HAL
open science

Rapid determination of RMSDs corresponding to macromolecular rigid body motions

Petr Popov, Sergei Grudinin

To cite this version:

Petr Popov, Sergei Grudinin. Rapid determination of RMSDs corresponding to macromolecular rigid body motions. Journal of Computational Chemistry, 2014, 35 (12), pp.950-956. 10.1002/jcc.23569 . hal-00952248

HAL Id: hal-00952248

https://hal.science/hal-00952248

Submitted on 26 Feb 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Rapid determination of RMSDs corresponding to macromolecular rigid body motions

Petr Popov[1,2] and Sergei Grudinin*[1,2]

[1]NANO-D, INRIA Grenoble – Rhone-Alpes, 38334 Saint Ismier Cedex, Montbonnot, France
[2]Laboratoire Jean Kuntzmann, B.P. 53, 38041 Grenoble Cedex 9, France

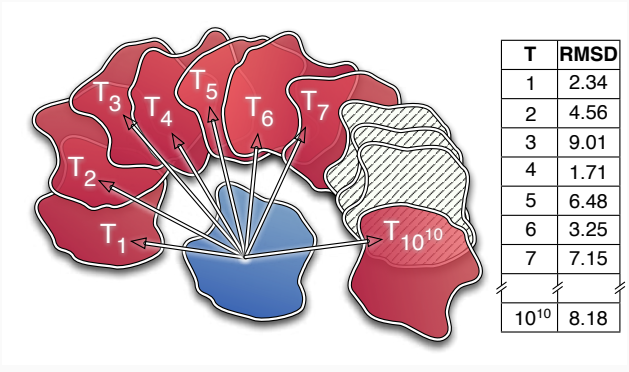
February 26, 2014

Abstract

Finding the root mean sum of squared deviations (RMSDs) between two coordinate vectors that correspond to the rigid body motion of a macromolecule is an important problem in structural bioinformatics, computational chemistry and molecular modeling. Standard algorithms compute the RMSD with time proportional to the number of atoms in the molecule. Here, we present *RigidRMSD*, a new algorithm that determines a set of RMSDs corresponding to a set of rigid body motions of a macromolecule in constant time with respect to the number of atoms in the molecule. Our algorithm is particularly useful for rigid body modeling applications such as rigid body docking, and also for high-throughput analysis of rigid body modeling and simulation results. We also introduce a constant-time rotation RMSD as a similarity measure for rigid molecules. A C++ implementation of our algorithm is available at `http://nano-d.inrialpes.fr/software/RigidRMSD`.

Keywords: RMSD calculation, rigid body motion, quaternion arithmetic, structure comparison, spatial transformations. ■

*sergei.grudinin@inria.fr



T	RMSD
1	2.34
2	4.56
3	9.01
4	1.71
5	6.48
6	3.25
7	7.15
⋮	⋮
10^{10}	8.18

Finding the root mean squared deviations (RMSDs) between two coordinate vectors that correspond to the rigid body motion of a macromolecule is an important problem in structural bioinformatics. We present a new algorithm that determines a set of RMSDs corresponding to a set of rigid body motions of a macromolecule in constant time with respect to the number of atoms in the molecule. Our algorithm is particularly useful for rigid body docking applications and for high-throughput analysis of rigid body modeling and simulation results.

INTRODUCTION

The root mean square deviation (RMSD) is a widely used and powerful criterion to estimate the similarity between two ordered sets of points. In structural biology and bioinformatics, RMSD has been widely accepted as a measure of similarity between macromolecules. For rigid body modeling applications such as rigid body molecular docking[1,2], rigid body molecular dynamics simulations[3,4], and rigid body Monte Carlo simulations[5], RMSD can be used as a measure of the rigid body motion of a molecule. However, determination of the RMSD can be a rate-limiting step for those applications where large number of rigid body motions should be compared. These applications range from conformation sampling in protein docking and structure-based drug design to high-throughput analysis of rigid body modeling and simulation results.

Much effort has been spent in developing algorithms for the optimal superposition of two molecules that minimizes the RMSD between the corresponding atoms[6–18]. In these methods, the squared RMSD is typically minimized with respect to the components of a rotation matrix or a rotation quaternion. However, in many applications of computational chemistry and structural bioinformatics a complementary problem emerges – given a set of rigid body motions of a reference molecule, compute the corresponding set of RMSDs. To the best of our knowledge, there exists no explicit description of an efficient algorithm for this problem in the literature. For the case of the RMSD between two positions of the same molecule after applying two spatial rigid body transformations, a formula can be found in the work of Rarey et al.[19], however, it contains an error, which we correct below. Here, we present *RigidRMSD*, a new algorithm for constant-time RMSD computations. In particular, we provide a connection between the RMSD and the axis and the angle of the rotation. Also, we consider rotations represented by both matrices and quaternions, since the two representations are widely used in the description of spatial transformations. We demonstrate that the quaternion representation could be more efficient than the matrix representation. Our algorithm initializes in time linear in the number of atoms in the molecule and then computes the RMSD corresponding to a rigid body motion in constant time. Our algorithm can be very useful when computing multiple RMSDs corresponding to a sequence of rigid body motions, as e.g. in the DockTrina method[20] or clustering applications, as each new RMSD computation takes only constant time. To demonstrate the efficiency of the *RigidRMSD* library, we implemented an RMSD-based clustering algorithm and compared it with the standard clustering method. Finally, we provide several source-code examples that demonstrate the usage of our library.

THEORETICAL FOUNDATION

Weighted RMSD

Given a set of N points $A = \{\mathbf{a}_i\}_N$ and $A' = \{\mathbf{a}'_i\}_N$ with associated weights $w = \{w_i\}_N$, the weighted RMSD between them is given as

$$\mathrm{RMSD}(A, A')^2 = \frac{1}{W} \sum_i w_i \left|\mathbf{a}_i - \mathbf{a}'_i\right|^2, \tag{1}$$

where $W = \sum_i w_i$. Here, $\{w_i\}_N$ are statistical weights that may emphasize the importance of a certain part of the structure, for example in case of a protein, the backbone or the side chains. These weights can be also equal to atomic masses (in this case W equals to the total mass of the molecule) or may be set to 1 (in this case $W = N$).

Quaternion arithmetic

A quaternion Q can be considered as a combination of a scalar s with a 3-component vector $\mathbf{q} = \{q_x, q_y, q_z\}^T$, $Q = [s, \mathbf{q}]$. The product of two quaternions $Q_1 = [s_1, \mathbf{q}_1]$ and $Q_2 = [s_2, \mathbf{q}_2]$ is a quaternion and can be expressed through a combination of scalar and vector products:

$$Q_1 \cdot Q_2 \equiv [s_1, \mathbf{q}_1] \cdot [s_2, \mathbf{q}_2] = [s_1 s_2 - (\mathbf{q}_1 \cdot \mathbf{q}_2), s_1 \mathbf{q}_2 + s_2 \mathbf{q}_1 + (\mathbf{q}_1 \times \mathbf{q}_2)]. \tag{2}$$

The squared norm of a quaternion Q is given as $|Q|^2 = s^2 + \mathbf{q} \cdot \mathbf{q}$, and a unit quaternion is a quaternion with its norm equal to 1. An inverse quaternion Q^{-1} is given as $Q^{-1} = [s, -\mathbf{q}]/|Q|^2$. A vector $\mathbf{v}$ can be treated as a quaternion with zero scalar component, $\mathbf{v} \equiv [0, \mathbf{v}]$. Then, a unit quaternion $\hat{Q}$ can be used to rotate vector $\mathbf{v}$ to a new position $\mathbf{v}'$ as follows

$$\left[0, \mathbf{v}'\right] = \hat{Q}\,[0, \mathbf{v}]\,\hat{Q}^{-1} = \left[0, (s^2 - \mathbf{q}^2)\mathbf{v} + 2s(\mathbf{q} \times \mathbf{v}) + 2(\mathbf{q} \cdot \mathbf{v})\mathbf{q}\right] = [0, \mathbf{v} + 2\mathbf{q} \times (\mathbf{q} \times \mathbf{v} + s\mathbf{v})]. \tag{3}$$

Equivalently, the same rotation can be represented with a rotation matrix $\mathbf{R}$, such that $\mathbf{v}' = \mathbf{R}\mathbf{v}$, where $\mathbf{R}$ can be expressed through the components of the quaternion $\hat{Q}$ as

$$\mathbf{R} = \begin{pmatrix} s^2 + q_x^2 - q_y^2 - q_z^2 & 2q_x q_y - 2sq_z & 2q_x q_z + 2sq_y \\ 2q_x q_y + 2sq_z & s^2 - q_x^2 + q_y^2 - q_z^2 & 2q_y q_z - 2sq_x \\ 2q_x q_z - 2sq_y & 2q_y q_z + 2sq_x & s^2 - q_x^2 - q_y^2 + q_z^2 \end{pmatrix}. \tag{4}$$

A unit quaternion $\hat{Q}$ corresponding to a rotation by an angle α around a unit axis $\mathbf{u}$ is given as $\hat{Q} = [\cos\frac{\alpha}{2}, \mathbf{u}\sin\frac{\alpha}{2}]$, and its inverse is $\hat{Q}^{-1} = [\cos\frac{\alpha}{2}, -\mathbf{u}\sin\frac{\alpha}{2}]$. Finally, N sequential rotations around different unit axes defined by unit quaternions $\{\hat{Q}_i\}_N$ result in a new vector $\mathbf{v}'$ according to

$$\left[0, \mathbf{v}'\right] = \hat{Q}_N \hat{Q}_{N-1} ... \hat{Q}_2 \hat{Q}_1 \left[0, \mathbf{v}\right] \hat{Q}_1^{-1} \hat{Q}_2^{-1} ... \hat{Q}_{N-1}^{-1} \hat{Q}_N^{-1}. \tag{5}$$

Rigid body motion described with quaternions

Let $\mathbf{R}$ be a rotation matrix and $\mathbf{T}$ - a translation vector applied to a molecule with N atoms at positions $A = \{\mathbf{a}_i\}_N$ with $\mathbf{a}_i = \{x_i, y_i, z_i\}^T$, such that the new positions $A' = \{\mathbf{a}'_i\}_N$ are given as $\mathbf{a}'_i = \mathbf{R}\mathbf{a}_i + \mathbf{T}$. Then, the weighted RMSD between A and A' is given as

$$\mathrm{RMSD}^2(A, A') = \frac{1}{W} \sum_i w_i \left|\mathbf{a}_i - \mathbf{R}\mathbf{a}_i - \mathbf{T}\right|^2. \tag{6}$$

We can rewrite the previous expression using quaternion representation of vectors $\mathbf{a}_i$ and $\mathbf{T}$ as

$$\mathrm{RMSD}^2 = \frac{1}{W} \sum_i w_i \left|[0, \mathbf{a}_i] - \hat{Q}[0, \mathbf{a}_i]\hat{Q}^{-1} - [0, \mathbf{T}]\right|^2. \tag{7}$$

Here, the unit quaternion $\hat{Q}$ corresponds to the rotation matrix $\mathbf{R}$. Since the norm of a quaternion does not change if we multiply it by a unit quaternion, we may right-multiply the kernel of the previous expression by $\hat{Q}$ to obtain

$$\mathrm{RMSD}^2 = \frac{1}{W} \sum_i w_i \left|[0, \mathbf{a}_i]\hat{Q} - \hat{Q}[0, a_i] - [0, \mathbf{T}]\hat{Q}\right|^2. \tag{8}$$

Using the scalar–vector representation of a quaternion, $\hat{Q} = [s, \mathbf{q}]$, we rewrite the previous RMSD expression as

$$\mathrm{RMSD}^2 = \frac{1}{W} \sum_i w_i \left[-\mathbf{q} \cdot \mathbf{T}, -s\mathbf{T} + (2\mathbf{a}_i - \mathbf{T}) \times \mathbf{q}\right]^2. \tag{9}$$

Performing scalar and vector products in Eq. (9), we obtain

$$\begin{aligned}
\mathrm{RMSD}^2 \;=\; & \frac{1}{W} \sum_i w_i \Big(\left[q_x T_x + q_y T_y + q_z T_z\right]^2 \\
+ \; & \left[-sT_x + q_y(2z_i - T_z) - q_z(2y_i - T_y)\right]^2 \\
+ \; & \left[-sT_y + q_z(2x_i - T_x) - q_x(2z_i - T_z)\right]^2 \\
+ \; & \left[-sT_z + q_x(2y_i - T_y) - q_y(2x_i - T_x)\right]^2 \Big).
\end{aligned} \tag{10}$$

Grouping terms in Eq. (10) that depend on atomic positions together, we obtain

$$\begin{aligned}
\mathrm{RMSD}^2 &= T_x^2 + T_y^2 + T_z^2 + \frac{4}{W}\sum_i w_i\{q_x^2(y_i^2+z_i^2) + q_y^2(x_i^2+z_i^2) + q_z^2(x_i^2+y_i^2) \\
&- 2q_xq_yx_iy_i - 2q_xq_zx_iz_i - 2q_yq_zz_iy_i\} \\
&+ \frac{4}{W}\left\{q_xq_zT_z + q_xq_yT_y - q_z^2T_x - q_y^2T_x + sq_zT_y - sq_yT_z\right\}\sum_i w_ix_i \\
&+ \frac{4}{W}\left\{q_yq_zT_z + q_xq_yT_x - q_x^2T_y - q_z^2T_y + sq_xT_z - sq_zT_x\right\}\sum_i w_iy_i \\
&+ \frac{4}{W}\left\{q_yq_zT_y + q_xq_zT_x - q_x^2T_z - q_y^2T_z + sq_yT_x - sq_xT_y\right\}\sum_i w_iz_i.
\end{aligned} \tag{11}$$

Introducing the inertia tensor $\mathbf{I}$, the rotation matrix $\mathbf{R}$, the center of mass vector $\mathbf{C}$, and the 3×3 identity matrix $\mathbf{E}_3$, we may simplify the previous expression to

$$\mathrm{RMSD}^2 = \mathbf{T}^2 + \frac{4}{W}\mathbf{q}^T\mathbf{I}\mathbf{q} + 2\mathbf{T}^T\left(\mathbf{R}-\mathbf{E}_3\right)\mathbf{C}, \tag{12}$$

where $\mathbf{C} = \frac{1}{W}\left\{\sum w_ix_i, \sum w_iy_i, \sum w_iz_i\right\}^T$, rotation matrix $\mathbf{R}$ corresponds to the rotation with the unit quaternion $\hat{Q}$ according to Eq. (4), and the inertia tensor $\mathbf{I}$ is given as

$$\mathbf{I} = \begin{pmatrix} \sum w_i(y_i^2+z_i^2) & -\sum w_ix_iy_i & -\sum w_ix_iz_i \\ -\sum w_ix_iy_i & \sum w_i(x_i^2+z_i^2) & -\sum w_iy_iz_i \\ -\sum w_ix_iz_i & -\sum w_iy_iz_i & \sum w_i(x_i^2+y_i^2) \end{pmatrix}. \tag{13}$$

Equation (12) is the principal result of this work. It consists of three parts, the pure translational contribution $\mathbf{T}^2$, the pure rotational contribution $\frac{4}{W}\mathbf{q}^T\mathbf{I}\mathbf{q}$, and the cross–term $2\mathbf{T}^T\left(\mathbf{R}-\mathbf{E}_3\right)\mathbf{C}$. In this equation, only two variables depend on the atomic positions $\{\mathbf{a}_i\}_N$, the inertia tensor $\mathbf{I}$, and the center of mass vector $\mathbf{C}$. Below, we will use this fact when computing RMSDs for a set of rigid body motions.

RMSD corresponding to a pure rotation

An interesting consequence of Eq. (12) is the analytical expression of the RMSD for a pure rigid rotation. Recall that a unit quaternion in Eq. (12) can be represented as a rotation about a unit axis $\mathbf{n}$ by an angle α, $\hat{\mathbf{Q}} = [\cos\frac{\alpha}{2}, \mathbf{n}\sin\frac{\alpha}{2}]$. Then, if a rigid molecule is rotated about this axis passing through the origin, the RMSD for such a rotation is given as

$$\mathrm{RMSD}^2 = \frac{4}{W}\sin^2\frac{\alpha}{2}I(\mathbf{n}), \tag{14}$$

where $I(\mathbf{n})$ is the reduction of the inertia tensor (13) to a scalar form for the unit axis $\mathbf{n}$:

$$I(\mathbf{n}) = \mathbf{n}^T\mathbf{I}\mathbf{n}. \tag{15}$$

Rigid body motion described with a rotation matrix

The pure rotational contribution $\frac{4}{W}\mathbf{q}^T\mathbf{I}\mathbf{q}$ in Eq. (12) can be rewritten in terms of a rotation matrix $\mathbf{R}$ as

$$\frac{4}{W}\mathbf{q}^T\mathbf{I}\mathbf{q} = \frac{4}{W}\mathrm{tr}\left(\left(\mathbf{q}\mathbf{q}^T\right)\mathbf{I}\right) = \frac{1}{W}\mathrm{tr}\left(\mathbf{I}\right)\left[1-\mathrm{tr}\left(\mathbf{R}\right)\right] + \frac{2}{W}\mathrm{tr}\left(\mathbf{I}\mathbf{R}\right). \tag{16}$$

Here, rotation matrix $\mathbf{R}$ is connected with the vector part of the rotation quaternion $\mathbf{q}$ by Eq. (4). Equivalently, Eq. (16) can be written as

$$\frac{4}{W}\mathbf{q}^T\mathbf{I}\mathbf{q} = \frac{2}{W}\sum_{i,j=1}^{3}\left(\delta_{ij}-R_{ij}\right)X_{ij}, \tag{17}$$

where matrix $\mathbf{X}$ is given as

$$\mathbf{X} = \begin{pmatrix} \sum w_i x_i^2 & \sum w_i x_i y_i & \sum w_i x_i z_i \\ \sum w_i x_i y_i & \sum w_i y_i^2 & \sum w_i y_i z_i \\ \sum w_i x_i z_i & \sum w_i y_i z_i & \sum w_i z_i^2 \end{pmatrix}. \tag{18}$$

Now, the weighted RMSD in Eq. (12) can be computed using the matrix description of the rotation:

$$\mathrm{RMSD}^2 = \mathbf{T}^2 + \frac{2}{W}\sum_{i,j=1}^{3}\left(\delta_{ij}-R_{ij}\right)X_{ij} + 2\mathbf{T}^T\left(\mathbf{R}-\mathbf{E}_3\right)\mathbf{C}. \tag{19}$$

RMSD corresponding to a relative rigid body motion

Let $\mathbf{R}_1$ and $\mathbf{R}_2$ be two rotation matrices and $\mathbf{T}_1$ and $\mathbf{T}_2$ – two translation vectors applied to a molecule with N atoms at positions $A = \{\mathbf{a}_i\}_N$, such that new positions $A_1 = \{\mathbf{a}_i^1\}_N$ and $A_2 = \{\mathbf{a}_i^2\}_N$ are given as $\mathbf{a}_i^1 = \mathbf{R}_1\mathbf{a}_i + \mathbf{T}_1$ and $\mathbf{a}_i^2 = \mathbf{R}_2\mathbf{a}_i + \mathbf{T}_2$. Let a unit quaternion $\hat{Q} = [s, \mathbf{q}]$ correspond to the relative rotation $\mathbf{R}_2^T\mathbf{R}_1$. Then, the weighted RMSD between positions A_1 and A_2 is given by a generalized version of Eq. (12) as

$$\mathrm{RMSD}^2(A_1, A_2) = \frac{4}{W}\mathbf{q}^T\mathbf{I}\mathbf{q} + \left(\mathbf{T}_1-\mathbf{T}_2\right)^2 + 2\left(\mathbf{T}_1-\mathbf{T}_2\right)^T\left(\mathbf{R}_1-\mathbf{R}_2\right)\mathbf{C}. \tag{20}$$

Using Eq. (17) we can rewrite the above equation using the matrix description of the rotation:

$$\mathrm{RMSD}^2(A_1, A_2) = \frac{2}{W}\sum_{i,j=1}^{3}\left(\delta_{ij}-\sum_{k=1}^{3}R_{ki}^1R_{kj}^2\right)X_{ij} + \left(\mathbf{T}_1-\mathbf{T}_2\right)^2 + 2\left(\mathbf{T}_1-\mathbf{T}_2\right)^T\left(\mathbf{R}_1-\mathbf{R}_2\right)\mathbf{C}. \tag{21}$$

The derived equation is equivalent to the formula obtained by Rarey et al. for clustering spatial motions in the FlexX docking tool[19], except that the formula of Rarey et al. contains an error in the rotational part. More precisely, it has an additional factor 2 preceding the $\sum_{k=1}^{3}R_{ki}^1R_{kj}^2$ term.

ALGORITHM IMPLEMENTATION

Computational considerations

In the above equations (12–21), as we have mentioned above, only two variables depend on the atomic positions of the reference molecular structure – the inertia tensor $\mathbf{I}$ (or, its equivalent matrix $\mathbf{X}$ if the rotation is given by the matrix representation), and the center of mass vector $\mathbf{C}$. Therefore, given a set of M spatial transformations, we compute these two variables only once at the initialization step. The computational complexity of this step is linear with respect to the number of atoms N in the molecule. After, each RMSD computation for a single spatial transformation takes only constant time. The total cost to compute M RMSD values for a rigid molecule with N atoms thus will be $O(N+M)$, which is usually much smaller compared to the cost of standard algorithms, $O(NM)$, particularly at large values of M and N. More precisely, a standard algorithm computes the RMSD for each spatial transformation in $O(N)$ operations according to Eq. (1), thus resulting in $O(NM)$ overall complexity for M spatial transformations. Below we discuss computational strategies that allow to reduce the constant in $O(N+M)$.

In Eq. (12), the cross–term vanishes in the reference frame bound to the center of mass (COM) of the molecule where $\mathbf{C}=\mathbf{0}$. In this reference frame, the rotation is preserved, while the translation $\mathbf{T}_{\mathrm{COM}}$ is given as

$$\mathbf{T}_{\mathrm{COM}} = \mathbf{R}\mathbf{C} + \mathbf{T} - \mathbf{C}. \tag{22}$$

We can equivalently obtain the translation in the COM reference frame using a rotation quaternion $\hat{Q}$ as

$$\mathbf{T}_{\mathrm{COM}} = \hat{Q}\mathbf{C}\hat{Q}^{-1} + \mathbf{T} - \mathbf{C}. \tag{23}$$

Therefore, in the COM reference frame, the RMSD can be computed with fewer arithmetic operations. More precisely, using the quaternion representation of the rotation, the RMSD is given as

$$\mathrm{RMSD}^2 = \mathbf{T}_{\mathrm{COM}}^2 + \frac{4}{W}\mathbf{q}^T\mathbf{I}_{\mathrm{COM}}\mathbf{q}. \tag{24}$$

Similarly, if we use the matrix representation of the rotation, the RMSD is given as

$$\mathrm{RMSD}^2 = \mathbf{T}_{\mathrm{COM}}^2 + \frac{2}{W}\sum_{i,j=1}^{3}\left(\delta_{ij} - R_{ij}\right)X_{ij}^{\mathrm{COM}}. \tag{25}$$

In the above equations, inertia tensor $\mathbf{I}_{\mathrm{COM}}$ and matrix $\mathbf{X}^{\mathrm{COM}}$ are computed in the COM coordinate system. A particularly interesting case is the computation of the RMSD in the principal axes of inertia (PAI) frame. The PAI frame is the coordinate system where the centre of mass vector $\mathbf{C}=\mathbf{0}$ and the molecule is aligned along its principal axes, i.e. matrices $\mathbf{I}_{\mathrm{COM}}$ and $\mathbf{X}^{\mathrm{COM}}$ are diagonal. In this frame, equations (24-25) are simpler. Also, in the PAI frame, RMSD corresponding to a

relative rigid body motion defined by two rotation quaternions $\hat{Q}_1$ and $\hat{Q}_2$ and two translation vectors $\mathbf{T}_1$ and $\mathbf{T}_2$ will be

$$\begin{aligned}\mathrm{RMSD}^2(A_1, A_2) =& \frac{4}{W}\big(\left(s_1 q_2^x - q_1^x s_2 - q_1^y q_2^z + q_1^z q_2^y\right)^2 I_{xx} + \left(s_1 q_2^y - q_1^y s_2 - q_1^z q_2^x + q_1^x q_2^z\right)^2 I_{yy} + \\ & \left(s_1 q_2^z - q_1^z s_2 - q_1^x q_2^y + q_1^y q_2^x\right)^2 I_{zz}\big) + \left(\mathbf{T}_1 - \mathbf{T}_2\right)^2 . \end{aligned} \tag{26}$$

This equation uses three times fewer arithmetic operations compared to the previously published Eq. (21). More precisely, Eq. (26) requires only 38 arithmetic operations compared to 114 operations in Eq. (21). Generally, Eqs. (22-26) are more efficient in the number of arithmetic operations compared to Eqs. (12) and (21), as it is summarized in Table 1. This table lists the number of arithmetic operations needed to compute the squared RMSD using different representations of the rigid motion in three different coordinate systems, the world frame, the COM frame, and the PAI frame. As listed in Table 1, to compute the squared RMSD we need 54 arithmetic operations in the worst case, when the rigid rotation is given as a quaternion in the world frame. If we choose the coordinate system properly (the PAI frame), we can compute the squared RMSD in just 14 operations. Table 1 demonstrates that in the world frame one requires a fewer number of arithmetic operations to compute the RMSD if rotations are represented with rotation matrices, whereas in the COM and PAI frames the number of operations is equal between the two representations. However, when performing sequences of rotations, the quaternion representation is more numerically stable and computationally efficient compared to the matrix representation irrespective of the choice of coordinate system. Indeed, one requires 45 arithmetic operations to multiply two rotation matrices, whereas quaternion multiplication requires only 28 operations. Finally, Table 1 demonstrates that the squared RMSD for a relative rigid motion computed with the quaternion representation in the PAI frame requires three times fewer operations compared to the one computed with the matrix representation in the world frame.

Numerical tests

Throughout the article, we count the number of arithmetic operations in different equations according to the source code of the *RigidRMSD* library. We would like to mention that the cost of different arithmetic operations is not the same – division and square root are usually more expensive than multiplication, which is in turn more expensive than addition and subtraction[21]. We should also mention that on modern computers minimizing the number of arithmetic operations is less important for the performance of a particular algorithm compared to increasing the amount of instruction level parallelism or improving memory access patterns and cache utilization, for example. Therefore, it is impossible to rigorously compare the performance of different algorithms solely based on their operation count. Thus, we only provide the total number of arithmetic operations as a rough estimation of the complexity of the equations and the corresponding algorithms. To get more practical numbers, in the following sections we run a series of tests with two different levels

of compiler optimization.

We implemented the tests using the C++ programming language and compiled them using g++ compiler version 4.6 with optimization levels –O0 and –O3. For the gcc family of compilers, optimization option –O0 disables compiler optimization, whereas optimization option –O3 enables heavy optimization including interprocedural optimization and vectorization. We ran the tests on a 64-bit Linux Fedora operating system with Intel(R) Xeon(R) CPU X5650 @ 2.67GHz.

RESULTS AND DISCUSSION

This section presents numerical tests and practical applications of the equations derived in this article. First, we compare the quaternion representation with the matrix representation when computing sequential rotations (i.e. a composition of several rotations) and when computing a product of rotations with the subsequent RMSD computation. Second, we discuss the similarity measure between molecules and demonstrate that the rotation RMSD (Eq. 14) can be advantageous over a simpler angular distance measure. Finally, we present a rigid body clustering algorithm as an example of the application of the derived equations.

Rotation representation

Quaternions provide another way to represent rotations compared to conventional rotation matrices. In practice, the quaternion representation has several benefits over the matrix representation. First, a quaternion compared to a matrix requires less storage, four values versus nine. Second, the orthonormalization of a quaternion costs much less than the orthogonalization of a matrix. More precisely, orthonormalization of a quaternion can be accomplished by dividing the quaternion by its norm, which requires twelve arithmetic operations including one square root. However, there is no universal method for matrix orthonormalization. In this case, one may use the Gram-Schmidt orthonormalization method, QR decomposition, singular value decomposition or other methods, which are more computationally expensive compared to the quaternion orthonormalization[22]. Third, a product of two rotations using quaternions requires fewer arithmetic operations compared to the matrix representation (28 versus 45). Finally, the matrix multiplication is less numerically stable due to the accumulation of rounding errors. In summary, applications that require sequential rotations (e.g. some docking applications) will gain in speed, memory and numerical precision when using the quaternion representation.

To demonstrate the numerical efficiency of the quaternion representation, we ran a series of tests with two different levels of compiler optimization. In the first test, we performed 10^8 products of rotations using the two types of rotation representation and compared the timing for a single product of rotations with and without compiler optimization. The results of this test are presented in Table 2. We see that a rotation with quaternions is about 60% faster compared to a rotation with matrices regardless of the optimization level. In the second test, we computed a product of

two rotations with the subsequent RMSD computation using Eqs. (22)–(25) and repeated these operations 10^8 times. Then, we calculated the time required for a single product of rotations with the subsequent RMSD computation. The results of this test are also presented in Table 2. Again, the quaternion representation is about 10% faster without optimization and 4% faster with optimization compared to the matrix representation. We should note that increasing the number of sequential rotations will provide a bigger speedup using the quaternion representation in this example. In the third test, we computed 10^8 RMSDs corresponding to a relative rigid body motion using the matrix representation of rotation (Eq. (21)) and the quaternion representation of rotation (Eq. (26)). We can see that our quaternion approach is 2.4–3.2 times faster compared to the matrix formula (Eq. (21)) depending on the level of compiler optimization.

To summarize, if a particular application operates with sequential rotations, as it happens in the DockTrina algorithm[20] or other docking applications, RMSD computations are more numerically efficient using the quaternion representation. Furthermore, the gain of using the quaternion representation is bigger up to 60 % when using a larger sequence of rotations.

Rotation RMSD as a similarity measure for molecular structures

It is still an open question how to measure the similarity between structures of a molecular complex[23]. For example, Rodrigues et al[24] developed a clustering method with the similarity measure based on the fraction of common contacts between two complexes. Another similarity measure was recently proposed by Vreven et al.[25], where the angular distance computed in constant–time is used as the criterion for the similarity between the predictions from rigid body docking. Nonetheless, the majority of the algorithms in the structural bioinformatics use the pair-wise RMSD as the similarity metric between the molecular structures.

Equation (14) is of particular interest when considered in relation to the aforementioned work of Vreven et al.[25], where the authors demonstrated that the angular distance can serve as a similarity measure for rigid molecules as an alternative for the RMSD. More precisely, they defined the angular distance as the angle between the rotations corresponding to two docking predictions, ignoring the translational degrees of freedom. Vreven et al. claimed that the drawback of using the RMSD is that it is computationally expensive. However, we demonstrated that the RMSD can also be computed in constant time. Furthermore, in the context of Eq. (14), the angular distance is simply equal to the rotation angle α and does not take into account the geometry of a rigid molecule. In particular, for a fixed rotation angle, the angular distance for molecules of different size will be equal, while the RMSD can be very different. Another example that demonstrates the difference between the two measures is the rotation of a long linear molecule. The RMSD for such a rotation will dramatically depend on the axis of the rotation, while the angular distance will be the same regardless the rotation axis.

To conclude, we would like to emphasize that for comparison of rigid molecules of different size or molecules of non-spherical shape, it may be more rigorous to use the similarity measure defined

by Eq. (14) instead of the angular distance. Particularly, our measure involves the scalar form of the inertia tensor (Eq. 15), thus taking into account the geometry and the rotation axis of the molecules.

Clustering algorithm

One of the possible applications of the *RigidRMSD* library can be the rigid body clustering. Molecular docking algorithms typically produce thousands of solutions, some of them having a very similar geometry. Therefore, it is practical to group these into clusters. As we have discussed above, there are multiple ways to measure the similarity between molecular structures[23], however, most of the modern state-of-the-art clustering algorithms use the pair-wise RMSD as the similarity metric between the predictions, as it is implemented, e.g., in the Hex[1] and ZDOCK[2] docking algorithms. In the worst case, the complexity of such a clustering algorithm can be quadratic with respect to the number of docking predictions. Thus, an efficient pair-wise RMSD test can dramatically improve the clustering performance. The clustering algorithm used by the Hex and ZDOCK applications consists of the following steps. First, the docking prediction with the best score (yet unassigned to any cluster) is taken as the seed for the new cluster. Second, the pair-wise RMSDs between the seed and all other predictions (in case of ZDOCK) or some best predictions (in case of Hex) are measured and the predictions with the RMSD lower than a certain threshold are put into the cluster. Finally, these two steps are iterated until all docking predictions are assigned to corresponding clusters.

In order to demonstrate the efficiency of the *RigidRMSD* library, we compared the clustering algorithm implemented with our library to the one from the Hex software. We chose Hex for the comparison because it is a very fast rigid body docking tool and also because it explicitly provides the clustering time. It is worth to note that Hex's clustering algorithm has linear complexity with respect to the number of docking predictions, i.e. it is faster (though less accurate) than the standard RMSD-based clustering algorithms, as it is implemented in ZDOCK. Both Hex and ZDOCK clustering algorithms use the standard RMSD test linear in the number of atoms in the protein.

For the comparison, we collected a benchmark of 23 protein dimers of various size (see Table S1 in SI). After, we launched Hex version 6.3 on this benchmark and collected docking solutions before clustering, sizes of clusters, and clustering time. We then also clustered these solutions using the *RigidRMSD* library. Figure 1 shows the clustering time of the HEX clustering algorithm with respect to our clustering using Eqs. (21) and (26) as a function of the number of atoms in the smaller protein (left) and the number of docking solutions before the clustering (right). We can clearly see that our implementation of the clustering algorithm is more than an order of magnitude faster compared to the Hex implementation. Also, the quaternion representation of rotation, Eq. (26), is on average three times more efficient compared to the matrix representation, Eq. (21). The efficiency of our clustering algorithm increases when using a larger RMSD threshold, as it is shown in Fig. S1 from SI. Also, mean cluster sizes obtained with our clustering algorithm are significantly

larger compared to the Hex clustering (see Fig. S1 from SI), particularly at large RMSD thresholds. This demonstrates that our implementation of the clustering algorithm is not only much faster, but also more accurate compared to the clustering in Hex, especially at large clustering thresholds.

CONCLUSIONS

We described a very fast and efficient way to compute the RMSD corresponding to the set of rigid body motions of a molecule. Our algorithm consists of an initialisation step followed by a series of constant-time RMSD computations. The initialization step has linear complexity with respect to the number of atoms in a molecule. However, each of the RMSD calculations requires only 14 to 54 arithmetic operations when using a single rigid motion (i.e. given with a single spatial rigid body transformation), or 38 to 114 arithmetic operations when using a relative rigid motion (i.e. given with a pair of spatial rigid body transformations), depending on the representation of the motion and the choice of the coordinate frame. This can be compared to 30 arithmetic operations needed to rotate a vector using a quaternion or 15 arithmetic operations needed to rotate a vector using a rotation matrix. We demonstrated that RMSD computations are more numerically efficient when using the quaternion representation of rotation. In particular, the gain of using the quaternion representation is bigger when using a larger sequence of rotations. We have also discussed two ways to measure the similarity between structures of a molecular complex. In particular, we claim that it may be more rigorous to use the rotation RMSD similarity measure defined by Eq. (14) instead of the simpler measure based on the angular distance. As an application of the *RigidRMSD* library, we implemented a clustering algorithm for solutions obtained with rigid body molecular docking tools. We showed that our implementation is more than one order of magnitude faster and also more accurate compared to the standard clustering algorithm used in the popular Hex docking software. A C++ implementation of our algorithm is available at `http://nano-d.inrialpes.fr/software/RigidRMSD` or by request from the authors.

ACKNOWLEDGMENTS

This work was supported by the Agence Nationale de la Recherche (ANR-2010-JCJC-0206-01 and ANR-11-MONU-006-01).

References

1. D. W. Ritchie and G. J. Kemp, Proteins Struct. Funct. Bioinf. **39**, 178 (2000).
2. R. Chen, L. Li, and Z. Weng, Proteins Struct. Funct. Bioinf. **52**, 80 (2003).
3. D. Evans and S. Murad, Mol. Phys. **34**, 327 (1977).
4. A. Mascioni, C. Karim, J. Zamoon, D. D. Thomas, and G. Veglia, J. Am. Chem. Soc. **124**, 9392 (2002).
5. I. Tunbridge, R. Best, J. Gain, and M. Kuttel, J. Chem. Theory Comput. **6**, 3588 (2010).
6. W. Kabsch, Acta Crystallogr., Sect. A: Found. Crystallogr. **32**, 922 (1976).
7. D. R. Ferro and J. Hermans, Acta Crystallogr., Sect. A: Found. Crystallogr. **33**, 345 (1977).
8. A. D. McLachlan, Acta Crystallogr., Sect. A: Found. Crystallogr. **38**, 871 (1982).
9. A. M. Lesk, Acta Crystallogr., Sect. A: Found. Crystallogr. **42**, 110 (1986).
10. B. K. Horn, J. Opt. Soc. Am. A. **4**, 629 (1987).
11. R. Diamond, Acta Crystallogr., Sect. A: Found. Crystallogr. **44**, 211 (1988).
12. B. K. Horn, H. M. Hilden, and S. Negahdaripour, J. Opt. Soc. Am. A. **5**, 1127 (1988).
13. S. K. Kearsley, Acta Crystallogr., Sect. A: Found. Crystallogr. **45**, 208 (1989).
14. G. R. Kneller, Mol. Simul. **7**, 113 (1991).
15. D. L. Theobald, Acta Crystallogr., Sect. A: Found. Crystallogr. **61**, 478 (2005).
16. E. A. Coutsias, C. Seok, and K. A. Dill, J. Comput. Chem. **25**, 1849 (2004).
17. C. F. Karney, J. Mol. Graphics Modell. **25**, 595 (2007).
18. P. Liu, D. K. Agrafiotis, and D. L. Theobald, J. Comput. Chem. **31**, 1561 (2010).
19. M. Rarey, S. Wefing, and T. Lengauer, J. Comput.-Aided Mol. Des. **10**, 41 (1996).
20. P. Popov, D. W. Ritchie, and S. Grudinin, Proteins Struct. Funct. Bioinf. **82**, 34 (2014).
21. R. P. Brent and P. Zimmermann, *Modern computer arithmetic* (Cambridge University Press, 2010).
22. G. H. Golub and C. F. V. Loan, *Matrix Computations* (JHU Press, 2012).
23. S. Wallin, J. Farwer, and U. Bastolla, Proteins Struct. Funct. Bioinf. **50**, 144 (2003).

24. J. P. G. L. M. Rodrigues, M. Trellet, C. Schmitz, P. Kastritis, E. Karaca, A. S. J. Melquiond, and A. M. J. J. Bonvin, Proteins Struct. Funct. Bioinf. **80**, 1810 (2012).

25. T. Vreven, H. Hwang, and Z. Weng, PLoS One **8**, e56645 (2013).

Table 1: Number of arithmetic operations for the RMSD calculations with respect to different rotation representations and a different choice of the coordinate frame. These numbers were computed according to the source code of the *RigidRMSD* library. The references to the corresponding equations are given in the last column. These equations comprise only multiplication and addition / subtraction arithmetic operations.

	Multiplies	Add/Subtract	Total	Equation
RMSD^2 (quaternion, world frame)	34	20	54	(24) and (23)
RMSD^2 (matrix, world frame)	19	26	45	(25) and (22)
RMSD^2 (quaternion, COM frame)	16	8	24	(24)
RMSD^2 (matrix, COM frame)	10	14	24	(25)
RMSD^2 (quaternion, PAI frame)	9	5	14	(24), $\mathbf{I}_{\text{COM}}$ is diagonal
RMSD^2 (matrix, PAI frame)	6	8	14	(25), $\mathbf{X}^{\text{COM}}$ is diagonal
RMSD^2 for clustering, (matrix, world frame)	55	59	114	(21)
RMSD^2 for clustering, (quaternion, PAI frame)	21	17	38	(26)

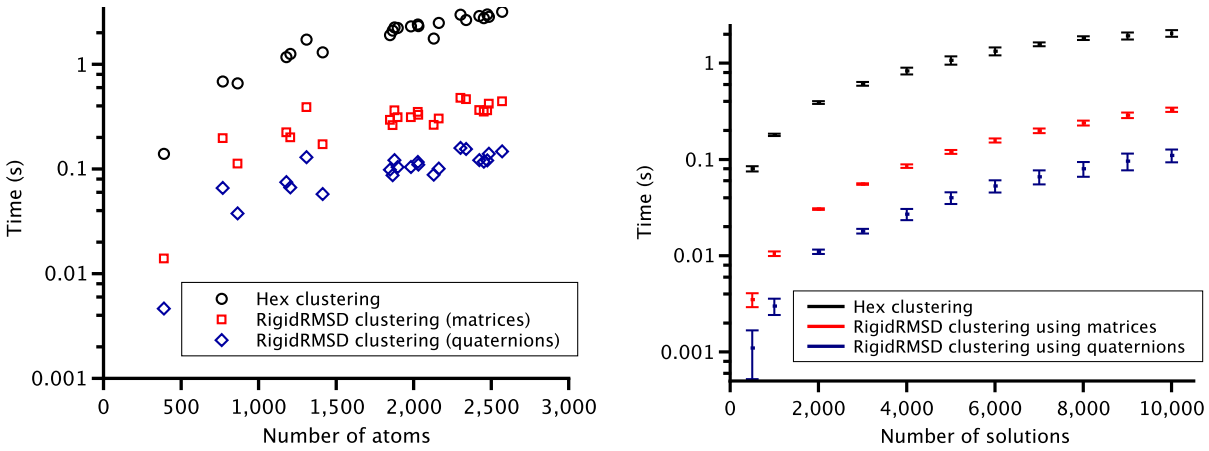


Figure 1: Left: Time spent on clustering docking solutions by Hex and *RigidRMSD* with respect to the number of atoms in the ligand protein. Each point on the plot corresponds to a protein complex from the protein benchmark (see Table S1 in SI). For each protein complex, the number of considered docking solutions was fixed to 10,000. Right: Average time spent on clustering docking solutions by Hex and *RigidRMSD* with respect to the number of docking solutions. For this plot, we chose five structures with the number of atoms in the ligand protein of about 2,000 such that they result in a similar number of clusters and plotted the standard deviation of the clustering time for these structures. For both plots, time is plotted on a logarithmic scale and the clustering RMSD threshold is fixed to 10.0 Å.

Table 2: Running time for three tests using two levels of compiler optimization. O0 optimization level disables optimization, whereas O3 optimization level enables heavy optimization including interprocedural optimization and vectorization. In the first test (columns 1 and 2), we performed 10^8 products of rotations using the two types of rotation representation and reported the timing for a single product of rotations. In the second test (columns 3 and 4), we computed a product of two rotations with the subsequent RMSD computation using Eqs. (22)–(25) and repeated these operations 10^8 times for averaging. In the last test (columns 5 and 6), we computed 10^8 RMSDs corresponding to a relative rigid body motion, as in the clustering application, using the matrix representation of rotation (Eq. 21) and the quaternion representation of rotation (Eq. 26) and reported the timing for a single RMSD calculation.

	Product of Rotations (–O0)	Product of Rotations (–O3)	Product of rotations and RMSD (–O0)	Product of rotations and RMSD (–O3)	Clustering (–O0)	Clustering (–O3)
Quaternion representation	2.96×10^{-8} s	0.73×10^{-8} s	7.79×10^{-8} s	2.29×10^{-8} s	4.17×10^{-8} s	1.19×10^{-8} s
Matrix representation	4.68×10^{-8} s	1.18×10^{-8} s	8.55×10^{-8} s	2.39×10^{-8} s	9.99×10^{-8} s	3.81×10^{-8} s