JOURNAL OF CHEMICAL INFORMATION AND MODELING

Article

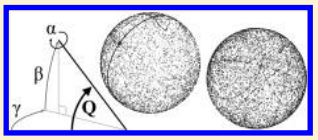
pubs.acs.org/jcim

Doing a Good Turn: The Use of Quaternions for Rotation in Molecular Docking

Gwyn Skone,* Stephen Cameron,* and Irina Voiculescu

Department of Computer Science, Oxford University, Wolfson Building, Parks Road, Oxford, OX1 3QD, United Kingdom

ABSTRACT: Much work has been done on algorithms for structure-based drug modeling in silico, and almost all these systems have a core need for three-dimensional geometric models. The manipulation of these models, particularly their transformation from one position to another, is a substantial computational task with design questions of its own. Solid body rotation is an important part of these transformations, and we present here a careful comparison of two established techniques: Euler angles and quaternions. The relative superiority of the quaternion method when applied to molecular docking is demonstrated by practical experiment, as is the crucial importance of proper adjustment calculations in search methods.



INTRODUCTION

Computational molecular modeling relies to a large extent on the methods used to represent and manipulate a molecule's structure. Protein−ligand docking is governed substantially by the shapes of the molecules involved. The interactions between them are often concentrated at their boundaries, and the extreme repulsion of atoms when colliding ensures that there cannot be any significant overlap. Consequently, the geometry of the molecules is an important source of guidance when searching for a good binding pose. Many models of these structures have been presented and reviewed over the course of the last two decades,[1−5] including simple collections of spheres, surface parametrizations,[6,7] and general functions such as energy maps.[8,9] In this paper we concern ourselves not with the choice of representation for docking, but what has to be done to it.

Figure 1 shows an abstract overview of a generalized virtual screening process. The shaded part represents the search method in use, exploring poses by selecting and evaluating possible arrangements and collecting their scores until some termination criteria are met. If this search generates thousands of poses (as would be typical) and thousands of ligand conformations are to be screened, the transformation step will be executed millions of times. This must at least translate and/or rotate the chosen structural representation into a new position, assuming no additional degrees of freedom are introduced by flexibility in the model. As the position of each atom in the ligand (or other spatial datum in less physical models) will have to be transformed many times, this adjustment stage is a keystone of any high-throughput system. The transformations must be efficient and behave linearly; any bias or inaccuracy in their execution will affect the overall model's behavior.

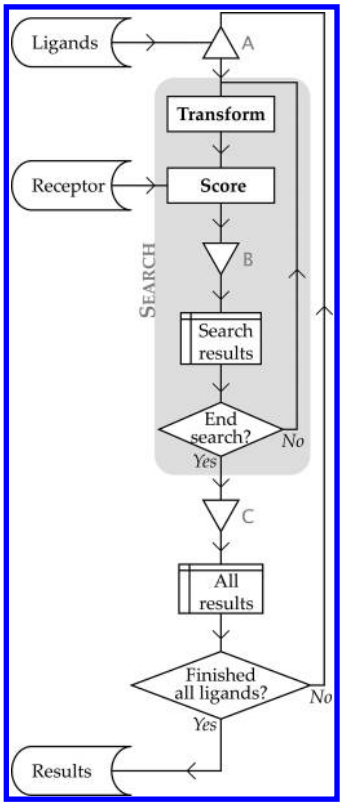


Figure 1. Abstract overview of the docking process for virtual screening. (A) Extracting a ligand/conformation from the input set. (B) Merging a scored pose with interim results. (C) Merging best poses with master result set.

METHODS

We consider the task of transforming a three-dimensional (3D) position in space: performing some operation that obtains a new point from an existing one. If this is applied to all points in a collection (for example, all atoms in a molecule), the whole molecule is thus moved. The two basic types of transformation applicable here are translation (moving along a straight line)

Received: September 4, 2013
Published: November 10, 2013

dx.doi.org/10.1021/ci4005139 | J. Chem. Inf. Model. 2013, 53, 3367−3372

and rotation (turning about a pivot), since these do not distort an overall shape if applied to its component parts separately. Mathematically, these can be modeled using vector arithmetic, where a vector $\mathbf{v}$ is a 3D coordinate: $\mathbf{v} \equiv [v_x, v_y, v_z]$.

Translations are simple to handle. A single vector $\mathbf{t}$ can record a displacement along each dimension, such that a translated point $\mathbf{v}'$ from $\mathbf{v}$ is given by

$$\mathbf{v}' = \mathbf{v} + \mathbf{t} \tag{1}$$

Rotation lends itself to many different designs, including matrices, Euler angles, and quaternions. Note that rotations are always described about the origin (zero vector). If some other pivot point $\mathbf{c}$ is required, this must be incorporated as an additional pre- and post-translation. Thus, for a rotation function R, the rotation of $\mathbf{v}$ to $\mathbf{v}'$ is

$$\mathbf{v}' = \mathrm{R}(\mathbf{v} - \mathbf{c}) + \mathbf{c} \tag{2}$$

and a complete rigid body transformation is

$$\mathbf{v}' = \mathrm{R}(\mathbf{v} - \mathbf{c}) + \mathbf{c} + \mathbf{t} \tag{3}$$

Note that, for efficiency, it is often helpful to keep the vectors being analyzed in their $\mathbf{v} - \mathbf{c}$ form, since the absolute position of a shape in space is usually a separate concern from its orientation. Furthermore, translations can be easily chained, and applying a rotation to a translation takes the same effort as applying a rotation to a point vector.

■ ROTATION REPRESENTATIONS

The rotation function R can be defined in several ways, each being suited to different situations.

Matrices. The most explicit method, which can be readily combined with any other transformations, is to premultiply by a 3×3 matrix $\mathbf{M}$, since this gives each component of the transformed vector as a linear combination of the original components. Using this system, the rotation function $\mathrm{R}_\mathbf{M}$ for a rotation matrix $\mathbf{M}$ is

$$\mathrm{R}_\mathbf{M}(\mathbf{v}) = \mathbf{M}\mathbf{v} \tag{4}$$

This method might be considered unduly general, since $\mathbf{M}$ could represent many transformations other than rotation, including scaling, shearing, or even some nonlinear distortion, all of which are unlikely to be useful in molecular modeling. Constructing $\mathbf{M}$ is nontrivial except when the rotation is about one axis. In these cases, only the sine and cosine functions of the desired angle are required.[10]

The inverse of a rotation is given by its inverted matrix. Since rotation is an orthogonal transformation its matrix is also orthogonal, and so the transpose is equal to the inverse but much easier to calculate. Rotations may be composed by matrix multiplication, demanding 27 multiplications and 18 additions.

A lesser consideration is that matrices are an inefficient use of storage space, since they require nine floating-point values, although these are not all independent if the matrix is required to represent only a rotation.

Euler Angles. Euler angles describe an arbitrary 3D rotation as a combination of three axial rotations: angles $\alpha \in (-\pi, +\pi]$, $\beta \in [-\pi/2, +\pi/2]$, and $\gamma \in (-\pi, +\pi]$ This requires only three numbers to be recorded but does not make explicit the choice of rotation axes. Whereas a matrix completely defines a transformation, Euler angles must use an implicit convention to determine how the three angles should be used. Many variations are equally valid, but it is crucial that whichever is chosen is applied consistently.

Since matrices for axial rotations can be constructed directly,[10] it is straightforward to define the Euler angle rotation function (using a z, y, x axis convention) as

$$\mathrm{R}_{\mathrm{zyx}_{\alpha,\beta,\gamma}}(\mathbf{v}) = (\mathbf{M}_{\mathbf{x}\gamma}\mathbf{M}_{\mathbf{y}\beta}\mathbf{M}_{\mathbf{z}\alpha})\mathbf{v} \tag{5}$$

$$= \mathbf{M}_{\mathbf{zyx}_{\alpha,\beta,\gamma}}\mathbf{v} \tag{6}$$

where $\mathbf{M}_{\mathbf{z}\alpha}$ is the matrix for a rotation by α about the z axis, and so on. It is not necessary to perform the full trio of matrix multiplications: the combined matrix representation can be computed directly from the angles instead, avoiding the many zero-products that would normally be involved.

Although Euler angles make it easy to describe a rotation, they do have several pitfalls. Perhaps the most significant is "gimbal lock": a degree of freedom is lost when one angle takes values near $\pm\pi/2$, resulting in the coincidence of two axes.[11]

Generating evenly distributed rotations (as is typically done to initialize a search) is possible, but care must be taken to avoid a bias toward the areas where gimbal lock occurs. Usually, the α and γ values can be selected in a uniform manner, but the β value should be chosen such that $\sin(\beta)$ has uniform distribution. The reason for this can be visualized easily by imagining the lines of latitude and longitude on a globe, and their increasing density around the poles.

The noncommutativity of matrix multiplication presents other problems. The composition of Euler angle rotations is not immediately representable as a single set of Euler angles, since the sequence of axial rotations will not be consistent:

$$\begin{aligned}&(\mathbf{M}_{\mathbf{x}\gamma 2}\mathbf{M}_{\mathbf{y}\beta 2}\mathbf{M}_{\mathbf{z}\alpha 2})(\mathbf{M}_{\mathbf{x}\gamma 1}\mathbf{M}_{\mathbf{y}\beta 1}\mathbf{M}_{\mathbf{z}\alpha 1})\\ &\quad\equiv \mathbf{M}_{\mathbf{x}\gamma 2}\mathbf{M}_{\mathbf{y}\beta 2}\mathbf{M}_{\mathbf{z}\alpha 2}\mathbf{M}_{\mathbf{x}\gamma 1}\mathbf{M}_{\mathbf{y}\beta 1}\mathbf{M}_{\mathbf{z}\alpha 1}\\ &\quad\not\equiv (\mathbf{M}_{\mathbf{x}\gamma 2}\mathbf{M}_{\mathbf{x}\gamma 1})(\mathbf{M}_{\mathbf{y}\beta 2}\mathbf{M}_{\mathbf{y}\beta 1})(\mathbf{M}_{\mathbf{z}\alpha 2}\mathbf{M}_{\mathbf{z}\alpha 1})\end{aligned} \tag{7}$$

Inversion may be difficult too, because the inverted axial rotations must also be applied in reverse order. If the first and third axes are the same, such as in an x, y, x system, then an inversion is trivial:

$$\mathrm{R}_{\mathbf{xyx}_{\alpha,\beta,\gamma}}{}^{-1} \equiv \mathrm{R}_{\mathbf{xyx}_{-\gamma,-\beta,-\alpha}} \tag{8}$$

However, in the more general case, this method of negating and reversing the three angles fails.

Quaternions. Quaternions are hyper-complex numbers, first described by Hamilton,[12] with one real and three imaginary parts:

$$\begin{aligned}&\mathbf{Q} \equiv a + xi + yj + zk\\ &\quad a, x, y, z \in \mathbb{R}\\ &\quad i^2 = j^2 = k^2 = ijk = -1\end{aligned} \tag{9}$$

They may, for geometric purposes, be regarded as 4-tuples:

$$\mathbf{Q} \equiv [a, x, y, z] \tag{10}$$

Their use for rotation is an established technique in engineering and robotics since it allows a continuous representation of all possible rotations. Whereas the space of Euler angle rotations may be thought of as the volume of a cuboid of dimensions $(2\pi, \pi, 2\pi)$, the space of all quaternion rotations is the surface of a unit 4D hypersphere:

$$|\mathbf{Q}| = a^2 + x^2 + y^2 + z^2 = 1 \tag{11}$$

dx.doi.org/10.1021/ci4005139 | J. Chem. Inf. Model. 2013, 53, 3367−3372

A unit quaternion $\mathbf{Q}$ represents a rotation by $2\arccos(a)$ about the axis vector $[x, y, z]$. Since this representation always explicitly defines the axis of rotation, there is no potential for confusion about frames of reference, as with Euler angles. The special case of a null axis, when $x = y = z = 0$, is useful since eq 11 guarantees that $a = \pm 1$, and thus the angle $2\arccos(a) = 0$ specifies the null rotation.

Quaternion rotations may be manipulated consistently and easily, detailed expositions of which have been published.[11] We summarize here for convenience the key operations relevant to docking.

The composition of two rotations is given by their quaternions' product. This allows Euler angles to be readily converted to a quaternion: the three components may be constructed directly from their angles and axes and, then, multiplied to give the single composite rotation.

A rotation $\mathbf{Q}$ may be inverted by quaternion conjugation, corresponding to the same rotation angle about the reverse axis: $\bar{\mathbf{Q}} \equiv [a, -x, -y, -z]$. Consequently, a relative rotation from orientation $\mathbf{Q}_1$ to $\mathbf{Q}_2$ is easily calculated as $\mathbf{Q}_2\bar{\mathbf{Q}}_1$.

Applying a rotation to a vector is done by quaternion multiplication, without the use of trigonometry:

$$R_{\mathbf{Q}}(\mathbf{v}) = \mathbf{Q}\mathbf{v}\bar{\mathbf{Q}} \tag{12}$$

where the quaternion representation of $\mathbf{v}$ is

$$\mathbf{v} = [0, v_x, v_y, v_z] \tag{13}$$

Expanding eq 12, substituting $|\mathbf{Q}| \equiv 1$ (from eq 11), and converting the result back to its corresponding vector (since the resulting a component is always zero) gives $R_{\mathbf{Q}}(\mathbf{v}) =$

$$2\begin{bmatrix} \frac{1}{2} - y^2 - z^2 & -az + xy & ay + xz \\ az + xy & \frac{1}{2} - x^2 - z^2 & -ax + yz \\ -ay + xz & ax + yz & \frac{1}{2} - x^2 - y^2 \end{bmatrix}\mathbf{v} \tag{14}$$

Note also that

$$R_{\mathbf{Q}}(\mathbf{v}) \equiv R_{-\mathbf{Q}}(\mathbf{v}) \tag{15}$$

where $-\mathbf{Q} \equiv [-a, -x, -y, -z]$, since for all $a \in [-1, +1]$ $2\arccos(a) + 2\arccos(-a) = 2\pi$, and a negative axis simply reverses the direction of rotation.

Quaternion rotations may be stored using only three of their component values, the fourth being determined (except for sign) by the unit magnitude constraint. Consistently selecting either the positive or negative value of this fourth component will provide a complete rotation space, since this merely eliminates the duplication resulting from eq 15. If $[x, y, z]$ are used, then quaternion space is the interior of a unit 3D sphere, with each tuple a vector whose direction defines an axis and whose length defines an angle. Although the reduced storage demands are unlikely to be a significant design factor in a virtual screening program, this could still be worthwhile in situations where most quaternions are picked at random or frequently modified. It allows only three components to be chosen from statistical distributions (which can be time-consuming), with the missing value being calculated analytically when needed. Unfortunately, this does lead to imprecision when $[x, y, z]$ is very small, since the value of a will vary significantly for minute changes in any of x, y, z. Consequently, it is often better to define a rotation as an axis and an angle separately, since the quaternion can be constructed easily from those values.

Selecting random orientations uniformly can be done by picking a, x, y, z each with Gaussian distribution, and normalizing the resulting quaternion. This is equivalent to picking a, x, y, z uniformly between 0 and 1, and applying the inverse cumulative distribution function.[13] An alternative method, but simpler to implement, is to pick a, x, y, z uniformly between -1 and $+1$, rejecting tuples with greater than unit magnitude, and normalize. This is simple to process, but by considering volumes we see that only $((1/2)\pi^2)/2^4 \approx 31\%$ of points generated will be usable. Which method is more efficient will then depend on the particular hardware architecture features and statistical implementations available, but the former is more suited to docking methods since it uses all values in a uniform range as is often expected by a search algorithm.

Comparison. The three representations discussed each have relative merits. Matrices are unambiguous, flexible enough to combine with other transformations, and straightforward to apply with standard matrix operations. Euler angles are more natural to understand and use, fairly quick for rotation, and compact for storage. Quaternions are as compact as Euler angles and as easily manipulated as matrices.

However, matrices are unwieldy data structures, requiring more data transfer regardless of any manipulations performed, and their ability to represent and combine many other kinds of transformation is not relevant to the rigid-body modeling problem considered here. A more intuitive, geometry-based representation is far more useful for search methods to manipulate, as well as for users to read in results data. Euler angles also have limitations, particularly the inability to compose and invert them directly, but also the degeneracies caused by gimbal lock introducing distortion to a search method. Quaternions, though, require more arithmetic instructions than Euler angles to apply a rotation to a vector: by far the most common operation required.

Table 1 lists, for each representation and common operation, the computational cost of execution. Recent desktop commodity processors claim speeds of the order 10^{10} floating point operations per second (FLOPS),[14] but in a multitasking operating system this capacity is unlikely to be fully allocated to a single process. On a PC with an AMD Athlon 3500+ processor, a simple program performing 10^9 additions and 10^9 multiplications of random floating-point data achieved 5.69×10^8 FLOPS (1.76 ns per operation), while an Intel Core 2 Duo P8400 ran the same program on a single core at 6.12×10^8 FLOPS (1.63 ns per operation). The SIMD vector operation counts (capital letters) assume that double precision floating point variables are used, allowing two values to be packed into each operand. Using single precision would approximately halve the number of each operation. Some very recent processors include dot product instructions, allowing for two (or four) scalar multiplications and one (or three) scalar additions to be combined where appropriate. These approximate figures give only a general scale of time costs; however, every computer's hardware configuration will have an impact on its capacity for handling and processing data, especially the quantity of memory and effectiveness of processor caching.

A docking program for virtual screening, as already discussed, must generate and evaluate many poses of a ligand in the context of a receptor. These poses require transformations, selected to place the small molecule in a position near an active

dx.doi.org/10.1021/ci4005139 | J. Chem. Inf. Model. 2013, 53, 3367−3372

Table 1. Arithmetic Operations Required for Common Manipulations of Rotation Representations[a]

task	matrix	Euler angles	quaternion
transform: R(v)	$9m + 6a$	$6t + 12m + 9a$	$21m + 18a$
or	$5M + 3A$	$6t + 6M + 5A$	$11M + 9A$
		once, then as matrix[b,d]	
compose: R_2R_1	$27m + 18a$	n/a	$16m + 12a$
or	$14M + 9A$		$8M + 6A$
invert: R^{-1}	only copy	n/a	$3a$
storage[c] (values)	9	3	4

[a]Key: m scalar multiplications, a scalar additions, t trigonometric functions, M vector multiplications, A vector additions. [b]Figures given for Euler angles are for directly constructing the single matrix representation of three Euler angle rotations, since this can be reused more efficiently than the three components, especially without vector processing instructions. [c]Storage size is the number of floating-point values, typically 4 or 8 bytes each. Load and store operations have not been included—they should be proportional to the data size. If single-instruction-multiple-data (SIMD) vector processing is available, as it is in most modern hardware (such as the x86 architecture), the vector operations may be performed as a single instruction if single-precision floating point values are used, or two instructions in double-precision. Without such capabilities, these will generally be equivalent to three or four instructions, one per vector component. [d]For efficiency, trigonometric functions can be precomputed in look-up tables, although many modern processors have optimizations built-in that make such stratagems unnecessary. It is also common to have a combined sine−cosine opcode available, halving the number of actual trigonometric instructions required. When the same rotation is to be used repeatedly, it is worth constructing the single matrix representation of the Euler angles once, and then using standard matrix operations to apply it each time it is required.

site. Translations are readily chosen from the bounding dimensions of the search space. Rotations, depending on the representation used, are more complicated. It is impractical to design an automatic search algorithm that produces all nine rotation matrix elements directly and appropriately to the context. A more logical model is generally used to reduce the number of parameters and make each more clearly meaningful: Euler angles and quaternions are both suited to this.

We suggest that quaternions are a simpler representation to apply in this kind of system, where frequent and versatile vector manipulations are required. The continuity of the model and explicit axial interpretation ensure that they can be combined easily and reliably, maintaining efficiency and accuracy.

RESULTS

To compare these two methods, the docking program *DOX* was compiled in two editions, one using each rotation representation, by way of C++ preprocessor directives.[15,16]

As noted in the Introduction, the choice of docking process is arbitrary provided it follows the basic outline of Figure 1. For the comparison being made here, we require a search method purely as a generator of ligand poses to be evaluated by some scoring function. Any variation in efficiency between rotation representations would be replicated in any search method. The uniformity of a rotation representation could affect different search methods to different extents, but any method that uses neighboring or previously considered cases in its iteration will be susceptible to inconsistencies of the geometric model. This would apply to gradient descent and hill climbing,[17] tabu search,[18] simulated annealing,[19] particle swarms,[20] and bee colonies,[21] among others.

The search method used here was a traditional genetic algorithm (GA)[22] based on the *GAlib* library,[23] using the empirical *XScore* function[24,25] to evaluate fitness scores. Local optimization was performed on the final populations, not at any intermediate stage, using the Nelder−Mead simplex algorithm.[17] The crossover operator was the default provided by GAlib: swapping the values of randomly chosen elements between the parent genomes. Mutation had to be modified from the default, because rotation values (angles) do not have a finitely bounded range, but are circular. The GAlib mutator would adjust the value of randomly selected elements by standard Gaussian random offsets, constraining the results to the permitted range of the genome. Instead, we amended this behavior to allow values to "wrap" around to the other extremity: for example, a mutation by 0.1 limited to the range $[-1, +1)$ in standard bounded mutation would be $0.93 + 0.10 = 1.00$; whereas, circular mutation would be $0.93 + 0.10 = -0.97$. To illustrate the importance of this, Figure 2 shows all of the

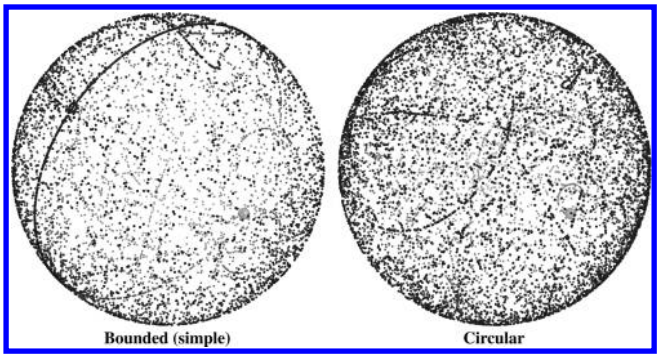


Figure 2. Effect of mutation operator choice when searching orientation space. The larger gray sphere (lower-right) is the starting position of all 1000 population members of 10 GA instances. After 3000 generations of evolution, using a constant scoring function giving no preference to any particular direction, the final positions are shown by the 10 000 small black dots. The shading of the dots is an indication of depth in the image.

results overlaid together from ten runs of evolving a population of 1000 single-atom molecules over 3000 generations using a constant scoring function, 80% mutation rate, 50% crossover rate, and no further optimization. Translation was ignored in this test. The heavy oversampling of a single great circle of orientations is clearly visible, showing the serious bias resulting from a bounded mutator. Using the circular mutator, a much more even coverage of rotation space is obtained.

The search range was specified to be an axis-aligned cuboid exactly containing the native ligand pose but with each dimension then doubled in length. The initial populations were uniformly distributed random translations and rotations. Each individual was an array of double-precision floating point values, containing three coarse translation values (within the search range), three fine translation values (within ±1 Å), and three orientation values. In both Euler angle and quaternion systems, the orientation was defined in an axis-angle style. The Euler angle system used the z, y, x axis convention (yaw-pitch-roll), while the quaternion system used the longitude and latitude of an axis and a rotation around it. In both cases, the second (β or latitude) value was stored as the sine of the angle in $[-1, +1)$ to counteract gimbal lock, while the third angle (γ or rotation) was stored as a value $r \in [-1, +1)$ where $r = (\gamma - \sin(\gamma))/\pi$ to ensure uniform distribution of orientations.[26]

dx.doi.org/10.1021/ci4005139 | J. Chem. Inf. Model. 2013, 53, 3367−3372

The test cases were those of the entire *Astex* Diverse Set, a well-established collection of 85 varied and practically relevant protein−ligand complexes illustrating a wide variety of behaviors and structures.[27] For each case, DOX redocked the ligand to its receptor and recorded the final set of 100 result poses, ordered by score. Statistics were also calculated and saved with the output, including root-mean-square deviations (RMSDs) from the ligand's native pose and the time taken to complete the search (excluding file reading/writing). This was performed six times over, to allow for the randomness of the GA method, producing 510 sets of results in total.

Figure 3 shows that the system using Euler angles took 25% longer to complete. This exemplifies the benefit from

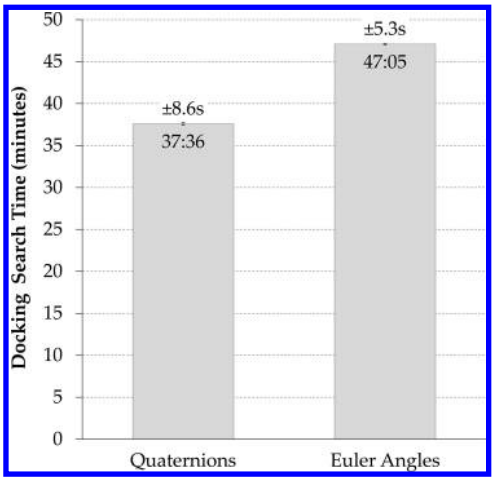


Figure 3. Execution speeds of the two rotation representations. Docking time refers to one complete set of redockings of the 85 complexes in the Astex set, excluding data reading/writing, and is the mean of 6 runs. Confidence intervals are at the 95% level. Hardware used: AMD Athlon 3500+ processor, 1 GB of RAM.

simplifying the computational cost of performing rotations in this kind of software and shows that the few extra basic arithmetic operations needed by quaternions are preferable to the complexities introduced by Euler rotations. The significant difference between the systems is the avoidance of trigonometric functions and reduced data access overhead (see Table 1). The overhead of loading the program, molecule data, scoring function look-up tables, etc. is not included in the times, so that the effects of disk transfer rates and caching are eliminated as far as possible. Similarly, all result recording and RMSD calculations were performed at the end of the process, and also excluded from the time counted.

In virtual screening applications, the desired output is a substantially reduced and ranked set of potential complexes, which should include several good poses. It is important that any changes in models or algorithms should not affect the pattern of results from a redocking test, since this would indicate that the changes had disrupted the search method's ability to explore conformation space. One way to verify this is to compare the distribution of RMSDs from the native pose in the result sets. When all the results were combined, their cumulative RMSD distributions were as shown in Figure 4. The two methods are virtually identical, allowing for the slight variation inherent with stochastic methods, confirming that the improvement in run time was achieved without detriment to the usefulness of the docking system.

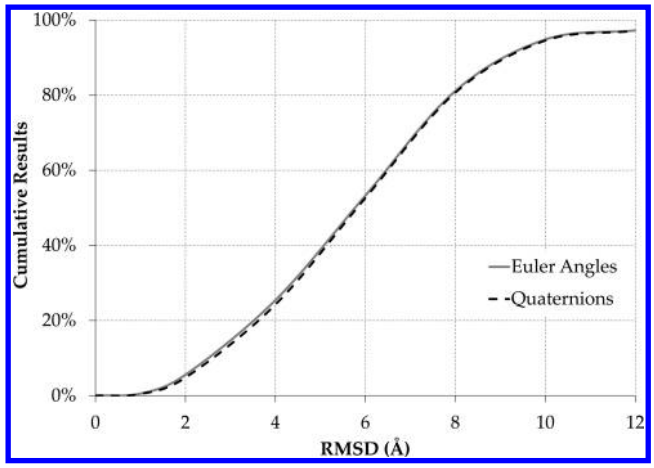


Figure 4. Resulting pose RMSDs from the two rotation representations. Cumulative frequency of poses within a given RMSD of its true native binding pose, calculated over the combined set of 51 000 results (100 from each of 6 iterations of 85 cases).

CONCLUSIONS

This paper has presented a careful comparison of the use of two rotation representations in a docking program for virtual screening. When using a simple genetic algorithm to search pose space, quaternions are significantly faster than Euler angles, and entirely interchangeable.

If a docking procedure is to be part of a high-throughput application pipeline, speed is a key factor in its usefulness. By merely changing some underlying mathematics, an increase is achieved easily in the capacity of any system that processes geometric data repeatedly, independently of the other algorithms involved. In addition, the ability to compose and invert cases without conversion to other forms provides a consistent model for sophisticated integration with other algorithms, such as in a pipeline. Quaternions are thus recommended strongly for use in three-dimensional simulation applications, and particularly for molecular modeling as demonstrated here.

AUTHOR INFORMATION

Corresponding Authors

*E-mail: gwyn@delphicorange.co.uk

*E-mail: stephen.cameron@cs.ox.ac.uk.

Notes

The authors declare no competing financial interest.

ACKNOWLEDGMENTS

The authors thank InhibOx Ltd. for funding and the use of DOX's source code as a foundation project.

REFERENCES

(1) Connolly, M. Molecular Surfaces: A Review. Network Science, 1996; http://www.netsci.org/Science/Compchem/feature14.html (accessed Nov 17 2013).

(2) Xu, D.; Xu, Y.; Uberbacher, E. Computational Tools For Protein Modelling. *Curr. Protein Pept. Sci.* **2000**, *1*, 1−21.

(3) Cavasotto, C.; Orry, A. Ligand Docking and Structure-based Virtual Screening in Drug Discovery. *Curr. Topics Med. Chem.* **2007**, *7*, 1006−1014.

(4) Putta, S.; Beroza, P. Shapes of Things: Computer Modeling of Molecular Shape in Drug Discovery. *Curr. Topics Med. Chem.* **2007**, *7*, 1514−1524.

(5) McGaughey, G.; Sheridan, R.; Bayly, C.; Culberson, J.; Kreatsoulas, C.; Lindsley, S.; Maiorov, V.; Truchon, J-F..; Cornell, W. Comparison of Topological, Shape, and Docking Methods in Virtual Screening. *J. Chem. Inf. Model.* **2007**, *47*, 1504−1519.
(6) Sanner, M.; Olson, A.; Spehner, J-C. Fast and Robust Computation of Molecular Surfaces. *Proceedings of the 11th Annual Symposium on Comp. Geometry*, Vancouver, BC, Canada, June 5−7, 1995; pp 406−407.
(7) Goldman, B.; Wipke, W. Quadratic Shape Descriptors. 2. Molecular Docking Using Quadratic Shape Descriptors (QSDock). *Proteins* **2000**, *38*, 79−94.
(8) Mandell, J.; Roberts, V.; Pique, M.; Kotlovyi, V.; Mitchell, J.; Nelson, E.; Tsigelny, I.; Eyck, L. T. Protein docking using continuum electrostatics and geometric fit. *Protein Eng.* **2001**, *14*, 105−113.
(9) Sotriffer, C.; Gohlke, H.; Klebe, G. Docking into Knowledge-Based Potential Fields: A Comparative Evaluation of DrugScore. *J. Med. Chem.* **2002**, *45*, 1967−1970.
(10) Goldstein, H.; Poole, C.; Safko, J. *Classical Mechanics*, 3rd ed.; Addison Wesley, 2001.
(11) Karney, C. Quaternions in Molecular Modeling. *J. Mol. Graph. Model.* **2006**, *25*, 595−604.
(12) Hamilton, W. In *The Mathematical Papers of Sir William Rowan Hamilton*; Halberstam, H., Ingram, R., Eds.; Cambridge University Press, 1967.
(13) Shoemake, K. In *Graphics Gems III*; Kirk, D., Ed.; Academic Press, 1992; pp 124−132.
(14) Intel Corp. Intel Microprocessor Export Compliance Metrics. 2013; http://www.intel.com/support/processors/sb/cs-017346.htm (accessed Nov 17 2013).
(15) InhibOx Ltd. DOX: Ligand-Protein Docking. 2009; http://www.inhibox.com/docking (accessed Nov 17 2013).
(16) Skone, G. *Stratagems for Effective Function Evaluation in Computational Chemistry*. Ph.D. thesis, University of Oxford, 2010.
(17) Nelder, J.; Mead, R. A Simplex Method for Function Minimization. *Comput. J.* **1965**, *7*, 308−313.
(18) Glover, F. Tabu Search − Part I. *ORSA J. Comput.* **1989**, *1*, 190−206.
(19) Kirkpatrick, S.; Gelatt, C.; Vecchi, M. Optimization by Simulated Annealing. *Science* **1983**, *220*, 671−680.
(20) Kennedy, J.; Eberhart, R. Particle Swarm Optimization. *Proceedings IEEE International Conference Neural Networks*, Perth, Australia, Nov 27−Dec 1, 1995; pp 1942−1948.
(21) Karaboga, D.; Basturk, B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *J. Global Opt.* **2007**, *39*, 459−471.
(22) Jones, G.; Willett, P.; Glen, R.; Leach, A.; Taylor, R. Development and Validation of a Genetic Algorithm for Flexible Docking. *J. Mol. Biol.* **1997**, *267*, 727−748.
(23) Wall, M. *GAlib*. Massachusetts Institute of Technology, 1996; http://lancet.mit.edu/ga (accessed Nov 17 2013).
(24) Wang, R.; Lai, L.; Wang, S. Further development and validation of empirical scoring functions for structure-based binding affinity prediction. *J. Comput. Aided Mol. Des.* **2002**, *16*, 11−26.
(25) Wang, R. *X-Score*. University of Michigan, 2003; http://sw16.im.med.umich.edu/software/xtool (accessed Nov 17 2013).
(26) Miles, R. On Random Rotations in R^3. *Biometrika* **1965**, *52*, 636−639.
(27) Hartshorn, M.; Verdonk, M.; Chessari, G.; Brewerton, S.; Mooij, W.; Mortenson, P.; Murray, C. Diverse, High-Quality Test Set for the Validation of Protein-Ligand Docking Performance. *J. Med. Chem.* **2007**, *50*, 726−741.