On Updating Torsion Angles of Molecular Conformations

Vicky Choi*

Department of Computer Science, Virginia Tech, 660 McBryde Hall, Blacksburg, Virginia 24061-0106

Received June 20, 2005

A conformation of a molecule is defined by the relative positions of atoms and by the chirality of asymmetric atoms in the molecule. The three main representations for conformations of molecules are Cartesian coordinates, a distance geometry descriptor (which consists of a distance matrix and the signs of the volumes of quadruples of atoms), and internal coordinates. In biochemistry, conformational changes of a molecule are usually described in terms of internal coordinates. However, for many applications, such as molecular docking, the Cartesian coordinates of atoms are needed for computation. Although, for each conformational change, the Cartesian coordinates of atoms can be updated in linear time (which is optimal asymptotically), the constant factor becomes significant if a large number of updates are needed. Zhang and Kavraki (*J. Chem. Inf. Comput. Sci.* **2002**, *42*, 64−70) examined three methods: the simple rotations, the Denavit−Hartenberg local frames, and the atom-group local frames. On the basis of their implementations, they showed that the atom-group local frames are more efficient than the other two. In this paper, by expressing the torsion-angle change as a composition of translations and rotations, we observe that the simple rotations can be implemented in an efficient way by taking advantage of consecutive operations. Both quantitative and experimental comparisons show that the improved simple rotations, in which rotations are expressed in unit quaternions, are as efficient as the atom-group local frames and, thus, have the advantage of avoiding the need of precomputations of a set of local frames and transformations between them.

1. INTRODUCTION

One major challenge in computing structures of molecules is that flexibility leads to many conformations. The three main representations for a conformation of a molecule are Cartesian coordinates, a distance geometry descriptor,[1,2] and internal coordinates. In Cartesian coordinates, each atom center is specified by *x*, *y*, and *z* coordinates. Many molecular file formats, such as .pdb or .mol2, use this representation. In a distance geometry description, one represents the relative atomic positions by all pairwise distances among atoms in a distance matrix and the chiralities by the signs (+1, −1, or 0) of the volumes of ordered quadruples of atoms. A chemically more-intuitive representation is given by internal coordinates: bond lengths, bond angles, and torsion angles. A Z-matrix[3,4] is an example of a molecule represented in internal coordinates.

Interconversion of these three different representations is essential for molecular modeling. One can easily convert Cartesian coordinates into a distance geometry descriptor or internal coordinates. A distance matrix can also be easily converted into Cartesian coordinates in linear time,[5] followed by reflection operations to correct the chiralities. However, converting internal coordinates into Cartesian coordinates or a distance matrix is inherently more complicated than the reverse.

In biochemical applications, conformational changes of a molecule are usually described in terms of its internal coordinates. Bond lengths and bond angles are usually fixed at the ideal values because changes in these parameters are small and also because this restriction improves computational efficiency. (We maintain this restriction throughout this paper.) If the molecule is represented by a Z-matrix, then the changes in torsion angles can be updated directly. However, for many other applications, such as molecular dynamics[6] and molecular docking, the Cartesian coordinates of atoms are required for computation. Consequently, algorithms for efficiently updating conformations in Cartesian coordinates (given changes in torsion angles) will accelerate the current programs for these applications.

For each conformational change, the Cartesian coordinates of atoms can be updated in linear time (which is optimal asymptotically). However, the constant factor that is hidden in the asymptotic time complexity becomes significant when a large number of updates are needed. This occurs for most biochemical applications, which require energy minimizations and conformational searches.

Zhang and Kavraki[8] examined three methods: the simple rotations, the Denavit−Hartenberg local frames, and the atom-group local frames. (The simple rotations method is a global reference-frame method, while the other two, as their names suggest, are local-frames methods.) They used formulas—rotation matrices and homogeneous matrices from robotics and graphics[9,10]—to express torsion-angle changes. They showed, both quantitatively and experimentally, that the atom-group local-frames method, in which a set of local frames and transformations between consecutive local frames can be precomputed, was more efficient than the simple rotations method and the Denavit−Hartenberg local-frames method.

In this paper, we express each torsion-angle change in terms of primitive operations: translations and rotations. With this expression, we observe that the simple rotations

* Corresponding author tel: 1-(540)-231-2919; e-mail: vchoi@cs.vt.edu.

10.1021/ci050253h CCC: $33.50
Published on Web 11/30/2005

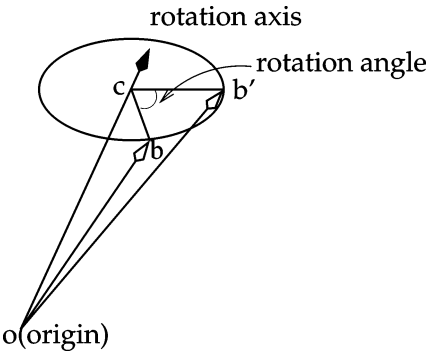

Figure 1. A rotation in three-dimensional space can be viewed as a rotation about an axis $v = \vec{oc}$ by an angle θ. In this figure, b' is the image of b after rotation.

can be implemented in an efficient way by taking advantage of consecutive operations. Both quantitative and experimental results show that the improved simple rotations (in which rotations are expressed in unit quaternions) are as efficient as the atom-group local frames. Therefore, the simple rotations method avoids the need of the relatively costly (compared with the total cost needed to update the molecule) precomputation of a set of local frames and transformations for changing consecutive frames.

2. REVIEW OF ROTATION AND RIGID MOTION

In this section, we recall the definition of rotation and rigid motion in $\mathbb{R}^d$, in particular for $d = 3$. A transformation from $\mathbb{R}^d$ to $\mathbb{R}^d$ that preserves distances is called an isometry. An orientation-preserving isometry is called a rigid motion. A rigid motion with the origin fixed is a rotation. That is, rotations keep the origin fixed and move the points while preserving distances among points. It can be easily shown that a rigid motion can be expressed by a rotation followed by a translation.

2.1. Rotation. A rotation can be expressed by an orthonormal matrix with determinant +1. This matrix is called a rotation matrix. From linear algebra, an orthonormal matrix **R** has only one real eigenvalue with a value of +1. The corresponding eigenvector v, which is invariant under the rotation, is called the rotation axis; $\theta = \arccos[(Tr(\mathbf{R}) - 1)/2]$ is called the rotation angle.[11] Geometrically, the rotation is described by an angle θ about the vector v, as illustrated in Figure 1. For rotations, there are several representations: Euler angle, angle−axis, unit quaternion, and so forth (see, for example, ref 12 and references therein).

In the Euler-angle representation, a rotation is represented by three angles about three mutually perpendicular axes. For example, the corresponding rotation matrix obtained by rotating about the X axis by angle α, then about the Y axis by angle β, and finally about the Z axis by angle γ is given by the following matrix

$$\begin{pmatrix} c\alpha\, c\beta & c\alpha\, s\beta\, s\gamma & c\alpha\, s\beta\, c\gamma + s\alpha\, s\gamma \\ s\alpha\, c\beta & s\alpha\, s\beta\, s\gamma + c\alpha\, c\gamma & s\alpha\, s\beta\, s\gamma - c\alpha\, s\gamma \\ -s\beta & c\beta\, s\gamma & c\beta\, c\gamma \end{pmatrix}$$

where $c\theta = \cos\theta$ and $s\theta = \sin\theta$.

In the angle−axis representation, a rotation is represented by the unit vector along the rotation axis, $v = (v_x, v_y, v_z)$, and the rotation angle, θ. The corresponding rotation matrix is given by the following matrix

$$\begin{pmatrix} v_x^2 + (1 - v_x^2)\, c\theta & v_x v_y(1 - c_\theta) + v_z\, s\theta & v_z v_x(1 - c\theta) + v_y\, s\theta \\ v_x v_y(1 - c\theta) + v_z\, s\theta & v_y^2 + (1 - v_y^2)\, c\theta & v_y v_z(1 - c\theta) - v_x\, s\theta \\ v_z v_x(1 - c\theta) - v_y\, s\theta & v_y v_z(1 - c\theta) + v_x\, s\theta & v_z^2 + (1 - v_z^2)\, c\theta \end{pmatrix} \quad (1)$$

In the unit-quaternion representation, a rotation is represented by a unit quaternion $q = (q_0, q_x, q_y, q_z)$, where $q_0 = \cos(\theta/2)$ and $(q_x, q_y, q_z) = \sin(\theta/2)\, v$, θ is the rotation angle, and v is the unit vector along the rotation axis (through the origin). Let $b \in \mathbb{R}^3$ and b' be the image of b after rotation q. Then, we have $\mathring{b}' = q\mathring{b}\tilde{q}$, where $\tilde{q} = (q_0, -q_x, -q_y, -q_z)$ is the conjugate of q and $\mathring{b} = (0, x, y, z)$ for any $b = (x, y, z) \in \mathbb{R}^3$. Multiplying the quaternions, we have the corresponding rotation matrix **Q**

$$\begin{pmatrix} 2(q_0^2 + q_x^2) - 1 & 2(q_x q_y - q_0 q_z) & 2(q_x q_z - q_0 q_y) \\ 2(q_x q_y + q_0 q_z) & 2(q_0^2 + q_y^2) - 1 & 2(q_y q_z - q_0 q_x) \\ 2(q_x q_z - q_0 q_y) & 2(q_y q_z + q_0 q_x) & 2(q_0^2 + q_z^2) - 1 \end{pmatrix} \quad (2)$$

2.2. Rigid Motion. Recall that a rigid motion M can be represented by a rotation R followed by a translation T, written as $M = T \times R$ or $M = [R,T]$. A convenient way of representing rigid motion is by a 4×4 homogeneous matrix **M**:

$$\begin{pmatrix} \mathbf{R} & t \\ \bar{0} & 1 \end{pmatrix}$$

where **R** is the rotation matrix of R, t is the translation vector of T, and $\bar{0} = (0, 0, 0)$.

Given a point $b = (x, y, z) \in \mathbb{R}^3$, let b' be the image of b after the rigid motion M. Then, we have $(b', 1)^T = \mathbf{M}(b, 1)^T$, where $(b, 1)^T$ is the transpose of $(x, y, z, 1)$. For simplicity, we omit the transpose and write $b' = \mathbf{M}(b)$ when there is no danger of confusion.

We can also express the rigid motion in the quaternion−vector form: $\mathring{b}' = q\mathring{b}\tilde{q} + \mathring{t}$, where q is the unit quaternion representation of the rotation R. When there is no danger of confusion, we denote the rigid motion in quaternion−vector form by $[q, t]$.

2.3. Composition of Rigid Motions. Given two rigid motions represented by homogeneous matrices, $\mathbf{M}_1$ and $\mathbf{M}_2$, we have $\mathbf{M}_2[\mathbf{M}_1(p)] = (\mathbf{M}_2 \times \mathbf{M}_1)(p)$ because matrix multiplication is associative. Hence, the composition of two rigid motions is just the product of their corresponding homogeneous matrices. That is, if

$$\mathbf{M}_1 = \begin{pmatrix} \mathbf{R}_1 & t_1 \\ \bar{0} & 1 \end{pmatrix}$$

and

$$\mathbf{M}_2 = \begin{pmatrix} \mathbf{R}_2 & t_2 \\ \bar{0} & 1 \end{pmatrix}$$

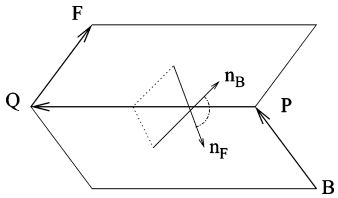

Figure 2. The absolute value of the torsion angle of (*B*, *P*, *Q*, *F*) is the dihedral angle between the planes formed by (*B*, *P*, *Q*) and (*P*, *Q*, *F*).

then we have

$$\mathbf{M}_2 \times \mathbf{M}_1 = \begin{pmatrix} \mathbf{R}_2 \times \mathbf{R}_1 & \mathbf{R}_2(t_1) + t_2 \\ \bar{0} & 1 \end{pmatrix}$$

Similarly, we can compute the composition of rigid motions in quaternion−vector form. Suppose q_1 (and q_2) is the unit quaternion corresponding to the rotation of $\mathbf{R}_1$ (and $\mathbf{R}_2$, respectively). Then, the rotation part of the composite transformation is q_2q_1 and the translation part is $q_2\mathring{t}_1\tilde{q}_2 + \mathring{t}_2$.

3. METHODS

3.1. Basics for Updating Molecular Conformations. In this section, we review some basics about molecules and describe how to represent bond rotations mathematically.

3.1.1. Rotatable Bonds and Torsion Angles. A molecule is represented by a set of atoms and a set of (covalent) bonds between atom pairs. A bond is rotatable if it is single, acyclic (not contained in a ring), and does not connect to a terminal atom. Given four consecutive connected atoms *BPQF*, the torsion angle is defined to be the angle required to rotate *F* clockwise until *B*, *P*, *Q*, and *F* are coplanar. It is customary to define the angle between $[-\pi, \pi]$; that is, when the angle $\theta > \pi$, we set $\theta = \theta - 2\pi$. This allows an alternative definition of the torsion angle based on the dihedral angle: the absolute value of θ is the dihedral angle between planes formed by (*B*, *P*, *Q*) and (*P*, *Q*, *F*). See Figure 2 for an illustration. Imagine that *B*, *P*, and *Q* are fixed and *F* is rotated about the bond $\overrightarrow{PQ}$; then, the corresponding rotation angle is defined to be the change in the torsion angle. A change in one or more torsion angles of a molecule results in a different conformation.

3.1.2. Rigid Fragmentation. A molecule can be divided into a set of rigid fragments (which were called atomgroups in ref 8) according to rotatable bonds. The rigid fragmentation is commonly used in the incremental construction approach of protein−small-molecule docking algorithms, for example, FlexX.[14] As all bonds in rings are regarded as nonrotatable for this application, a molecule can be represented as a tree with rigid fragments as vertices and rotatable bonds as edges. Choose any rigid fragment as the root; the molecule can be represented as a rooted tree, see Figure 3 for an example. Here, the root serves as a reference-rotation point and can be chosen arbitrarily. (The rigid fragmentation procedure can be easily done by the Breadth First Search algorithm, which takes time linearly in the number of bonds of the molecule.)

3.1.3. Representation of Bond Rotation. Denote the *i*th rotatable bond by b_i and let P_i and Q_i be the parent and child atoms of bond b_i, respectively. When bond b_i rotates by angle θ_i, all descendant atoms of Q_i rotate about b_i by θ_i. Denote the transformation corresponding to a rotation about bond $\overrightarrow{P_iQ_i}$ by θ_i as M_i. As described in Figure 1, a mathematical rotation can be represented by rotating about an axis $v = \overrightarrow{oc}$ through the origin by an angle θ. However, an issue arises. In general, $\overrightarrow{P_iQ_i}$ is not a vector through the origin; the rotation R_i obtained by the unit vector $u_{\overrightarrow{P_iQ_i}}$ and θ_i (either by angle axis or unit quaternion) corresponds to a rotation about axis $\overrightarrow{OY}$ with angle θ_i, where $Y = Q_i - P_i$. Thus, R_i is not the same as M_i. (It appears in a recent paper,[15] however, that R_i was erroneously used for M_i.) To use R_i to express M_i, one first translates it by $-Q_i$ such that Q_i becomes the origin; then, one performs the rotation R_i and translates it back by Q_i.

Suppose D rotates about the bond P_iQ_i by angle θ_i. Let D' be the new position (see Figure 4). Then, we have $D' = R_i(D - Q_i) + Q_i = [Q_i - R_i(Q_i)] \circ R_i(D)$ and, thus, $M_i = [R_i, Q_i - R_i(Q_i)]$, where R_i is defined as above.

3.2. Updating Molecular Conformations. In this section, we describe how the method of simple rotations can be implemented efficiently. To compare with the results in ref 8, we also review the atom-group local-frames method introduced there.

3.2.1. The Method of Atom-Group Local Frames. The method of simple rotations is a straightforward approach for computing the new positions of atoms. Suppose $b_1, ..., b_{i-1}$, b_i is a sequence of bonds in the path from atom Q_1 to atom Q_i. Let M_k be the transformation for rotating about bond b_k by angle θ_k, for $k = 1, ..., i$. Then, the new position of atom Q_i, after rotating about bonds $b_1, ..., b_{i-1}$ by angles $\theta_1, ..., \theta_{i-1}$, respectively, is given by $Q_i' = M_{i-1} \times ... \times M_2 \times M_1(Q_i)$. See Figure 5 for an illustration.

Let $N_i = M_i \times ... \times M_2 \times M_1$, for $i \geq 1$, be the accumulated rigid motion required to update atoms. Then, we have $N_i = M_i \times N_{i-1}$ for $i > 1$. Thus, the total number of rigid motions required can be computed in linear time $O(n_{rb})$, where n_{rb} is the number of rotatable bonds. In ref 8, each rigid motion M_i is represented by a homogeneous matrix

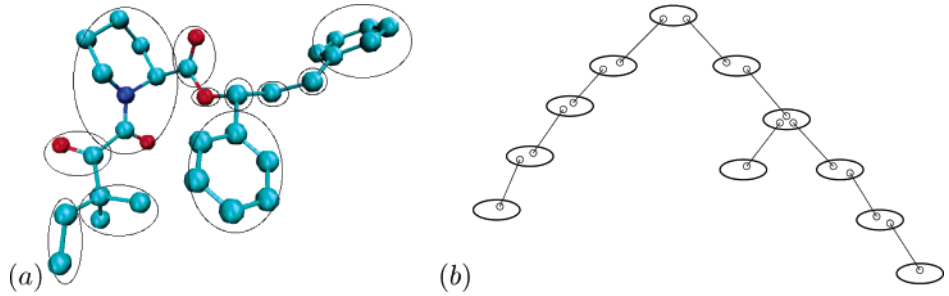

Figure 3. (a) A molecule can be divided into a set of rigid fragments according to rotatable bonds. (b) Tree representation of the molecule in part a with rigid fragments as vertices and rotatable bonds as edges.

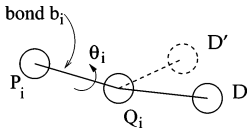

Figure 4. Position D is changed to D' after rotation of angle θ_i about bond b_i.

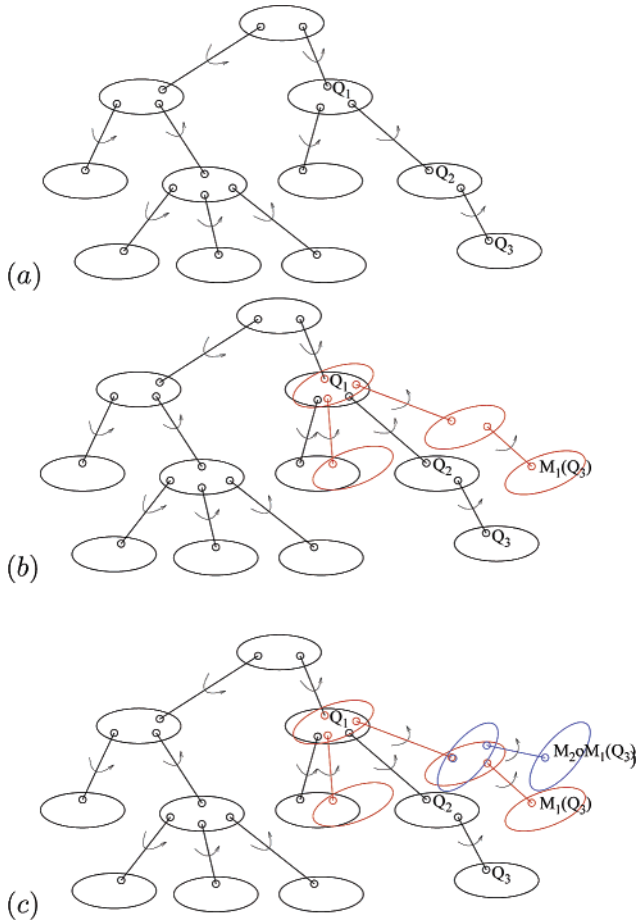

Figure 5. (a) Path Q_1, Q_2, Q_3 in the tree. (b) All descendant atoms of Q_1 are transformed by M_1. In particular, the new position of $Q_3 = M_1(Q_3)$. (c) All descendant atoms of Q_2 are transformed by M_2. The new position of $Q_3 = M_2 \times M_1(Q_3)$.

taken from ref 10; M_i is first computed in order to compute N_i.

However, observe that the rigid motions required for computations are only N_i's and we do not need to compute M_i separately. As there is a common rotation between M_i and N_{i-1}, we can improve the implementation of the above simple scheme by taking advantage of consecutive operations to save a rotation. As described in Section 3.1.3, $M_i = [R_i, Q_i - R_i(Q_i)]$. Let $N_i = [S_i, T_i]$, where S_i is the rotation and T_i is the translation. In general, we have

$$\begin{cases} N_1 = M_1 & i = 1 \\ N_i = M_i \times N_{i-1} = [R_i \times S_{i-1}, R_i(T_{i-1}) + Q_i - R_i(Q_i)] & i > 1 \\ \quad = [R_i \times S_{i-1}, R_i(T_{i-1} - Q_i) + Q_i] \end{cases}$$

That is, one can delay the computation of the translation part—$Q_i - R_i(Q_i)$—of M_i until the computation of $M_i \times N_{i-1}$ and save one rotation operation.

As reviewed in Section 2.1, there are several ways to represent rotations. We implement rotations using unit quaternions and angle−axis rotation matrices. For example, when using unit quaternions, we have

$$\begin{cases} N_1 = M_1 = [q_1, Q_1 - q_1 \mathring{Q}_1 \tilde{q}_1] & i = 1 \\ N_i = M_i \times N_{i-1} = [q_i s_{i-1}, q_i(T_{i-1}^{\circ} - Q_i)\tilde{q}_i + Q_i] & i > 1 \end{cases}$$

where q_i (and s_i) is the unit quaternion to represent R_i (and S_i, respectively).

3.2.2. The Method of Atom-Group Local Frames. For the purpose of comparison, we review the atom-group local-frames method introduced in ref 8. A local frame is attached to a rigid fragment as shown in Figure 6. Local frame $F_i = \{Q_i; u_i, v_i, w_i\}$ is attached to the rigid fragment g_i, where w_i is the unit vector along bond b_i, u_i is any unit vector perpendicular to w_i, and v_i is the unit vector perpendicular to both w_i and u_i; Q_i is one end of the bond b_i in rigid fragment g_i.

Two consecutive local frames F_{i-1} and F_i are related by the following rigid transformation:

$$P_i = \begin{pmatrix} u_{i-1}\cdot u_i & u_{i-1}\cdot v_i & u_{i-1}\cdot w_i & u_{i-1}\cdot(Q_i - Q_{i-1}) \\ v_{i-1}\cdot u_i & v_{i-1}\cdot v_i & v_{i-1}\cdot w_i & v_{i-1}\cdot(Q_i - Q_{i-1}) \\ w_{i-1}\cdot u_i & w_{i-1}\cdot v_i & w_{i-1}\cdot w_i & w_{i-1}\cdot(Q_i - Q_{i-1}) \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

That is, if the coordinates of an atom D in the local frame F_i [and F_{i-1}] are (x_i, y_i, z_i) [and $(x_{i-1}, y_{i-1}, z_{i-1})$, respectively], then $(x_{i-1}, y_{i-1}, z_{i-1}, 1)^T = P_i(x_i, y_i, z_i, 1)^T$. Notice that P_i is rigid-motion invariant. This arises because both frames F_{i-1} and F_i are rotating simultaneously for all bonds on or before b_{i-1}; all terms in P_i are dot products, which rigid motions preserve. Suppose D is an atom in g_i, where its coordinates in the local frame F_i are (x_i, y_i, z_i). The matrix for rotating D around w_i by θ_i is as follows:

$$\begin{pmatrix} x_i' \\ y_i' \\ z_i' \end{pmatrix} = \begin{pmatrix} \cos\theta_i & -\sin\theta_i & 0 \\ \sin\theta_i & \cos\theta_i & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix}$$

Hence, we can get the new position in local frame F_{i-1} with the following expression: $(x_{i-1}', y_{i-1}', z_{i-1}', 1)^T = P_i(x_i', y_i', z_i', 1)^T = M_i(x_i, y_i, z_i, 1)^T$, where

$$M_i = P_i \times \begin{pmatrix} \cos\theta_i & -\sin\theta_i & 0 & 0 \\ \sin\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Therefore, the coordinates of an atom in local frame F_i can be represented in a global frame after a series of transformations: $(x_g, y_g, z_g, 1)^T = M_1 \times M_2 ... \times M_i(x_i, y_i, z_i, 1)^T$.

4. COMPARISON OF METHODS

In this section, we compare our improved simple rotations method with the atom-group local-frames method. There are two implementations of our method: one with rotations represented in unit quaternions and the other with rotations represented in angle−axis rotation matrices. Following ref 8, we compare these methods with respect to the number of multiplications needed to update the positions of all atoms in a molecule. In this comparison, n_a and n_{rb} are the numbers

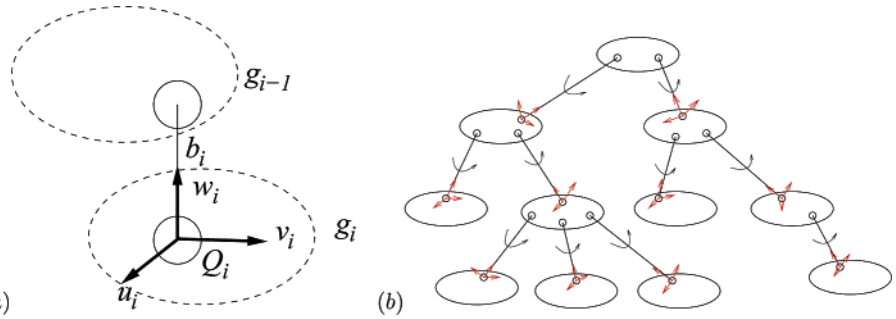

Figure 6. (a) Local frame $F_i = \{Q_i; u_i, v_i, w_i\}$ at rigid fragment g_i. (b) A local frame is attached to each rigid fragment.

Table 1. Comparisons of the Number of Multiplications in the Four Methods[a]

	naïve simple rotations (SR)	atom group local frames	improved SR in unit quaternions	improved SR in rotation matrices
# multiplications	$62n_{rb} + 9n_a$	$48n_{rb} + 9n_a$	$50n_{rb} + 9n_a$	$53n_{rb} + 9n_a$

[a] n_a is the number of atoms, and n_{rb} is the number of rotatable bonds.

of atoms and rotatable bonds, respectively. The cost of computing the cosine and sine functions of torsion angles are ignored in these comparisons because the cost is the same for these methods.

In our quaternion-based implementation, the rotation part of N_i, which equals the multiplication of two quaternions, requires 16 multiplications. Converting a quaternion into a rotation matrix by eq 2 requires nine additional multiplications. Generation of the quaternion q_i requires computing a unit vector along the bond b_i; this computation uses three divisions of the bond length. This length is precomputed because it is rigid-motion invariant. Computing the four coordinates of a quaternion requires one division and three multiplications. Consequently, 32 multiplications are needed for the rotation part. For the translation part, converting a quaternion into a rotation matrix by eq 2 requires nine multiplications; nine multiplications are needed for computing the new translation. Thus, a total of 18 multiplications are needed for the translation part. Finally, each atom needs nine multiplications to obtain its coordinates. In summary, this implementation requires $50n_{rb} + 9n_a$ multiplications to update a molecular conformation.

In our matrix-based implementation, the rotation part requires 27 multiplications to multiply two rotation matrices; three divisions are required to compute the unit vector $(v_x, v_y, v_z)^T$ along the bond b_i, and 14 additional multiplications are required to generate rotation matrix $\mathbf{R}_i$ by eq 1. Thus, 44 multiplications are required for the rotation part, and nine multiplications are required for the translation part. Each atom also needs nine multiplications to obtain its coordinates. Hence, the total number of multiplications in this implementation is $53n_{rb} + 9n_a$.

We implemented the four methods in the program language C++: (1) naïve implementation of the simple rotations method, (2) our improved simple rotations method with rotations represented in unit quaternions, (3) our improved simple rotations method with rotations represented in angle−axis rotation matrices, and (4) the atom-group local-frames method. Table 1 presents the comparison of these four methods with respect to the number of multiplications. Notice that, in our implementation for 1, the number of multiplications is $62n_{rb} + 9n_a$ instead of $75n_{rb} + 9n_a$ as stated in ref 8.

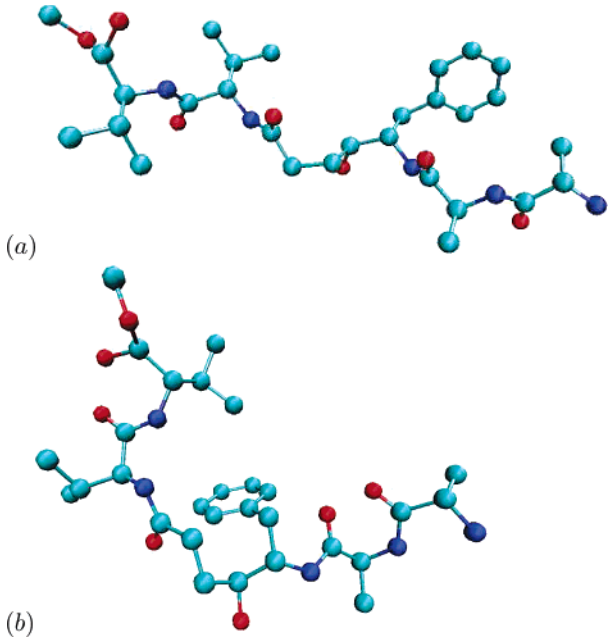

Figure 7. (a) Bound conformation of a molecule (the ligand of 1aaq). (b) A conformation of the molecule after random torsion-angles change.

We tested the four programs on 100 small molecules, whose number of rotatable bonds ranges from 1 to 22, on a 3.0 GHz Pentium 4 computer running Linux Fedora 8.0. See Figure 7 for an example of a small molecule (the ligand of the complex 1aaq). We measured the average running time for 10 000 rounds of random rotations of each molecule. Small differences of running time appeared in these four programs. The improved simple rotations method with rotations represented in unit quaternions was as efficient as the atom-group local-frames method and was slightly faster than the same method with rotations represented in angle−axis rotation matrices. The latter was slightly faster than the naïve implementation of the simple rotations method. We also tested our programs on proteins with the number of rotatable bonds ranging from 300 to 1000; each protein was tested for 1000 rounds of random rotations. See Figure 8 for an example of a protein (the protein of the complex 1cbs). Similar time differences were consistently observed. (See

Table 2. Normalized Running Time of the Four Methods[a]

	molecule 1 (ligand of 8atc)	molecule 2 (ligand of 1glp)	molecule 3 (ligand of 1aaq)	molecule 4 (protein of 1cbs)	molecule 5 (protein of 1hdc)
number of rotatable bonds	5	9	17	534	871
(1) naïve simple rotations (SR)	1	1	1	1	1
(2) improved SR in unit quaternion	0.85	0.81	0.81	0.82	0.85
(3) improved SR in rotation matrix	0.93	0.95	0.95	0.96	0.97
(4) atom group local frames	0.85	0.88	0.85	0.82	0.83

[a] The precomputation time is not included in the running time.

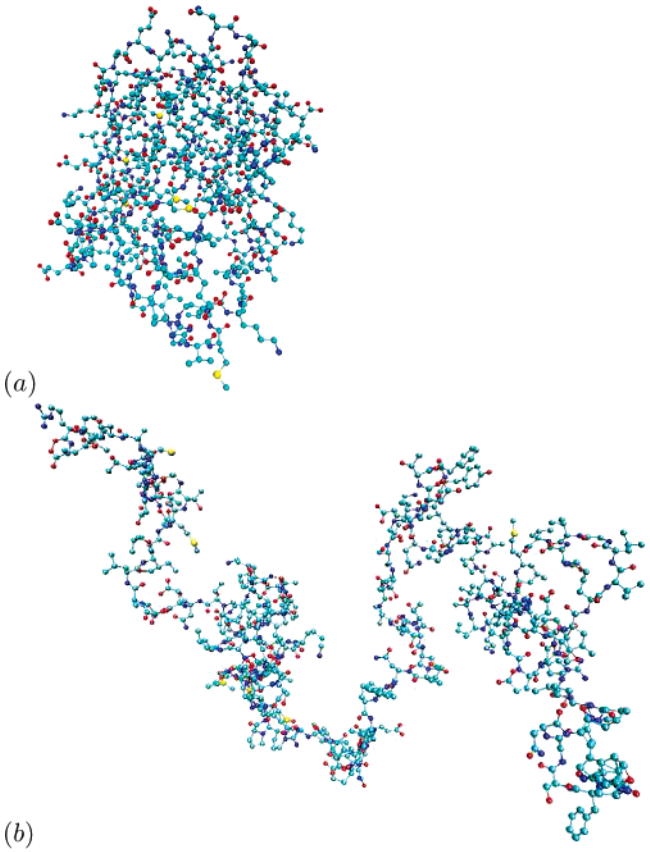

Figure 8. (a) Bound conformation of a molecule (the protein of 1cbs). (b) A conformation of the molecule after random torsion-angles change.

Table 2 for the normalized running time on examples of three small molecules and two proteins.)

5. DISCUSSION

Both methods (the simple rotations and the atom-group local frames) are implemented in terms of rigid fragments (or atom groups). The difference is that the simple rotations method is a global frame method while the atom-group local-frames method is a local-frame method. Compared to the simple rotations method, the local-frames method does not require the position of any other atoms explicitly to update one particular atom. This is in contrast to the simple rotations method in which one needs to update the ancestor atoms along the path. Thus, for the applications in which only some key atom positions require updating, the local-frames method might be better than the simple rotations method, provided that local frames and transformations for changing frames are precomputed. However, for applications that require updating the entire molecule, such as a conformation generator, both methods have almost the same efficiency (when the simple rotations method was appropriately implemented as described in this paper). Unit quaternions are usually computationally more efficient in representing rotations.[16] In particular, unit quaternions have the advantage of computing the composition of rotations efficiently (with 16 multiplications). However, in our case, because we need to compute the coordinates also, converting a unit quaternion into its matrix requires another nine multiplications. Thus, in terms of the number of multiplications, a composition of rotations requires a total of 25 multiplications with unit quaternions versus 27 multiplications with angle−axis rotation matrices. Overall, as our experiments showed, using unit quaternions to represent rotations for updating molecular conformations is slightly more efficient than using angle−axis rotation matrices.

In ref 8, it was argued that the simple rotations method needs bookkeeping (which means the position of ancestor atoms need to be calculated first) and might accumulate numerical errors. While the atom-group local-frames method does not need bookkeeping and will not accumulate numerical errors through bookkeeping, the local-frames method could accumulate the numerical errors through the local-frames changes. The fundamental difference between the two methods is that one computes "forward" and the other "backward". Both methods involve a sequence of transformations, in which errors might accumulate.

The efficiency of the local-frames method relies on precomputing local frames and transformations for changing frames. In doing so, one assumes that the molecular conformation changes are only due to changes in torsion angles. If the bond lengths/bond angles also change, then local frames and transformations can no longer be precomputed; their computational cost will lead to a considerably higher running time.

Our simple rotations algorithm is used in our developing conformation generator that is part of our docking algorithm YUCCA.[17] The source code of our program is available upon request.

ACKNOWLEDGMENT

The author gratefully acknowledges her colleague, Rich Gandour, for his generosity and patience in teaching her an organic chemist's view of conformational analysis and coaching her in English (American) writing. She thanks Xiaoyan Yu and Wenjie Zheng for implementing an early version of the program. Thanks also to the anonymous reviewers for several helpful comments.

REFERENCES AND NOTES

(1) Crippen, G. M.; Havel, T. F. *Distance Geometry and Molecular Conformation*; J. Wiley & Sons: New York, 1988.

(2) Havel, T. F. Distance geometry: theory, algorithms, and chemical applications. In *Encyclopedia of Computational Chemistry*; Ragué, V., Schreiner, P. R., Allinger, N. L., Clark, T., Gasteiger, J., Kollman, P. A., Schaefer, H. F., III, Eds.; J. Wiley & Sons: New York, 1998; pp 723−742.
(3) Leach, A. R. *Molecular Modelling: Principles and Applications*, 2nd ed.; Prentice Hall: New York, 2001.
(4) *Chemoinformatics*; Gasteiger, J., Engel, T., Eds.; Wiley-VCH: Weinheim, Germany, 2003.
(5) Dong, Q.; Wu, Z. A linear-time algorithm for solving the molecular distance geometry problem with exact interatomic distances. *J. Global Optimization* **2002**, *22*, 365−375.
(6) Unless using the energy function as in ICFF (ref 7), which is expressed in terms of internal coordinates.
(7) Katritch, V.; Totrov, M. M.; Abagyan, R. A. ICCF: a new method to incorporate implicit flexibility into an internal coordinate force field. *J. Comput. Chem.* **2003**, *24*, 254−65.
(8) Zhang, M.; Kavraki, L. E. A new method for fast and accurate derivation of molecular conformations. *J. Chem. Inf. Comput. Sci.* **2002**, *42*, 64−70.
(9) Craig, J. J. *Introduction to robotics*; Addison-Wesley: Reading, MA, 1989.
(10) Foley, J. D.; van Dam, A.; Feiner, S. K.; Hughes, J. F. *Computer Graphics: Principles and Practice*; Addison-Wesley: Reading, MA, 1990.
(11) Kuipers, J. B. *Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace and Virtual Reality*; Princeton University Press: Princeton, New Jersey, 2002.
(12) Flower, D. R. Rotational superposition: a review of methods. *J. Mol. Graphics Modell.* **1999**, *17*, 238−244.
(13) Horn, B. K. P. Closed-form solution of absolute orientation using unit quaternions. *J. Opt. Soc. Am. A* **1987**, *4*, 629−642.
(14) Rarey, M.; Kramer, B.; Lengauer, T.; Klebe, G. A fast flexible docking method using an incremental construction algorithm. *J. Mol. Biol.* **1996**, *261*, 470−489.
(15) Alvarado, C.; Kazerounian, K. On the rotational operators in protein structure simulations. *Protein Eng.* **2003**, *16*, 717−720.
(16) Mason, M. T. *Mechanics of Robotic Manipulation*; Massachusetts Institute of Technology Press: Cambridge, MA, 2001.
(17) Choi, V. YUCCA: An Efficient Algorithm for Small Molecule Docking. *Chem. Biodiversity* **2005**, *11*, 1517−1524.

CI050253H