsymbol{r'_{1,i}}\right) + s^2 \left\|R\left(\boldsymbol{r'_{1,i}}\right)\right\|^2\right).$$

$S4.13$

Noting the relation in equation $S4.2$, let us denote the terms in the equation $S4.13$ as

$$S_1 = \sum_{i=1}^{n} \left\| \boldsymbol{r'}_{\mathbf{1},\boldsymbol{i}} \right\|^2 , \quad S_2 = \sum_{i=1}^{n} \left\| \boldsymbol{r'}_{\mathbf{2},\boldsymbol{i}} \right\|^2$$

$S4.14$

$$D_R = \sum_{i=1}^{n} \boldsymbol{r'}_{\mathbf{2},\boldsymbol{i}} \cdot R\left(\boldsymbol{r'}_{\mathbf{1},\boldsymbol{i}}\right) .$$

$S4.15$

Here we will consider only the relevant symmetric case from the Horn paper. The transformations from coordinate system 1 to 2 and coordinate system 2 to 1 should be symmetric in terms of the scale factor. The scale factor when we go from $\boldsymbol{r'}_{\mathbf{1},\boldsymbol{i}} \rightarrow \boldsymbol{r'}_{\mathbf{2},\boldsymbol{i}}$ using $R\left(\boldsymbol{r'}_{\mathbf{1},\boldsymbol{i}}\right)$ is inverse of the scale factor when we go from $\boldsymbol{r'}_{\mathbf{2},\boldsymbol{i}} \rightarrow \boldsymbol{r'}_{\mathbf{1},\boldsymbol{i}}$ using $R\left(\boldsymbol{r'}_{\mathbf{2},\boldsymbol{i}}\right)$. So, instead of the expression in equation $S4.13$ we should use a symmetric expression for the residual error for transformation; by dividing the right hand side by s to get

$$\boldsymbol{E} = \frac{1}{s} S_2 - 2D_R + s\, S_1 .$$

$S4.16$

We can complete the square in s for $\boldsymbol{E}$ as follows

$$\boldsymbol{E} = \left(\sqrt{s}\,\sqrt{S_1} - \frac{1}{\sqrt{s}}\sqrt{S_2}\right)^2 + 2\left(\sqrt{S_1}\sqrt{S_2} - D_R\right) .$$

$S4.17$

We can see that $\boldsymbol{E}$ can be minimized w.r.t s if we have the first term as zero.

$$\Rightarrow s = \left(\frac{S_2}{S_1}\right)^{1/2} = \left(\sum_{i=1}^{n} \left\|\boldsymbol{r'}_{\boldsymbol{2,i}}\right\|^2 \Big/ \sum_{i=1}^{n} \left\|\boldsymbol{r'}_{\boldsymbol{1,i}}\right\|^2\right)^{1/2} .$$

$S4.18$

Therefore the optimal scale factor is essentially the ratio of the root mean square deviation of the points in coordinate systems 2 and 1.

$$\boldsymbol{E}|_{s=s^*} = 2\left(\sqrt{S_1 S_2} - D_R\right) . \qquad S4.19$$

So far we have seen that getting the optimal translation offset and scale factor is independent of the rotation of the points for the best transformation. Now to find the best rotation we focus on the term D_R describing the rotation. From equation $S4.19$ we can see that the residual error is further minimized if we maximize the term D_R. The objective is to find a unit quaternion $\mathring{q}$ that maximizes the sum of the dot products (D_R) of corresponding coordinates in system 2 with the rotated coordinates in system 1. Also the vectors can be represented using the corresponding (purely imaginary) quaternion and the rotation $R\left(\boldsymbol{r'}_{\boldsymbol{1,i}}\right)$ can be expressed using a unit quaternion products such as $\mathring{q}\, \mathring{r}'_{1,i}\, \mathring{q}^*$. Therefore from equation we have:

$$D_R = \sum_{i=1}^{n} (\mathring{q}\, \mathring{r}'_{1,i}\, \mathring{q}^*) \cdot \mathring{r}'_{2,i} ,$$

$S4.20$

where $\mathring{q}\, r'_{1,i}\, \mathring{q}^*$ are the rotated coordinates of system 1, with the unit quaternion $\mathring{q}$ causing the rotation. So the sum of dot product D_R becomes larger as each vector in coordinate system 2 gets closer to the corresponding rotated vector in the system 1 and we obtain a better estimate of rotation.

Also, we have $(\mathring{q}\,\mathring{p}) \cdot (\mathring{q}\,\mathring{r}) = \mathring{p} \cdot \mathring{r}$, since $\mathring{q} \cdot \mathring{q} = 1$, and $(\mathring{p}\ \mathring{q}\,) \cdot \mathring{r} = \mathring{p} \cdot (\mathring{r}\,\mathring{q}^{*})$ then equation $S4.20$ becomes

$$D_R = \sum_{i=1}^{n} (\mathring{q}\,\mathring{r}'_{1,i}\,) \cdot \left(\mathring{r}'_{2,i}\,\mathring{q}\right).$$

$S4.21$

Using equations $S3.5$ and $S3.6$ we can express the products in $S4.21$ as

$$\mathring{q}\mathring{r}'_{1,i} = \bar{M}_1 \mathring{q} \qquad S4.22$$

$$\mathring{r}'_{2,i}\mathring{q} = M_2 \mathring{q} \qquad S4.23$$

Using equations $S4.22$ and $S4.23$ in equation 4.21, the sum to be maximized is:

$$D_R = \sum_{i=1}^{n} \left(\bar{M}_{1,i}\,\mathring{q}\right) \cdot \left(M_{2,i}\,\mathring{q}\right)$$

$$= \mathring{q}^T \left(\sum_{i=1}^{n} \bar{M}_{1,i}{}^T M_{2,i} \right) \mathring{q} \qquad \text{[since matrix multiplication is associative]}$$

$$= \mathring{q}^T \left(\sum_{i=1}^{n} N_i \right) \mathring{q}$$

$$= \mathring{q}^T N \mathring{q},$$

$S4.24$

where $N_i = \bar{M}_{1,i}{}^T M_{2,i}$ and $N = \sum_{i=1}^{n} N_i$. The matrices N_i's are symmetric and hence N is symmetric. It is shown in Horn's paper that the unit quaternion which maximizes $D_R = \mathring{q}^T N \mathring{q}$ is

the eigenvector of the matrix N corresponding to the largest positive eigenvalue and is unique if the eigenvector is distinct.

Therefore, the remaining task is to perform the eigen-decomposition of N. To this end we first compute the matrix M, whose elements are sums of products of coordinates measured in the centroid-corrected system 1 and 2, and which is given by

$$M = \sum_{i=1}^{n} \boldsymbol{r'}_{\mathbf{1},i}\, \boldsymbol{r'^{T}}_{\mathbf{2},i} \qquad S4.25$$

$$M = \begin{bmatrix} S_{xx} & S_{xy} & S_{xz} \\ S_{yx} & S_{yy} & S_{yz} \\ S_{zx} & S_{zy} & S_{zz} \end{bmatrix} \qquad S4.26$$

where $\boldsymbol{S_{xx}} = \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \boldsymbol{x'}_{\mathbf{1},i} \cdot \boldsymbol{x'}_{\mathbf{2},i}$ and $\boldsymbol{S_{xy}} = \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \boldsymbol{x'}_{\mathbf{1},i} \cdot \boldsymbol{y'}_{\mathbf{2},i}$, and so on.

Next, using the elements of M, we can construct the 4 x 4, real and symmetric matrix N as shown below:

$$N = \begin{bmatrix} (S_{xx}+S_{yy}+S_{zz}) & S_{yz}-S_{zy} & S_{zx}-S_{xz} & S_{xy}-S_{yx} \\ S_{yz}-S_{zy} & (S_{xx}-S_{yy}-S_{zz}) & S_{xy}+S_{yx} & S_{zx}+S_{xz} \\ S_{zx}-S_{xz} & S_{xy}+S_{yx} & (-S_{xx}+S_{yy}-S_{zz}) & S_{yz}+S_{zy} \\ S_{xy}-S_{yx} & S_{zx}+S_{xz} & S_{yz}+S_{zy} & (-S_{xx}-S_{yy}+S_{zz}) \end{bmatrix} \qquad S4.27$$

The final step is to perform the eigen-decomposition of matrix N and compute the eigenvector $\boldsymbol{e}_{max}$ with the most positive eigenvalue, as discussed earlier. The unit quaternion $\mathring{q}$ in the direction of $\boldsymbol{e}_{max}$ is the required solution to the absolute orientation problem.

Text S5. Decomposition of a Unit Quaternion. To determine the decomposition[17-19] of a unit quaternion, we followed the algorithm (Section S5.1) described in the paper[19] on swing-twist decomposition of a spinor and derived an equivalent approach by replacing the spinor with a unit quaternion, since a spinor in three dimensions is a quaternion. A spinor is represented as $\mathring{s} = a + b\boldsymbol{e_{12}} + c\boldsymbol{e_{23}} + d\boldsymbol{e_{31}}$, where the unit bivectors $\boldsymbol{e_{23}}, \boldsymbol{e_{31}}, \boldsymbol{e_{12}}$ are respectively equal to i, j, k for quaternions ($S3.1$). Using the steps in the section below we can decompose the unit quaternion $\mathring{q}$ into its twist ($\mathring{q}_t$) and swing ($\mathring{q}_s$) components such that $\mathring{q} = \mathring{q}_s\mathring{q}_t$. This particular approach is called as swing after twist decomposition.

S5.1. Given a unit quaternion $\mathring{q} = (q_w, q_x, q_y, q_z)$ and a non-zero vector $\vec{v} = (v_x,\ v_y,\ v_z)^T$ we can determine the decomposition of $\mathring{q} = \mathring{q}_s\mathring{q}_t$ using the following steps:

a. $u \leftarrow v_x q_x + v_y q_y + v_z q_z$

b. $n \leftarrow {v_x}^2 + {v_y}^2 + {v_z}^2$

c. $m \leftarrow q_w\, n$

d. $l \leftarrow \sqrt{m^2 + u^2 n}$

e. $\mathring{q}_t \leftarrow \frac{m}{l} + \frac{v_x u}{l} i + \frac{v_y u}{l} j + \frac{v_z u}{l} k$

f. $\mathring{q}_s \leftarrow \mathring{q}\ \mathring{q}_t^{-1}$

We will illustrate how $\mathring{q}_t$, as determined above, is equivalent to a projection of $\mathring{q}$ onto $\vec{v}$ and determines the component of rotation around the given vector $\vec{v}$.

If we let $\vec{r}_q = \left(q_x,\ q_y,\ q_z\right)^T$ be the rotation axis component of $\mathring{q}$, then we obtain the following terms (Section S5.1 steps (a) - (d)) :

$u = v_x q_x + v_y q_y + v_z q_z = \vec{r}_q \cdot \vec{v}$

$n = {v_x}^2 + {v_y}^2 + {v_z}^2 = \|\vec{v}\|^2$

$m = q_w\ n = q_w\ \|\vec{v}\|^2$

$l = \sqrt{m^2 + u^2 n} = n\sqrt{{q_w}^2 + \frac{u^2}{n}} = \|\vec{v}\|^2 \sqrt{{q_w}^2 + \frac{\left(\vec{r}_q \cdot \vec{v}\right)^2}{\|\vec{v}\|^2}} = \|\vec{v}\|^2 \sqrt{{q_w}^2 + \left\|\vec{r}_q \cdot \hat{v}\right\|^2}$, where $\hat{v} = \frac{\vec{v}}{\|v\|}$ is a unit vector along $\vec{v}$.

If we let $f = \sqrt{{q_w}^2 + \left\|\vec{r}_q \cdot \hat{v}\right\|^2}$, then we obtain the following terms (Section S5.1 step (e)):

$$\frac{m}{l} = \frac{q_w}{f}, \qquad \frac{v_x u}{l} = \frac{\left(\vec{r}_q \cdot \vec{v}\right)}{f\|\vec{v}\|^2} v_x, \qquad \frac{v_y u}{l} = \frac{\left(\vec{r}_q \cdot \vec{v}\right)}{f\|\vec{v}\|^2} v_y, \qquad \frac{v_z u}{l} = \frac{\left(\vec{r}_q \cdot \vec{v}\right)}{f\|\vec{v}\|^2} v_z$$

*S*5.1

Combining the last three expressions (*S*5.1), we obtain the vector part of $\mathring{q}_t$as:

$$\mathring{q}_{tv} = \frac{(\vec{r}_q \cdot \vec{v})}{f\|\vec{v}\|^2}(v_x, v_y, v_z) = \frac{1}{f}\frac{(\vec{r}_q \cdot \vec{v})\vec{v}}{\|\vec{v}\|^2} = \frac{1}{f}\vec{P}_{r_q v}$$

$S5.2$

where $\vec{P}_{r_q v} = \frac{(\vec{r}_q \cdot \vec{v})\vec{v}}{\|\vec{v}\|^2} = (\vec{r}_q \cdot \hat{v})\,\hat{v}$ is the projection of $\vec{r}_q$ on $\vec{v}$. Using equations $S5.1$ and 5.2 , we obtain :

$$\mathring{q}_t = \left(\frac{q_w}{f}, \frac{1}{f}\vec{P}_{r_q v}\right) = \frac{1}{f}\left(q_w, \vec{P}_{r_q v}\right)$$

$$= \frac{1}{\sqrt{{q_w}^2 + \left\|\vec{r}_q \cdot \hat{v}\right\|^2}}\left(q_w, (\vec{r}_q \cdot \hat{v})\,\hat{v}\right)$$

$S5.3$

It is easy to verify that $\left\|\mathring{q}_t\right\| = 1$.

Also, when we choose $\vec{v} = \vec{r}_q$, it is straightforward to verify that we recover the original quaternion.

$$(\vec{r}_q \cdot \hat{v})\,\hat{v} = \left(\vec{r}_q \cdot \frac{\vec{r}_q}{\left\|\vec{r}_q\right\|}\right)\frac{\vec{r}_q}{\left\|\vec{r}_q\right\|} = \vec{r}_q$$

$S5.4$

$${q_w}^2 + \left\|\vec{r}_q \cdot \hat{v}\right\|^2 = {q_w}^2 + \left\|\vec{r}_q\right\|^2 = \|\mathring{q}\|^2 = 1 \qquad S5.5$$

and using equations $S5.4$ and $S5.5$ in equation $S5.3$ we obtain $\mathring{q}_t = (q_w, \vec{r}_q) = \mathring{q}$.

Therefore, $\mathring{q}_t$ is a unit quaternion representing the component of rotation around the given vector $\vec{v}$. The component $\mathring{q}_s$ (Section S5.1, step (f)) describes the rotation around a vector $\vec{v}_p$ which is orthogonal to $\vec{v}$.

We can either use this direct geometrical approach to construct the twist component ($S5.3$) of the unit quaternion $\mathring{q}$ or use the equivalent algorithmic approach as described above (Section S5.1).

Text S6. Conventional Rotation Angle Calculation. (a). We obtain some of the conventional angle information such as 'tilt' angle θ (polar angle) and 'swivel' angle Ψ (azimuthal), in the spherical coordinate system (r, θ, Ψ) directly from the principal axes coordinates:

$$x = rsin(\theta)cos(\Psi),\ \ y = rsin(\theta)sin(\Psi),\ \ z = rcos(\theta) \qquad S6.1$$

$$\theta = \cos^{-1}\left(\frac{z}{r}\right),\ \ \Psi = \tan^{-1}\left(\frac{y}{x}\right) \qquad S6.2$$

(b). We also measure one set of three Euler rotation angles around the three principal axes by calculating the projections of the three orthogonal axes against a reference coordinate system.

We calculate these angles when the reference system is fixed and aligned to the XYZ Cartesian coordinate system. Then we can get a measure of the motion by calculating the differences between the angles in the different states.

Text S7. Full Method Workflow. Here we briefly summarize the overall workflow using the different sections of the tool described in more details in their corresponding sections 1-3. The steps are as follows:

a. Segment the input pdb structure into relevant domains as discussed in Section 1, following the segmentation steps in Section 1.2

b. Select the domains of interest, reference domain (R_T) and target domain (Dm_T) (Figure 4A).

c**.** Compute the inertia tensor and principal axes for the reference domain as discussed in Section 2.1

d. Align the principal axes of the reference domain with the Cartesian coordinate system XYZ. Details of the alignment is provided through an example in Section 2.2. The alignment ensures that the characterization of the motion of any domain is performed relative to a fixed coordinate frame, in this case, the principal axes of the reference domain.

e. Next, compute the inertia tensor and principal axes for the target domain.

f. Repeat the steps (a) – (e) for subsequent pdb structures. We now have the principal axes of the target domain for multiple input structures (different states).

g. Finally, following the steps in Section 3.1, solve the least square absolute orientation problem for the pair of coordinate axes for the target domain transforming from state A to B. The solution of this coordinate axes transformation is given by a unit quaternion. The axis and angle of rotation for the target domain, undergoing the transformation, can now be derived from the unit quaternion.

Text S8. Implementation Details and Limitations. At present VMD has limited capability of processing volume density maps, with few available packages such as volutil and volmap which lack the tool for segmentation. For this reason we implemented the segmentation method using Matlab (The MathWorks Inc.) along with DIPimage[20] (version 2.7), a Matlab toolbox for scientific image processing. We then transformed the complete set of Matlab codes into a standalone executable program, which can be run from a command line in VMD using a Tcl wrapper script. The limitation of the executable program for segmentation is that the output, which in our case are segmented density map files, can only be analyzed using the Tcl script through an I/O approach; i.e., first saving the segmented map files to the disk and then loading the maps onto the VMD platform for further analysis. As an alternative, one can implement the same volume segmentation method in Python using the image processing library in Python and execute it on VMD.

We used the program pdb2mrc[21] for generating a volume density map (mrc) from the pdb coordinates of a structure. We used a resolution of $1.7\,\text{Å}$, and a pixel size of $1.25\,\text{Å}$ for generating the map with a size of $240 \times 240 \times 240$. In order to prevent over-segmentation, we performed a Gaussian smoothing of the density map prior to the application of the Watershed and region merging algorithms. We used a Gaussian filter of size $SZ = [1.49\ 1.49\ 1.49]$, with a standard deviation $\sigma = (SZ/2)\ /2.3548$. The resolution at which the map is generated from the pdb model and the Gaussian smoothing filter size can be adjusted as needed for obtaining a reasonable segmentation. An isovalue threshold of 0.05 was used here for extracting the residues contained within the individual segmented volumes.

The steps for computing the inertia tensor and principal axes for each domain, as discussed in Section 2, is implemented in Tcl using the Orient package[22] for VMD and the Hume

Linear Algebra Tcl package la1.0. We have modified and customized the Orient package for our specific requirements. We also implemented the absolute orientation with unit quaternion method in Tcl. In the future we expect that VMD would have an inbuilt tool for volume segmentation or at least the capability of writing segmentation code directly in VMD such that the whole toolset described here can be seamlessly integrated into one single package which would greatly improve the input/output capability and analysis on the same platform.

Text S9. Illustration of Absolute Orientation with Example data. We demonstrate the working principle of the Absolute Orientation algorithm (Section 3.1) using an example case of transformation between coordinate axes system 1 and 2. Let the coordinate system 1 (matrix columns as the axes), such as the tensor Dm_T (Figures 4A & 4B) be

$$PA_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Then we rotate the axes system PA_1 by an angle $\theta_{u,1\to 2} = 36.0$ around the axis $\hat{\boldsymbol{u}} = \frac{1}{\sqrt{6}}[1, 1, 2]$ passing through the point $x_0 = (0, \quad 0, \quad 0)^T$ (Figure 4C).

Then the transformed coordinate system 2 like the tensor $Dm_T{}'$ (Figures 4A & 4B) is

$$PA_2 = \begin{bmatrix} 0.8408 & -0.4481 & 0.3036 \\ 0.5118 & 0.8408 & -0.1763 \\ -0.1763 & 0.3036 & 0.9363 \end{bmatrix}$$

Therefore we have to solve the absolute orientation problem for the transformation $PA_1 \rightarrow PA_2$. From step (c) in section 3.1 we obtain the matrix

$$M = \begin{bmatrix} 0.6087 & 0.1197 & -0.5309 \\ -0.6802 & 0.4487 & -0.0509 \\ 0.0715 & -0.5684 & 0.5818 \end{bmatrix}$$

and from step (e) in section 3.1 we obtain the matrix

$$N = \begin{bmatrix} 1.6393 & 0.5175 & 0.6024 & 0.7999 \\ 0.5175 & -0.4218 & -0.5606 & -0.4594 \\ 0.6024 & -0.5606 & -0.7418 & -0.6193 \\ 0.7999 & -0.4594 & -0.6193 & -0.4757 \end{bmatrix}$$

The unit quaternion obtained after the eigen-decomposition of N is given by

$$\mathring{q} = (0.9511, 0.1262, 0.1262, 0.2523)^T$$

When $\mathring{q}$ is expressed in the angle-axis form (Figure 4D), we get the corresponding axis of rotation as $\hat{\boldsymbol{e}} = (0.408248,\ 0.408248,\ 0.816497)^T$ and the angle of rotation around the axis $\hat{\boldsymbol{e}}$ is $\theta = 36.0$, which is exactly the same as $\widehat{\boldsymbol{u}} = \frac{1}{\sqrt{6}}[1, 1, 2]$ and $\theta_{u,1\to 2}$, respectively. We should note that there is no change in the scale factor s for the transformation, so $s = 1$, and there is no translation, so the offset $r_0 = 0$.

Text S10. Estimating the Error in Segmentation and the Rotation Angle Calculations.

We evaluated the segmentation quality by using a segmented reference and also estimated the error in the rotation angle for the SSU domain. We have used precision and recall measures[23] as

defined below to measure the segmentation error. For a segment S and given the corresponding reference segment R, we can calculate the $Precision$ (Pc) and $Recall$ (Rc) as:

$$Pc = \frac{|R \cap S|}{|S|} = \frac{TP}{TP + FP} \qquad S10.1$$

$$Rc = \frac{|R \cap S|}{|R|} = \frac{TP}{TP + FN} \qquad S10.2$$

where TP, FP and FN denote true positive (correct identification), false positive (incorrect identification) and false negative (incorrect rejection) instances respectively.

For a series of segments $\{S_i\}_{i=1\ldots n}$, given the reference segments $\{R_i\}_{i=1\ldots m}$, we can calculate the $Precision$ (Pc) and $Recall$ (Rc) as:

$$Pc = \sum_{i=1}^{n} |R_{im} \cap S_i| \Big/ \sum_{i=1}^{n} |S_i| \qquad S10.3$$

$$Rc = \sum_{i=1}^{m} |R_i \cap S_{im}| \Big/ \sum_{i=1}^{m} |R_i| \qquad S10.4$$

where R_{im} and S_{im} are the segments with maximum overlap with the individual segments $\{S_i\}_{i=1\dots n}$ and $\{R_i\}_{i=1\dots m}$, respectively. We can also calculate the $-measure$, which is the weighted harmonic mean of the $Precision$ (Pc) and $Recall$ (Rc) values, given by:

$$F = \frac{1}{\alpha \frac{1}{Pc} + (1-\alpha)\frac{1}{Rc}}$$

$S10.5$

Here, we have calculated all three measures based on the number of atoms in the segmented domains relative to the atoms in the reference domains. The reference model for the ribosome comprised the LSU with 221495 atoms, the SSU with 136903 atoms and a bound ligand as third domain (red segment in Figures S3, S4, S5) with 12943 atoms. As mentioned earlier (Results and Discussion section), we excluded the third domain from all structures for all subsequent analysis. For calculating the $F-measure$ (Table S3), we have used $\alpha = 0.5$.

We evaluated several instances of segmentation by varying the parameters (Text S8, Table S1) and summarized the quality measures (Figure S6, Table S3). For each case (Table S1), we calculated the SSU rotation from structure A to B and the error in SSU rotation angle (Table S1). Since a reference segmentation for SSU body versus SSU head was not available, we have reported (Figure 6C) the SSU Body and SSU Head rotation for all the cases. The detailed rotation calculations with rotation axes and rotation angles for case 1 is presented in Table 1.

Case	Voxel Size (Å)	Effective Resolution of the Segmented Map (Å)	Merge-Level	Max Merge Size x,y,z (Voxels)	θ_{SSU} (degree)	$\Delta\theta_{SSU} = \theta_{SSU}(Ref) - \theta_{SSU}$
$0\ (Ref)$	–	–	–	–	10.62	–
1	1.25	2.44	11	50^3	10.24	0.38
2	1.25	2.36	11	50^3	10.17	0.45
3	1.25	2.46	11	50^3	10.29	0.33
4	1.26	2.40	10	50^3	10.28	0.34
5	1.25	2.92	11	50^3	9.67	0.95
6	1.25	2.65	11	50^3	10.78	−0.16
7	1.25	2.60	11	50^3	9.91	0.71
8	1.25	2.62	11	50^3	11.50	−0.88
9	1.25	2.86	11	50^3	8.79	1.83
10	1.26	2.91	10	50^3	9.03	1.59
11	1.25	3.56	10	50^3	8.63	1.99
12	1.26	2.77	10	50^3	9.84	0.78
13	1.26	2.79	10	50^3	9.91	0.71
14	1.26	4.09	10	50^3	8.6	2.02
15	1.25	2.93	11	50^3	10.0	0.62

Table S1. Comparison of SSU rotation for the various cases of segmentation (Table S3) and the reference model with known LSU and SSU segmentation.

Case	Voxel Size (Å)	Effective Resolution of the Segmented Map (Å)	Merge-Level	Max Merge Size x,y,z (Voxels)	θ_{body} (degree)	θ_{head} (degree)
1	1.25	2.44	11	50^3	14.26	16.68
2	1.25	2.36	11	50^3	13.76	16.20
3	1.25	2.46	11	50^3	13.84	16.09
4	1.26	2.40	10	50^3	14.22	16.44
5	1.25	2.92	11	50^3	11.82	20.10
6	1.25	2.65	11	50^3	09.28	11.44
7	1.25	2.60	11	50^3	11.83	12.96
8	1.25	2.62	10	50^3	10.97	12.44
9	1.25	2.86	11	50^3	10.96	12.09
10	1.26	2.91	10	50^3	11.31	12.85
11	1.25	3.56	10	50^3	10.64	12.50
12	1.26	2.77	10	50^3	13.83	15.49
13	1.26	2.79	10	50^3	14.04	15.69
14	1.26	4.09	10	50^3	10.28	12.01
15	1.25	2.93	11	50^3	12.34	18.90

Table S2. SSU Body and SSU Head rotation angle estimates for the segmentation cases shown in Table S1.

Case	Domain	Segmented Atoms (S)	Correctly Segmented Atoms	$Precision$ (Pc)	$Recall$ (Rc)	$F-measure$ (F)
0 (Ref)	LSU	221495	–	–	–	–
	SSU	136903	–	–	–	–
	All	358398	–	–	–	–
1	LSU	220831	218129	0.9878	0.9848	0.9863
	SSU	139844	135633	0.9699	0.9907	0.9802
	All	360675	353762	0.9808	0.9871	0.9839
2	LSU	221034	218164	0.9870	0.9850	0.9860
	SSU	139270	135193	0.9707	0.9875	0.9790
	All	360304	353357	0.9807	0.9859	0.9833
3	LSU	221111	218517	0.9883	0.9866	0.9874
	SSU	139189	135483	0.9734	0.9896	0.9814
	All	360300	354000	0.9825	0.9877	0.9851
4	LSU	220664	218455	0.9900	0.9863	0.9881
	SSU	139791	135931	0.9724	0.9929	0.9825
	All	360455	354386	0.9832	0.9888	0.9860
5	LSU	217919	215260	0.9878	0.9719	0.9798
	SSU	144375	135868	0.9411	0.9924	0.9661
	All	362294	351128	0.9692	0.9797	0.9744
6	LSU	214594	213261	0.9938	0.9628	0.9781
	SSU	147484	136188	0.9234	0.9948	0.9578
	All	362078	349449	0.9651	0.9750	0.9701
7	LSU	217825	214896	0.9866	0.9702	0.9783
	SSU	144777	135175	0.9337	0.9874	0.9598
	All	362602	350071	0.9654	0.9768	0.9711
8	LSU	212687	211243	0.9932	0.9537	0.9731
	SSU	149615	136135	0.9099	0.9944	0.9503
	All	362302	347378	0.9588	0.9693	0.9640
9	LSU	229764	217859	0.9482	0.9836	0.9656

	SSU	132998	127287	0.9571	0.9298	0.9432
	All	362762	345146	0.9514	0.9630	0.9572
10	LSU	229480	218054	0.9502	0.9845	0.9670
	SSU	132914	127526	0.9595	0.9315	0.9453
	All	362394	345580	0.9536	0.9642	0.9589
11	LSU	226685	216111	0.9533	0.9757	0.9644
	SSU	136867	128258	0.9371	0.9369	0.9370
	All	363552	344369	0.9472	0.9609	0.9540
12	LSU	219098	216628	0.9887	0.9780	0.9833
	SSU	143468	135853	0.9469	0.9923	0.9691
	All	362566	352481	0.9722	0.9835	0.9778
13	LSU	219411	216893	0.9885	0.9792	0.9839
	SSU	142611	135635	0.9511	0.9907	0.9705
	All	362022	352528	0.9738	0.9836	0.9787
14	LSU	226785	215625	0.9508	0.9735	0.9620
	SSU	136880	127923	0.9346	0.9344	0.9345
	All	363665	343548	0.9447	0.9586	0.9516
15	LSU	218936	216592	0.9893	0.9779	0.9835
	SSU	143287	135906	0.9485	0.9927	0.9700
	All	362223	352498	0.9732	0.9835	0.9783

Table S3. Evaluation of the **s**egmentation quality for various cases with the parameters reported in Tables S1, S2. The $Precision$, $Recall$ and $F-measure$ values for individual domains as well as the overall structure, calculated based on the atoms in each of the segmented domains relative to the reference domains (case 0) provides an estimate of the segmentation error.

Segmentation Case	Ranked by SSU $F-measure$	Ranked by error in SSU Rotation Angle
1	3	4
2	4	5
3	2	2
4	1	3
5	8	11
6	10	1
7	9	8
8	11	10
9	13	13
10	12	12
11	14	14
12	7	9
13	5	7
14	15	15
15	6	6

Table S4. Ranking of the Segmentation Cases based on the $F-measure$ of SSU and error in the SSU rotation angles. Case 6 has relatively high segmentation error as indicated by the $Precision$ measure of SSU and case 8 has the lowest $Precision$ measure for SSU segmentation. Case 14 has the lowest SSU $F-measur$e.

S10.1. Sensitivity of the Rotation Measurements to Segmentation Errors. If we allow the error cut-off for SSU rotation angle (Table S1, Figure 6B) to be $\pm 1.0°$ instead of $\pm 0.5°$ (Results and Discussion) then this would include the segmentation cases $5, 7, 8, 12, 13, 15$ (Table S1, S3, S4) with much lower segmentation quality ($Precision, Recall, F-measure\ between\ 0.90-0.97$).

.

Supporting Information Figures

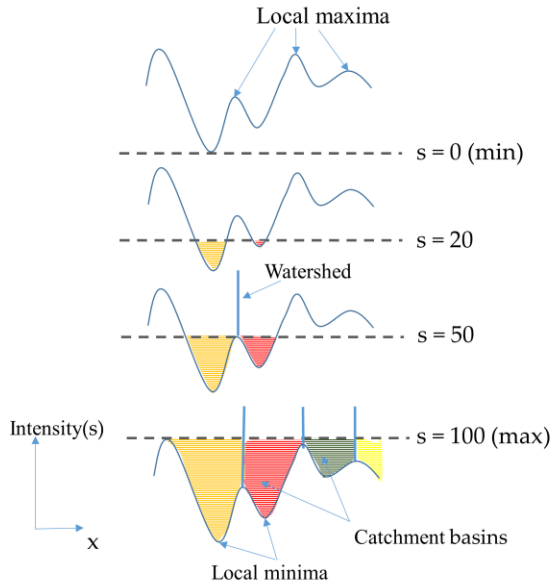

Figure S1. Illustration of classical Watershed flooding algorithm for a 1D image. The idea remains the same for a 2D image or 3D density map. ‘s’ denotes the flooding level or intensity (density) (arbitrary values here). Colored regions are catchment basins and the vertical lines are the Watershed lines demarcating the boundaries between the basins. The flooding starts from a source at the catchment basin at the minimum intensity level for the and gradually fills up the basins with increasing flooding levels until it reaches the maximum level. In the process of flooding, the Watershed is marked between two neighboring basin at the moment when the flood water starts to spill into the neighboring basin.

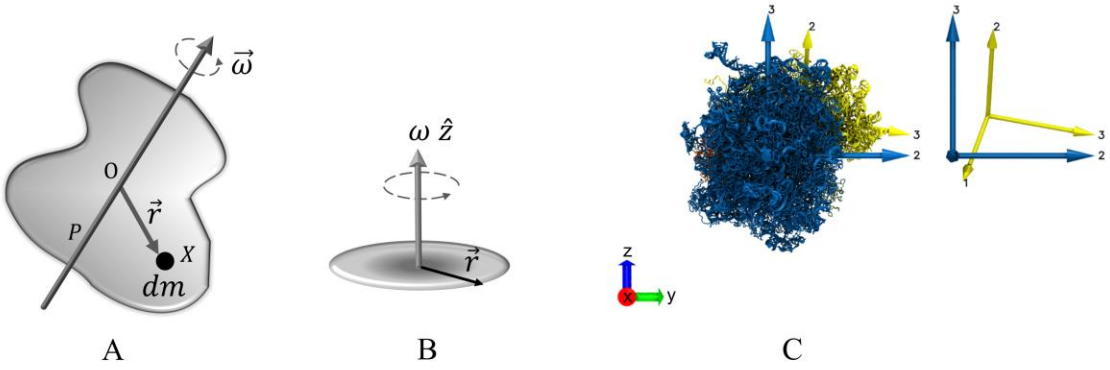

Figure S2. Inertia Tensor and Principal Axes of a Rigid Body. **A.** An arbitrary rigid body with an axis of rotation P passing through the origin O (center of mass). An infinitesimally small element with mass dm is situated at X with position vector $\vec{r}$. The rigid body is rotating with an angular velocity of $\vec{\omega}$. **B.** An example of a disc with radius r (and radial vector $\vec{r}$) rotating around Z axis passing through the center of the disc. **C.** Three principal axes computed for each of the two domains of the ribosome. The two domains shown are LSU (blue) and SSU body (yellow). The principal axes for individual domains are shown with the same color as the corresponding domain. The inset on the right panel shows the principal axes only.

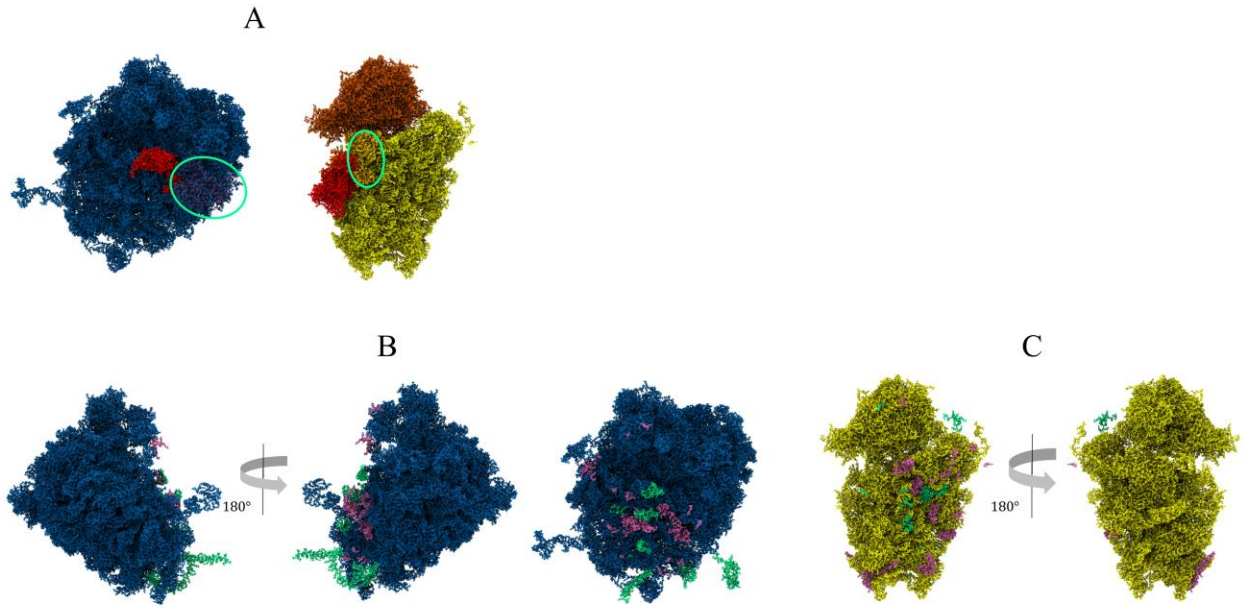

Figure S3. Segmentation Error for the Segmentation Case 1 in Tables 1, S1, S3. **A.** Initial segmentation of the three relevant domains LSU (blue), SSU Body (yellow), SSU Head (orange). The red segment could not be automatically segmented out. Part of the red segment belongs to SSU (highlighted region) and the other part to LSU (highlighted region). **B.** The false positive (incorrectly identified; pink) and false negative (incorrectly rejected; green) atoms are shown on the segmented LSU domain (blue). The third panel shows the interface view of the LSU**. C.** The false positives (pink) and false negatives (green) are shown on the segmented SSU domain (yellow). The residues belonging to the red segment are not shown in (B) and (C) and were excluded from the LSU and SSU residues in subsequent calculations.

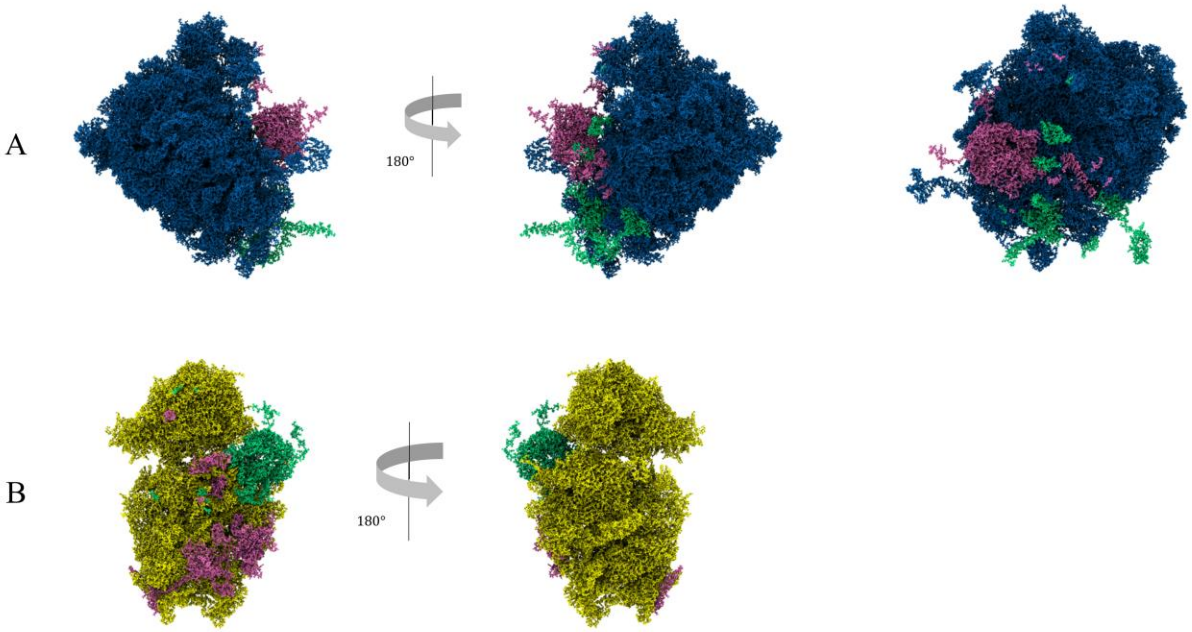

Figure S4. Segmentation Error for the Segmentation Case 14 in Tables S1, S2, S3. **A.** The false positives (pink) and false negatives (green) are shown on the segmented LSU domain (blue). The third panel shows the segmentation error in the interface view of the LSU. **B.** The false positives (pink) and false negatives (green) are shown on the segmented SSU domain (yellow). The residues belonging to the red segment were excluded from the LSU and SSU residues in subsequent calculations.

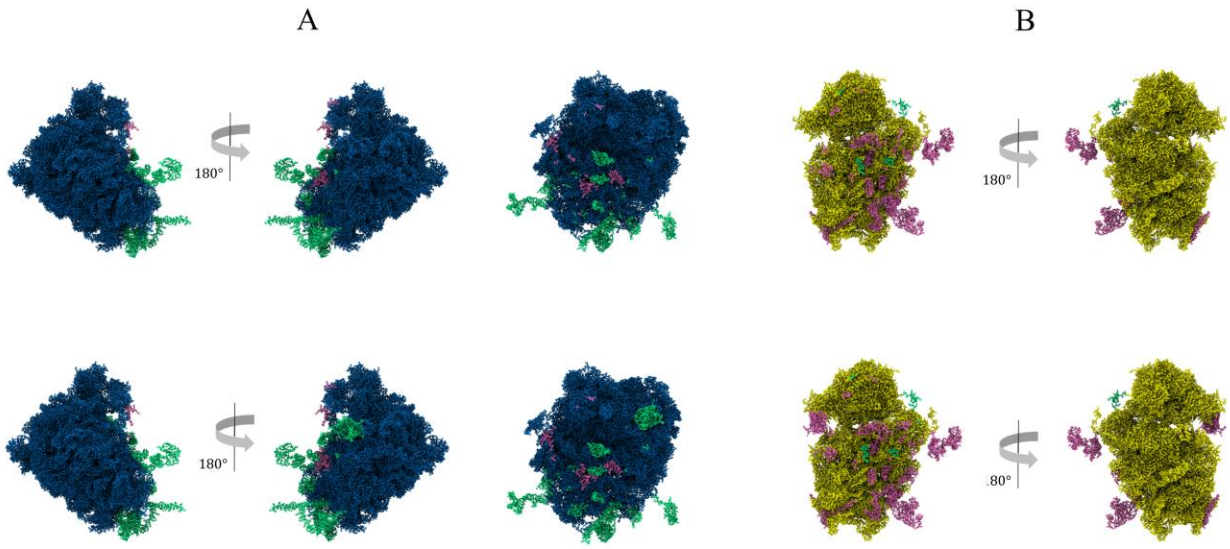

Figure S5. Segmentation Error for the Segmentation Case 6 (top row) and Case 8 (bottom row) in Tables S1, S2, S3. **A.** The false positives (pink) and false negatives (green) are shown on the segmented LSU domain (blue). The third panel shows the interface view of the LSU. **B.** The false positives (pink) and false negatives (green) are shown on the segmented SSU domain (yellow). The residues belonging to the red segment were excluded from the LSU and SSU residues in subsequent calculations.

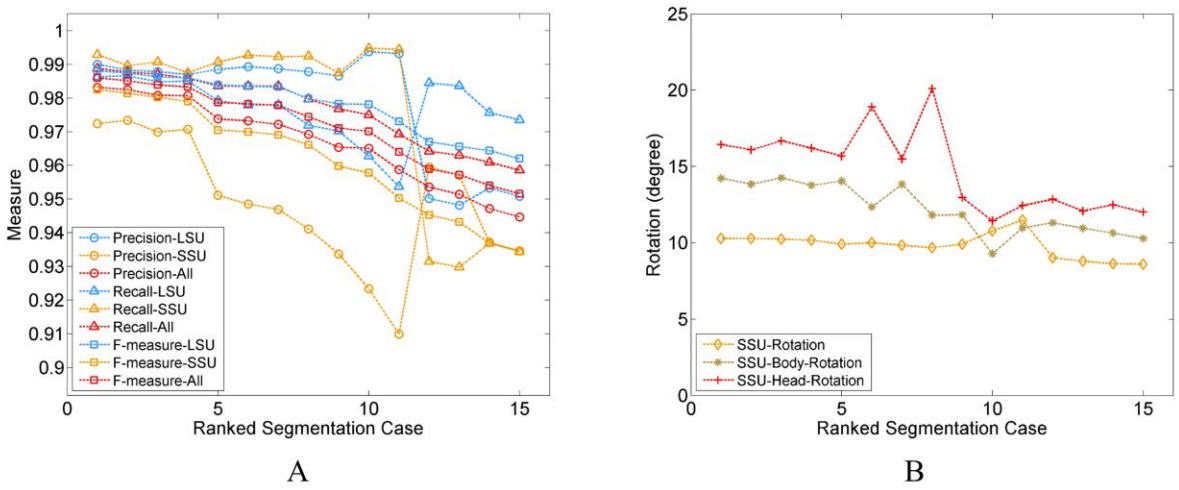

Figure S6. Characterizing the Error in Domain Segmentation and Domain Rotation Angle. **A.** Segmentation quality measures with $Precision$, $Recall$, $F-measure$ calculated for the individual domains and the full structure, in all the segmentation cases listed in Table S1. The plotted values are ranked based on the $F-measure$ of the SSU (Table S4). Across all the segmentation cases, the $Precision$ and $Recall$ values for the individual subunit seemed to vary more than the corresponding measures for the full structure and the $F-measure$. **B.** The rotation angle for SSU, SSU body and SSU head for the corresponding segmentation cases and the plotted angles are ranked in the same manner as in A.

Supporting Information Movies

Movie MS1. Fixed LSU with SSU body rotation from state "A" to "B."

Movie MS2. Fixed SSU Body with SSU head rotation from state "A" to "B."

Movie MS3. Fixed LSU with Full SSU rotation from state "A" to "B."

Supporting References

1. Rosenfel.A; Pfaltz, J. L., Sequential operations in digital picture processing. *J. Acm* **1966,** *13* (4), 471-&.
2. Coeurjolly, D.; Vacavant, A., Separable distance transformation and its applications. In *Digital geometry algorithms*, Springer: 2012; pp 189-214.
3. Bailey, D. G. In *An efficient euclidean distance transform*, International workshop on combinatorial image analysis, Springer: 2004; pp 394-408.
4. Fabbri, R.; Costa, L. D. F.; Torelli, J. C.; Bruno, O. M., 2d euclidean distance transform algorithms: A comparative survey. *ACM Comput. Surv.* **2008,** *40* (1), 1-44.
5. Felzenszwalb, P.; Huttenlocher, D. *Distance transforms of sampled functions*; Cornell University: 2004.
6. Danielsson, P. E., Euclidean distance mapping. *Comput. Graph. Img. Proc.* **1980,** *14* (3), 227-248.
7. Deza, M. M. D., Elena, Encyclopedia of distances. 3 ed.; Springer: Berlin, 2014.
8. Akleman, E.; Chen, J. N., Generalized distance functions. In *Shape Modeling International '99 - International Conference on Shape Modeling and Applications, Proceedings*, 1999; pp 72-79.
9. Sezgin, M.; Sankur, B., Survey over image thresholding techniques and quantitative performance evaluation. *J. Electron. Imaging.* **2004,** *13* (1), 146-168.
10. Goldstein, H., Classical mechanics. **1980**.
11. Dam, E. B.; Koch, M.; Lillholm, M., *Quaternions, interpolation and animation*. Datalogisk Institut, Københavns Universitet: 1998.
12. Pervin, E.; Webb, J. A., *Quaternions in computer vision and robotics*. Carnegie-Mellon University, Department of Computer Science: 1982.
13. Shoemake, K., Animating rotation with quaternion curves. *SIGGRAPH Computer Graphics* **1985,** *19* (3), 245-254.
14. Lovren, N.; Pieper, J. K., Error analysis of direction cosines and quaternion parameters techniques for aircraft attitude determination. *IEEE Trans. Aerosp. Electron. Syst.* **1998,** *34* (3), 983-989.

15. Horn, B. K. P., Closed-form solution of absolute orientation using unit quaternions. *J. Opt. Soc. Am. A* **1987,** *4* (4), 629-642.
16. Umeyama, S., Least-squares estimation of transformation parameters between two point patterns. *IEEE Trans. Pattern Anal. Mach. Intell.* **1991,** *13* (4), 376-380.
17. Baerlocher, P.; Boulic, R., Parametrization and range of motion of the ball-and-socket joint. In *Deformable avatars*, Springer: 2001; pp 180-190.
18. Shoemake, K., Fiber bundle twist reduction. In *Graphics gems iv*, Paul, S. H., Ed. Academic Press Professional, Inc.: 1994; pp 230-236.
19. Dobrowolski, P. Swing-twist decomposition in clifford algebra *ArXiv e-prints* [Online], 2015. https://arxiv.org/abs/1506.05481.
20. Hendriks, C. L.; Van Vliet, L.; Rieger, B.; van Kempen, G.; van Ginkel, M. Dipimage: A scientific image processing toolbox for matlab *Quantitative Imaging Group, Faculty of Applied Sciences, Delft University of Technology, Delft, The Netherlands* [Online], 1999.
21. Ludtke, S. J.; Baldwin, P. R.; Chiu, W., Eman: Semiautomated software for high-resolution single-particle reconstructions. *J. Struct. Biol.* **1999,** *128* (1), 82-97.
22. Grayson, P. *Orient*, 1.0; 2002.
23. Zhang, X. L.; Feng, X. Z.; Xiao, P. F.; He, G. J.; Zhu, L. J., Segmentation quality evaluation using region-based precision and recall measures for remote sensing images. *ISPRS J. Photogramm.* **2015,** *102*, 73-84.