Journal of Computer-Aided Molecular Design (2005) 19: 293–299
DOI 10.1007/s10822-005-5122-2

Implementation of smooth continuous camera trajectories for viewing PDB and VRML objects

Milan Pahor*
School of Mathematics, University of NSW, 2052, Sydney NSW, Australia

Accepted in revised form 5 April 2005

Key words: animation, camera, PDB, protein, trajectory, VRML

Summary

A parametrically defined camera trajectory enabling the accurate and systematic scanning of the entire surface of virtual (Protein Data Bank (PDB)) proteins is developed and implemented. The smooth continuous path guarantees that each local region of the protein is inspected from a variety of directions in a controlled and uniform manner. Manipulation of several parameters governs the density, character and duration of the scan. Applications to the analysis of other real and virtual objects are also considered.

Introduction

The use of interactive computer visualisations to analyse and investigate three-dimensional objects of scientific interest is a field of intense activity and innovation. Through the use of Virtual Reality Modelling Language (VRML), Java, Flash and various specialised graphics applications (3D Studio Max and AutoCad for example) it is possible to both produce and manipulate virtual three dimensional (3D) computer models across a broad spectrum of scientific, engineering and mathematical disciplines.

Applications range from the analysis of brain structures through neuroimaging [1], virtual imaging of organs prior to surgery [2], interactive visualisations of crystal packing [3], to virtual support for the Mars Pathfinder mission [4].

Perhaps the most spectacular success has been the Protein Data Bank (PDB) [5], a worldwide repository of experimentally determined biological structure data from which interactive 3D atom-level visualisations of macromolecular protein structure may be rendered. It has long been understood that the 3D shape of a protein dictates and defines its biological function, thus it is critical that protein analysis takes into account the molecular structure in all 3D. Specialised software packages such as RASMOL [6] transform the PDB data into an immense catalogue of interactive protein visualisations. Alternatively the PDB data may be converted into the generic VRML format and then visualised as a VRML world using browser plug-ins such as *Cortona*.

The protein can then be rotated in all directions in space, enabling a complete and authentic analysis of the structure of the molecule. However the inspection process is often completely open ended with the user needing to scan the entire surface of the object from as many different points of view as possible in order to discover and then investigate interactions and subtle variations in structure (Figure 1).

The problem

Due to the freedom that exists in the choice of viewpoint it is difficult to track which parts of the molecule have been investigated. Furthermore it is

*To whom correspondence should be addressed. E-mail: pahor@maths.unsw.edu.au

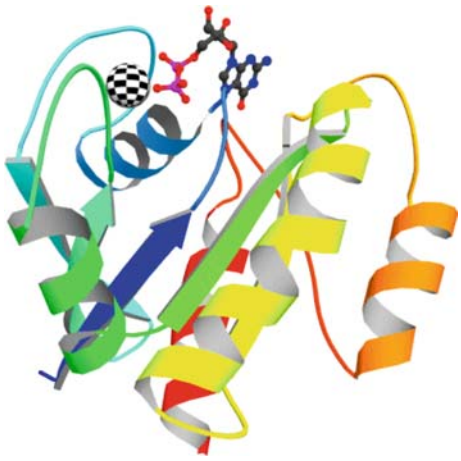


Figure 1. An example of a visualisation rendered from the PDB (PDB code 1CRQ) is the Ras p21 protein with structure determined by NMR spectroscopy [7]. Note that some standard colours have been modified to clarify the structures in animation.

necessary to monitor the *orientation* of the observations over any particular point since structures, which are clear when viewed from east to west, may be obscured if approached from north to south. The problem then is as follows:

Is it possible to manage the viewpoint (referred to henceforth as the camera) in such a manner that:

- Every region on the surface of the protein is eventually scanned to a controllable level of precision.
- For each region the observations are made from a variety of different directions.
- The scanning process is both smooth and continuous, minimising dislocation and maintaining context.
- It is the issue of protein analysis that motivates the constructions of this article, however the techniques outlined below will apply to any bounded object in space (real or virtual) whose surface needs to be scanned with precision. The mathematical constructions are not deep and only use the relatively simple concepts of spherical coordinates and rotation matrices.

Defining the camera trajectory

Our approach is to define the camera trajectory as a parametric path over the surface of the unit sphere. The protein is embedded within the sphere and the scanning camera inspects the protein with a controlled glide over the surface of the enveloping sphere. Most viewing software will also provide the opportunity to have the camera *target* move over an independent path. Options for the target path will be discussed in the next section however for the moment we will assume that the protein is placed at the origin and that the camera targets the origin at all times. We will also assume that the complete analysis takes place over an interval of T seconds.

The final path is the composition of two independent trajectories. The first (the spiral trajectory) guarantees that the path passes sufficiently close to all the regions. The second component (the daisy trajectory) generates a multiplicity of scanning directions for any given region.

The spiral trajectory

The standard equations in spherical coordinates defining the surface of the unit sphere are

$$\left\{\begin{array}{l} x = \sin\phi\cos\theta \\ y = \sin\phi\sin\theta \\ z = \cos\phi \end{array}\right\} \phi : 0 \to \pi, \quad \theta : 0 \to 2\pi,$$

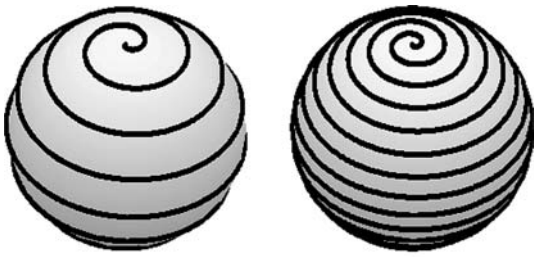
where the angle ϕ is measured off the positive z-axis and θ from the positive x-axis (when viewed from above).

(It should be noted that the role of θ and ϕ are often swapped in many sources).

To generate a path from the north pole ($\phi = 0$) to the equator ($\phi = \frac{\pi}{2}$) over the interval of time $t : 0 \to T$ we set $\phi = \frac{\pi t}{2T}$. In order that the camera make n *rotations* about the vertical axis in the same interval of time we set $\theta = \frac{2n\pi t}{T}$ This generates the following spiral path from north pole to equator:

$$\left\{\begin{array}{l} x = \sin\left(\dfrac{\pi t}{2T}\right)\cos\left(\dfrac{2n\pi t}{T}\right) \\ y = \sin\left(\dfrac{\pi t}{2T}\right)\sin\left(\dfrac{2n\pi t}{T}\right) \\ z = \cos\left(\dfrac{\pi t}{2T}\right) \end{array}\right\} t : 0 \to T$$

The following two diagrams display the spiral paths for $n = 4$ and $n = 8$ as $t : 0 \to 2T$ (thus producing the entire sphere). Observe that the density of the path increases with n. MAPLE



n=4 n=8

animations of the spiral path can be found at www.maths.unsw.edu.au/~pahor.

Clearly a camera moving along the path from north pole to south pole will scan all regions of the sphere, with precision easily regulated by making appropriate choices of n. This trajectory however only visits each region from west to east, and thus provides no variety in the orientation of the viewpoint. To generate this character of diversity we will compose the spiral path with the following daisy path.

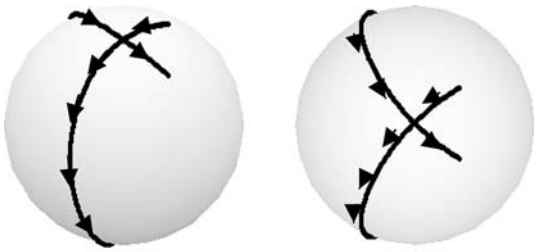
The daisy trajectory

The daisy path is designed to approach the *poles* from a multitude of different directions. Suppose that the observation takes place over W seconds and consider again the parametric equations defining the surface of the unit sphere:

$$\left\{\begin{array}{l} x = \sin\phi\cos\theta \\ y = \sin\phi\sin\theta \\ z = \cos\phi \end{array}\right\} \phi : 0 \to \pi, \quad \theta : 0 \to 2\pi.$$

Setting $\phi = \frac{2m\pi t}{W}$as $t : 0 \to W$ guarantees m (essentially vertical) loops over the time interval. Simultaneously we demand that $\phi = \frac{2\pi t}{W}$ as $t : 0 \to W$ so that the camera completes one horizontal rotation as the m vertical loops are produced. The final equations are:

$$\left\{\begin{array}{l} x = \sin\left(\dfrac{2m\pi t}{W}\right)\cos\left(\dfrac{2\pi t}{W}\right) \\ y = \sin\left(\dfrac{2m\pi t}{W}\right)\sin\left(\dfrac{2\pi t}{W}\right) \\ z = \cos\left(\dfrac{2m\pi t}{W}\right) \end{array}\right\} t : 0 \to W$$

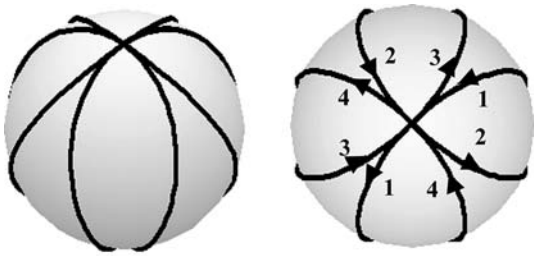


PERSPECTIVE PLAN

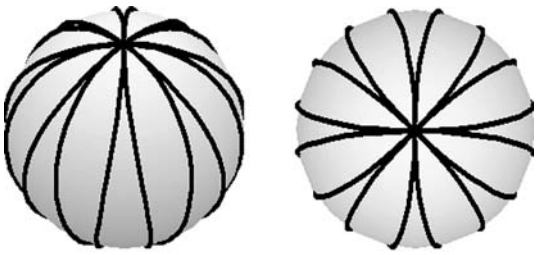
Consider the following perspective and plan view of one quarter of the path when $m = 4$.

This displays one complete loop from north pole to south pole returning to the north pole. A crucial feature to note is that the path from pole to pole is *twisted* with the direction of the second approach to the north pole incremented by 90°. As the path evolves the north pole will be approached from four equally spaced directions.

Complete perspectives and plans for $m = 4$ and $m = 8$ presented below.



PERSPECTIVE PLAN



PERSPECTIVE PLAN

For any given m there are m passes over each of the poles with each scan approaching (and departing) the pole along m unique evenly spaced "longitudes". MAPLE animations of the spiral trajectory can be found at www.maths.unsw.edu.au/~pahor.

In summary the spiral trajectory covers the surface of the sphere effectively but is deficient in its variety of approaches to any particular point while the daisy path scans in a multiplicity of different directions but does so only at the poles of the sphere. Our final scanning trajectory will merge these critical features into one path.

The scanning trajectory

We are now in a position to define the final scanning trajectory. First we present a simple lemma, which will help to combine the paths.

Lemma 1

Suppose that $\begin{pmatrix} a \\ b \\ c \end{pmatrix}$ is a point on the surface of the unit sphere. Then the matrix of rotation mapping $\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$ to $\begin{pmatrix} a \\ b \\ c \end{pmatrix}$ is given by

$$R = \begin{pmatrix} \frac{a^2c+b^2}{a^2+b^2} & \frac{ab(c-1)}{a^2+b^2} & a \\ \frac{ab(c-1)}{a^2+b^2} & \frac{b^2c+a^2}{a^2+b^2} & b \\ -a & -b & c \end{pmatrix}$$

Proof

It is easily checked that det $R = 1$ that $R^TR = I$ and that $R\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$, implying that R is the required matrix of rotation. However a constructive proof is more revealing and is presented in Appendix 1.

The strategy is to now slide the daisies down the spiral trajectory *as they are produced*. Note that since the daisy paths simultaneously cover both poles we need only transform their production from north pole to equator and hence $t : 0 \to T$ is sufficient. Note also that this process will slightly deform their structure (there will no longer be a well defined centre). However each local region will still be visited from m different directions provided that sufficiently many daisies are produced in the transition from pole to equator. The production of p daisies over the spiral path is achieved by setting W to be equal to $\frac{T}{p}$.

The final parametric equation of the path $\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$ can thus be obtained by setting $\left\{ \begin{array}{l} a = \sin\left(\frac{\pi t}{2T}\right)\cos\left(\frac{2n\pi t}{T}\right) \\ b = \sin\left(\frac{\pi t}{2T}\right\}\sin\left(\frac{2n\pi t}{T}\right) \\ c = \cos\left(\frac{\pi t}{2T}\right) \end{array} \right\}$ in the above rotation matrix R to produce the rotation matrix $R(t)$ which rotates the north pole to points along the spiral trajectory. Rotating the daisies into position yields

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = R(t)\begin{pmatrix} \sin\left(\frac{2m\pi t}{W}\right)\cos\left(\frac{2\pi t}{W}\right) \\ \sin\left(\frac{2m\pi t}{W}\right)\sin\left(\frac{2\pi t}{W}\right) \\ \cos\left(\frac{2m\pi t}{W}\right) \end{pmatrix}$$
$$= R(t)\begin{pmatrix} \sin\left(\frac{2mp\pi t}{T}\right)\cos\left(\frac{2p\pi t}{T}\right) \\ \sin\left(\frac{2mp\pi t}{T}\right)\sin\left(\frac{2p\pi t}{T}\right) \\ \cos\left(\frac{2mp\pi t}{T}\right) \end{pmatrix} t : 0 \to T.$$

Evaluation and simplification of the matrix product results in the following final parametric equation for the desired path:

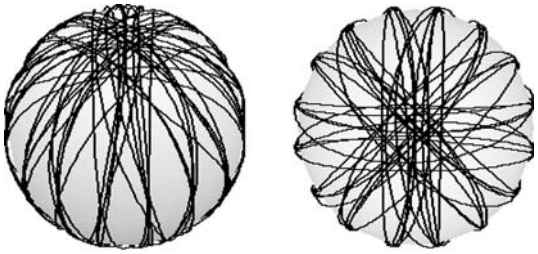
$$\begin{aligned} X(t) = & \left\{ \cos^2\left(\frac{2n\pi t}{T}\right)\cos\left(\frac{\pi t}{2T}\right) \right. \\ & \left. + \sin^2\left(\frac{2n\pi t}{T}\right) \right\} \sin\left(\frac{2mp\pi t}{T}\right)\cos\left(\frac{2p\pi t}{T}\right) \\ & + \cos\left(\frac{2n\pi t}{T}\right)\sin\left(\frac{2n\pi t}{T}\right)\left\{\cos\left(\frac{\pi t}{2T}\right) - 1\right\} \\ & \times \sin\left(\frac{2mp\pi t}{T}\right)\sin\left(\frac{2p\pi t}{T}\right) \\ & + \sin\left(\frac{\pi t}{2T}\right)\cos\left(\frac{2n\pi t}{T}\right)\cos\left(\frac{2mp\pi t}{T}\right) \end{aligned}$$

$$\begin{aligned} Y(t) = & \left\{ \sin^2\left(\frac{2n\pi t}{T}\right)\cos\left(\frac{\pi t}{2T}\right) + \cos^2\left(\frac{2n\pi t}{T}\right) \right\} \\ & \times \sin\left(\frac{2mp\pi t}{T}\right)\sin\left(\frac{2p\pi t}{T}\right) \\ & + \cos\left(\frac{2n\pi t}{T}\right)\sin\left(\frac{2n\pi t}{T}\right)\left\{\cos\left(\frac{\pi t}{2T}\right) - 1\right\} \\ & \times \sin\left(\frac{2mp\pi t}{T}\right)\cos\left(\frac{2p\pi t}{T}\right) \\ & + \sin\left(\frac{\pi t}{2T}\right)\sin\left(\frac{2n\pi t}{T}\right)\cos\left(\frac{2mp\pi t}{T}\right) \end{aligned}$$

$$Z(t) = -\sin\left(\frac{\pi t}{2T}\right) sin\left(\frac{2mp\pi t}{T}\right)\cos\left(\frac{2\pi t}{T}(n-p)\right) + \cos\left(\frac{\pi t}{2T}\right)\cos\left(\frac{2mp\pi t}{T}\right)$$

$t : 0 \to T$ This path is continuous and smooth. Furthermore suitable choice of parameters *n, m* and *p* regulate the precision of the scan. Recall that *n* controls the density of the spiral path, *m* is the number of petals in the daisy trajectory, and *p* is the number of daisies laid out along the length of the spiral path.

Consider the following perspective and plan for the case where $n = 5$ (spins of the spiral), $m = 8$ (petals per daisy), $p = 21$ (daisies per spiral) and $t : 0 \to \frac{T}{4}$. This will clarify the early stages of the scan by only presenting a quarter ($\sim$5) of the daisies:



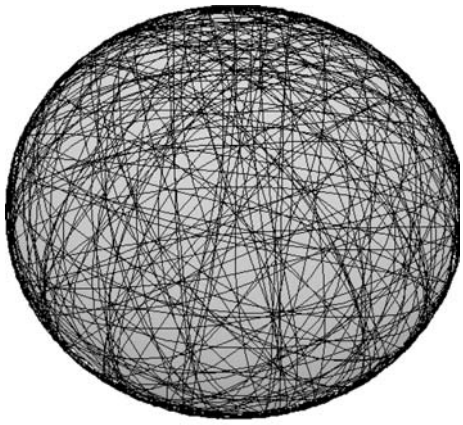
PERSPECTIVE **PLAN**

Observe that the daisies are *not* produced over the top of each other but rather are being drawn down the spiral path. One quarter of the way through the scan we already have excellent coverage of both polar regions, however the equator still remains to be fully explored.

The complete trajectory $t : 0 \to T$ is presented below from the same perspective. Observe the remarkable uniformity and density of the coverage.

MAPLE animations of various stages of the evolution of the final path can be found at www.maths.unsw.edu.au/$\sim$pahor.

The above trajectory satisfies our criteria. It is smooth, continuous, scans the entire sphere to a controllable level of precision and visits each local region from a variety of different directions. We turn now to technical considerations that need to be addressed if these paths are to be used as camera trajectories for the inspection of virtual objects. Each different piece of viewing software will deal with camera, trajectory and target issues differently so the discussion is best held at a generic level.



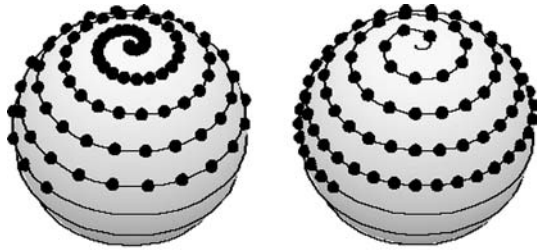
n=5, m=8, p=21

Technical issues

1. If the object being scanned is roughly spherical of radius α we simply need to place it at the origin and assign a camera trajectory of $r(t) = \begin{pmatrix} \alpha X(t) \\ \alpha Y(t) \\ \alpha Z(t) \end{pmatrix}$ where X, Y and Z are the final path components defined above and α is sufficiently large so as to enable the camera to view the entire object. For objects with an irregular shape dilate the path components by using $r(t) = \begin{pmatrix} \alpha X(t) \\ \beta Y(t) \\ \gamma Z(t) \end{pmatrix}$ with appropriate α, β and γ. This will transform the enveloping sphere into an ellipsoid with semi-major axes of length α, β and γ.
2. In most applications there will be an opportunity to assign the camera *target* to a path as well as the camera itself. A simple option is to target the origin (that is the centre of the object), however it is also possible to point the camera tangentially *along* the path. This is achieved by using the target path $R(t) = \begin{pmatrix} X(t+\alpha) \\ Y(t+\alpha) \\ Z(t+\alpha) \end{pmatrix}$. That is point the camera

to a target α seconds further down the path than the camera itself where α is reasonably small. The use of $-\alpha$ will force the camera to look backwards along the path instead of forwards.

3. In order to reduce the possibility that the various rotations in the definition of the trajectory interact in unexpected ways it is best to choose m, n and p to be relatively prime with $p >> n$. Typical choice would be $m = 4$, $n = 3$ and $p = 11$. That is 3 rotations on the spiral path, 4 petals per daisy and 11 daisies in total. Note that there will be mp orbits in total, so in this case the path will encircle the sphere 44 times. Assigning 10 s to each orbit for example, yields a total animation length of $T = 440$ s.
4. Over-scanning at the poles can be averted by simply rescaling the time variable from t to $\sqrt{t}$. To achieve this without affecting the duration T of the scan replace all occurrences of t in the path equations with $\sqrt{tT}$. The two sketches below display the position of the camera on the spiral path at each second of a 100 s scan from north pole to equator.



5. With some pieces of software the camera viewpoint will spin wildly if the vector from the camera to target becomes vertical. This can be minimised by swapping the X and Z coordinates of the scanning trajectory so that the scan operates from left to right rather than top to bottom. Alternatively Euler angles or Quaternions can be used to generate an equivalent rotation of the object itself under a *fixed* camera.
6. Although the final formulae are complicated there are only a few different terms so cutting and pasting can be used to enter the equations into software packages. Alternatively the equations can be constructed within the software by entering and implementing the rotation matrix $R(t)$. It should be noted however that the final equations above have already been streamlined by significant algebraic simplification.
7. Choices of input and output formats will depend upon the needs of the practitioner and the software being used. Sophisticated packages such as 3D Studio Max will import protein visualisations in VRML format and easily assign a camera trajectory to any parametrically defined path. A simple and workable alternative is to *directly* encode the camera trajectory in VRML.
8. Outputting to movie formats such as Quicktime and Avi will produce stable results across different machines but the file size is large. The production of such bitmap based animations is time consuming and requires significant computing power. An AVI displaying a partial examination (35 s with $m = 4$, $n = 3$ and $p = 11$) of the protein Ras p21 (PDB code 1CRQ) [7] can be found at www.maths.unsw.edu.au/∼pahor. Note that some of the usual colours have been altered to clarify the trajectory. Once created the animation is of course fixed hence the movie formats are useful for demonstrations but do not provide the user with an interactive environment. They do however produce a high quality render.
9. Interactive formats such a VRML and RASMOL provide a far greater degree of flexibility. The animation can be paused at any stage with the user enjoying the freedom to make further investigations in any direction in real time. Furthermore a single file may access a multitude of different animated and fixed cameras viewpoints. File sizes are small and the underlying code is easily manipulated.
10. In order to analyse the computational loading of our path we consider the performance of a VRML animation of the Ras p21 protein [7] using a camera trajectory with $n = 5$, $m = 8$ and $p = 21$. The VRML file drawn from the PDB (Code: 1CRQ) also contained a number of pre-determined fixed camera viewpoints and was played on the Cortona viewer. The animation was run full screen at

30 frames per second at a resolution of 1024×768 in 32 bit colour on two systems.

System 1: PENTIUM 4, 2.8 GHz, 1 GB RAM, 32 MB VIDEO, WINDOWS 2000 CPU LOADING: 55%

System 2: PENTIUM 3, 650 MHz, 256 MB RAM, 16 MB VIDEO, WINDOWS 2000 CPU LOADING: 85%

Both animations ran smoothly with effortless transitions between animated and fixed viewpoints. It is significant that a second VRML file of the same protein rotated only about the vertical axis yielded almost identical loadings. The task of computing the camera path compares favourably with that of a simple rotation and is a small burden when compared with that of producing the actual render.

Many visualisation packages already use a simple rotation about a single axis as a viewing platform. See for example the PDB to Multigif viewer [8] or the "rock n roll" feature in RASMOL [6]. The next generation of these programs may instead implement the camera trajectory of this article as a sophisticated foundation from which to further explore the protein's structure.

Acknowledgements

The initial enquiries for this work stemmed directly from discussions during my participation in the UNSW Innovative Teaching and Educational Technology (ITET) fellowship scheme. My thanks go to Prof. Adrian Lee and all the ITET fellows for their inspiration and support.

Appendix 1 (Constructive proof of Lemma 1)

Observe that $\left\{ \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} \frac{a}{\sqrt{a^2+b^2}} \\ \frac{b}{\sqrt{a^2+b^2}} \\ 0 \end{pmatrix} \right\}$ is an orthonormal basis for $\operatorname{span}\left\{ \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} a \\ b \\ c \end{pmatrix} \right\}$, which is the two dimensional subspace in which the rotation is implemented. Furthermore $\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \times \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} -b \\ a \\ 0 \end{pmatrix}$ yielding an orthonormal basis $B = \left\{ \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} \frac{a}{\sqrt{a^2+b^2}} \\ \frac{b}{\sqrt{a^2+b^2}} \\ 0 \end{pmatrix}, \begin{pmatrix} \frac{-b}{\sqrt{a^2+b^2}} \\ \frac{a}{\sqrt{a^2+b^2}} \\ 0 \end{pmatrix} \right\}$ for $\Re^3$. The orthogonal matrix $Q = \begin{pmatrix} 0 & \frac{a}{\sqrt{a^2+b^2}} & \frac{-b}{\sqrt{a^2+b^2}} \\ 0 & \frac{b}{\sqrt{a^2+b^2}} & \frac{a}{\sqrt{a^2+b^2}} \\ 1 & 0 & 0 \end{pmatrix}$ serves as a matrix of change of basis from B to $\Re^3$.

The angle of rotation is given by $\cos(\alpha) = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} a \\ b \\ c \end{pmatrix} = c$ (since both vectors are of unit length). Thus the matrix which implements the rotation respect to basis B is Hence the required rotation matrix is $R = QUQ^{-1} = \begin{pmatrix} \frac{a^2c+b^2}{a^2+b^2} & \frac{ab(c-1)}{a^2+b^2} & a \\ \frac{ab(c-1)}{a^2+b^2} & \frac{b^2c+a^2}{a^2+b^2} & b \\ -a & -b & c \end{pmatrix}$ as required.

References

1. Nielsen, F.A. and Hansen, L.K., Visualization Development Environments 2000 Proceedings (VDE2000). 27–28 (2000) 76–81.
2. Holten-Lund, H., Hvidtfeldt, M., Madsen, J. and Pedersen, S., Proceedings of the Web3D-VRML 2000 Fifth Symposium on Virtual Reality Modelling Language. 2000, pp. 111–118.
3. Fu, T.Y. and Chen, Y.W., J. Appl. Cryst., 29 (1996) 594.
4. N.A.S.A., http://marsprogram.jpl.nasa.gov/MPF/vrml/vrml.html.
5. Berman, H.M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T.N., Weissig, H., Shindyalov, I.N. and Bourne, P.E., Nucleic Acids Res., 28 (2000) 235.
6. Sayle, R. and James Milner-White, E., Trends Biochem. Sci.,, 20 (1995) 374.
7. Kraulis, P.J., Domaille, P.J., Campbell-Burk, S.L., Van Aken, T. and Laue, E.D., Biochemistry, 33 (1994) 3515.
8. Bohne, A., J. Mol. Model., 4 (1998) 344.