Proceedings of the 7th Asian Technology Conference in Mathematics 2002.
Invited Paper

Quaternions: From Classical Mechanics to Computer Graphics, and Beyond

R. Mukundan
Department of Computer Science
University of Canterbury
Christchurch, New Zealand.

mukund@cosc.canterbury.ac.nz

Abstract:

The theory of quaternions was introduced in the mid nineteenth century, and it found many applications in classical mechanics, quantum mechanics, and the theory of relativity. Quaternions were also later used in aerospace applications and flight simulators, particularly when inertial attitude referencing and related control schemes where employed. However, it is only in the recent past that graphics and game programmers discovered the true potential of quaternions and started using it as a powerful tool for describing rotations about an arbitrary axis. From computer graphics, the application domain of quaternions soon expanded into other fields such as visualization, fractals and virtual reality.

This paper provides an overview of the various analytical properties of quaternions and their usefulness in the areas of computer/robot vision, computer graphics and animation. Particular emphasis is given to vision algorithms for 3D pose estimation, animation techniques involving viewpoint/object rotations, motion interpolation algorithms, and quaternion fractals. The benefit of using quaternions over other representations such as Euler angles is not just limited to singularity free kinematics relations – Quaternions allow us to derive closed form solutions for algebraic systems involving unknown rotational parameters. Some of the neat mathematical characteristics of quaternions in the complex space together with a set of useful formulae are included for the benefit of the mathematically inclined.

1 Introduction

The discovery of quaternions is attributed to the Irish mathematician William Rowan Hamilton (1805-1865) whose persistent struggle to extend the notion of complex numbers represented as algebraic pairs to triplets, finally gave birth to a very interesting non-commutative algebra[1,2]. Mathematicians hailed the discovery as one of the three highly significant developments in the nineteenth century, the other two being the developments of the non-Euclidean geometry and the theoretical foundation for calculus. Even though quaternions remained a "solution in search of a problem" for many years after their discovery, applications in the fields of classical mechanics and the theory of relativity were identified in the early twentieth century. The capability of quaternions

to succinctly represent three-dimensional rotations about an arbitrary axis motivated researchers to employ quaternion algebra in rotational kinematics equations. As a result, several new application areas involving quaternion based algorithms emerged [3]. These include diverse fields such as robotics, orbital mechanics, aerospace technologies, and inertial navigation systems.

The motivation for this paper is derived from the recent developments in the fields of computer graphics and games programming where quaternions have been very effectively used. Graphics programmers have now realized the potential of quaternions as a very general and powerful rotation operator. Recent graphics APIs provide functions for quaternion operations. For example, the Java-3D API has classes (javax.vecmath.Quat4d) for creating quaternion objects, Povray (Persistence of Vision Ray Tracer) supports quaternions, Quickdraw 3D provides routines for quaternion operations, and Mathematica has an add-on package "Algebra `Quaternions`". Quaternions are also used in advanced algorithms in games programming and animation such as keyframe interpolations and the simulation of camera motion in a three-dimensional space [4, 5]. Some of the virtual world interaction devices employ quaternions to parameterize rotations (Logitech 3D mouse has a quaternion output mode).

The paper is organized as follows. The next section introduces quaternions which form the only non-commutative example of a real division algebra. Section 3 introduces the quaternion rotation operator, which has found several uses in classical mechanics and computer graphics. Applications of quaternions in computer and robot vision in three-dimensional pose estimation are outlined in Section 4. Section 5 compares parameterization of rotations using Euler angles and quaternions, and explains why quaternions have become popular in the field of computer graphics. This section also introduces the spherical linear interpolation method, which is being increasingly used by animators. Some graphics related applications in fractals are also given. As a whole, the paper is intended to provide a comprehensive understanding of the quaternion algebra and the way its application domain is currently expanding, with particular emphasis on the utility aspects from a graphics perspective.

2 Quaternion Algebra

Quaternions are hyper-complex numbers of rank 4, constituting a four dimensional vector space over the field of real numbers [2]. We use the following four-tuple notation to represent a quaternion:

$$\begin{aligned}\mathbf{q} &= (q_0 , q_1, q_2, q_3) \\ &= (q_0 , \boldsymbol{w}) \\ &= q_0 + \mathbf{i}q_1 + \mathbf{j}q_2 + \mathbf{k}q_3\end{aligned} \tag{1}$$

where q_0 is the scalar component of $\mathbf{q}$, and $\boldsymbol{w}=\{q_1, q_2, q_3\}$ form the vector part, and the entire set of $\mathbf{q}$'s is spanned by the basis quaternions:

$$\begin{aligned}1 &= (1, 0, 0, 0) \\ \mathbf{i} &= (0, 1, 0, 0) \\ \mathbf{j} &= (0, 0, 1, 0) \\ \mathbf{k} &= (0, 0, 0, 1)\end{aligned} \tag{2}$$

The orthonormal basis components **i**, **j**, **k**, as defined above satisfy the following well-known rules:

$$\begin{aligned}
&\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1 \\
&\mathbf{ij} = -\mathbf{ji} = \mathbf{k} \\
&\mathbf{jk} = -\mathbf{kj} = \mathbf{i} \\
&\mathbf{ki} = -\mathbf{ik} = \mathbf{j}
\end{aligned} \tag{3}$$

where it is implied that the product of any two terms in the above expressions is indeed a *quaternion product*, which is defined in the most general form for two quaternions $\mathbf{p} = (p_0, \boldsymbol{v})$, and $\mathbf{q} = (q_0, \boldsymbol{w})$ as

$$\mathbf{pq} = (p_0 q_0 - \boldsymbol{v}.\boldsymbol{w} , \quad p_0 \boldsymbol{w} + q_0 \boldsymbol{v} + \boldsymbol{v}\times\boldsymbol{w}) \tag{4}$$

where $\boldsymbol{v}.\boldsymbol{w}$ and $\boldsymbol{v}\times\boldsymbol{w}$ denote the familiar dot and cross products respectively, between the three-dimensional vectors $\boldsymbol{v}$ and $\boldsymbol{w}$. Obviously, quaternion multiplication is not commutative, since the vector cross product is not. The set of elements $\{\pm 1, \pm\mathbf{i}, \pm\mathbf{j}, \pm\mathbf{k}\}$ form a group (known as the *quaternion group*) of order 8 under multiplication. Quaternion addition is a relatively simpler operation consisting of the addition of the corresponding components.

$$\mathbf{p} + \mathbf{q} = (p_0 + q_0 , \boldsymbol{v} + \boldsymbol{w}) \tag{5}$$

Quaternions also form a commutative group under addition, (0,0,0,0) being the identity element. Quaternion multiplication is associative, and distributes over addition:

$$\begin{aligned}
&(\mathbf{pq})\mathbf{r} = \mathbf{p}(\mathbf{qr}) \\
&(\mathbf{p} + \mathbf{q})\mathbf{r} = \mathbf{pr} + \mathbf{qr} \\
&\mathbf{p}(\mathbf{q} + \mathbf{r}) = \mathbf{pq} + \mathbf{pr}
\end{aligned} \tag{6}$$

As in the case of ordinary complex numbers, we can define the conjugate $\mathbf{q}^*$ of a quaternion $\mathbf{q} = (q_0, \boldsymbol{w})$ as follows:

$$\mathbf{q}^* = (q_0, -\boldsymbol{w}) = (q_0 , -q_1, -q_2, -q_3) \tag{7}$$

The above equation allows us to define the quaternion norm $\|\mathbf{q}\|$ as

$$\|\mathbf{q}\|^2 = \mathbf{q}\mathbf{q}^* = \quad q_0^2 + q_1^2 + q_2^2 + q_3^2 \tag{8}$$

In particular, a *unit quaternion* is the one for which $\|\mathbf{q}\| = 1$. The components of a unit quaternion will all have an absolute value less than or equal to 1. Any quaternion **q** can be normalized by dividing by its norm, to obtain a unit quaternion. The *inverse of a quaternion* given by

$$\mathbf{q}^{-1} = \frac{1}{\|q\|^2}\, \mathbf{q}^* , \qquad \|\mathbf{q}\| \neq 0. \tag{9}$$

satisfies the relation $\mathbf{q}\,\mathbf{q}^{-1} = \mathbf{q}^{-1}\mathbf{q} = 1$. Since every non-zero quaternion **q** has a multiplicative inverse, the system of quaternions forms a non-commutative division ring.

A quaternion $\mathbf{q} = (a, \mathbf{0})$ whose vector part is zero is called a *real quaternion*. Similarly, $\mathbf{q} = (0, \boldsymbol{w})$ whose real part is zero is called a *pure quaternion* (or *vector quaternion*). Since $\boldsymbol{w}$ is a three-dimensional vector, clearly there is a one-to-one correspondence between vectors in three-dimensional space and the quaternion sub-space consisting of pure quaternions (Fig. 1).

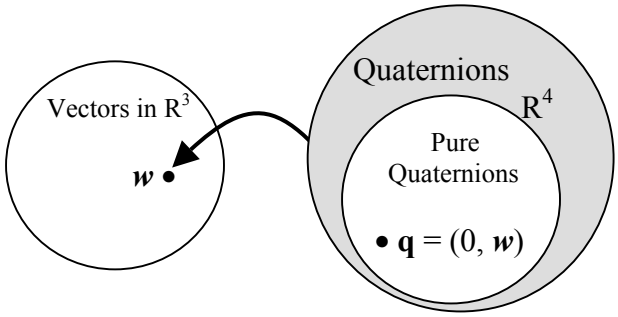

Fig. 1 A one-to-one correspondence between pure quaternions and vectors in R^3.

If $\mathbf{q}$ is a unit quaternion, we can write $\mathbf{q}$ in the form $(\cos\theta, \sin\theta\ \mathbf{v})$ where $|\mathbf{v}|=1$. The logarithm of $\mathbf{q}$ is a pure quaternion, defined as

$$\log(\mathbf{q}) = (0, \theta\ \mathbf{v}). \tag{10}$$

Conversely, if we have a pure quaternion of the form $\mathbf{q} = (0, \theta\ \mathbf{v})$ with $|\mathbf{v}|=1$, then the applying the exponentiation function we get

$$\exp(\mathbf{q}) = (\cos\theta, \sin\theta\ \mathbf{v}) \tag{11}$$

Based on the above two equations, we can also define

$$\mathbf{q}^t = \exp(t \log(\mathbf{q})) \tag{12}$$

3 Quaternion Applications in Mechanics

Perhaps the most important property of quaternions is that they can characterize rotations in a three dimensional space. The conventional way of representing three-dimensional rotations is by using a set of Euler angles $\{\psi, \phi, \theta\}$ which denote rotations about independent coordinate axes. Any general rotation can then be obtained as the result of a sequence of rotations, as given below.

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\varphi & 0 & \sin\varphi \\ 0 & 1 & 0 \\ -\sin\varphi & 0 & \cos\varphi \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi \\ 0 & \sin\psi & \cos\psi \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{13}$$

According to the well-known Euler's theorem in classical mechanics, the most general rotation of a rigid body with a fixed point can be achieved by a single rotation about an axis through the fixed

point. If $\boldsymbol{v} = (l, m, n)$ denote a vector in three-dimensional space, then a rotational transformation by an angle θ about this vector is given by

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} l^2(1-\cos\theta)+\cos\theta & lm(1-\cos\theta)-n\sin\theta & nl(1-\cos\theta)+m\sin\theta \\ lm(1-\cos\theta)+n\sin\theta & m^2(1-\cos\theta)+\cos\theta & mn(1-\cos\theta)-l\sin\theta \\ nl(1-\cos\theta)-m\sin\theta & mn(1-\cos\theta)+l\sin\theta & n^2(1-\cos\theta)+\cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{14}$$

Clearly, a single rotation about an equivalent axis defines the shortest path between two object orientations. The above transformation can be conveniently represented by a quaternion product:

$$\mathbf{w} = \mathbf{q}\,\mathbf{v}\,\mathbf{q}^* \tag{15}$$

where $\mathbf{w} = (0, x', y', z')$, $\mathbf{v} = (0, x, y, z)$, and $\mathbf{q} = \left(\cos\frac{\theta}{2},\ l\sin\frac{\theta}{2},\ m\sin\frac{\theta}{2},\ n\sin\frac{\theta}{2}\right)$.

Obviously, a rotation does not affect the length of a vector, and hence $\|\mathbf{w}\| = \|\mathbf{v}\|$. Thus for a proper rotation $\mathbf{q}$ in (15) must be a unit vector. Also note that when $\mathbf{q}$ is a unit quaternion, we have the property $\mathbf{q}^{-1} = \mathbf{q}^*$, from (9). We can thus define

$$\mathrm{L}_q(\mathbf{v}) = \mathbf{q}\,\mathbf{v}\,\mathbf{q}^* \tag{16}$$

as a quaternion rotation operator acting on any vector $\mathbf{v}$ in the three-dimensional space. This operator plays an important role in classical mechanics as well as in computer graphics. It can be easily shown that the operator is linear, so that

$$\mathrm{L}_q(k\mathbf{a}+\mathbf{b}) = k\,\mathrm{L}_q(\mathbf{a}) + \mathrm{L}_q(\mathbf{b}), \tag{17}$$

for any constant k, and vector quaternions $\mathbf{a}$ and $\mathbf{b}$.

Associated with large angle rotations are the notions of angular velocity, angular momentum, rotational frame transformations, kinematics and inverse kinematics relations. Inverse kinematics relations involve singularities when rotations are defined in terms of Euler angles. The singularity occurs when one of the Euler angles is 90 or 270 degrees. Similar singularities are encountered when trying to extract Euler angles from a given matrix (Direction Cosine Matrix) corresponding to an arbitrary rotation in space [6,7]. Quaternion based kinematics equations are numerically stable and free of singularities. Quaternions are therefore extensively used in robotics [8], strapdown inertial navigation systems, and spacecraft large angle maneuvers [9].

4 Quaternion Applications in Computer Vision

Estimating three-dimensional object position and orientation parameters from image data is an important aspect of many computer vision problems. Such problems are generally solved by extracting certain shape features from the transformed image, and by relating them to the corresponding features of the standard image using the unknown transformation parameters (Fig. 2).

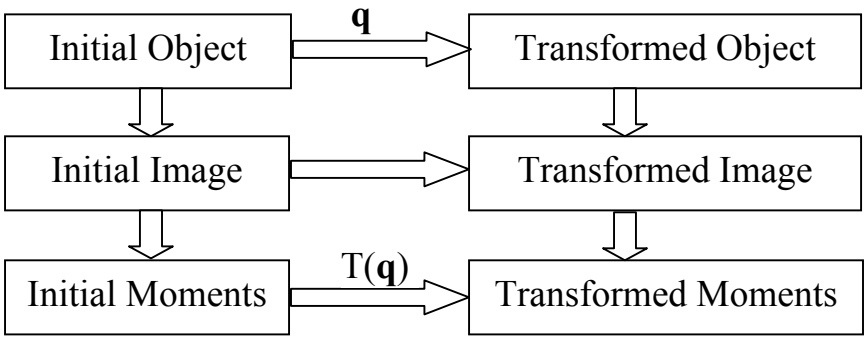

Fig. 2 Schematic of a quaternion based pose estimation problem.

There are primarily two advantages with a quaternion representation of the rotational parameters: (i) Since we get an algebraic system of equations in the unknown pose parameters, it is easier to look for closed-form solutions for the quaternion elements. Closed-form solutions are sought after in many vision applications where turn-around time and accuracy are of great concern. (ii) Even in situations where a large feature vector is used (where it is required to estimate 3D rotational parameters from a general object configuration involving occlusions), we still get a non-linear system of algebraic equations that can be solved using well-known numerical techniques.

Moment functions are used for characterizing various geometrical features of shape in pattern recognition applications. They can also be used in pose estimation problems to determine the relative attitude parameters, as described above. The mathematical framework for deriving closed form solutions of moment-quaternion equations is given in [10]. Autonomous vision based guidance and control algorithms in robotics as well as spacecraft rendezvous and docking require fast and accurate mechanisms for estimating 3D orientation from images. Quaternion based solutions for such problems can be found in [11-14].

5 Quaternion Applications in Graphics and Animation

We have earlier discussed the utility of the quaternion rotation operator (refer to eqs.(15),(16),(17)). Let $\mathbf{q}_1$, $\mathbf{q}_2$ be two unit quaternions representing arbitrary rotations in three-dimensional space. The composition of the first rotation followed by the second can be expressed in quaternion notation as

$$\begin{aligned} L_{q2}(L_{q1}(\mathbf{v})) &= \mathbf{q}_2 (\mathbf{q}_1 \mathbf{v} \mathbf{q}_1^*) \mathbf{q}_2^* \\ &= (\mathbf{q}_2\mathbf{q}_1) \mathbf{v} (\mathbf{q}_2\mathbf{q}_1)^* = L_{q2q1}(\mathbf{v}) \end{aligned} \tag{18}$$

Thus the quaternion product $\mathbf{q}_2\mathbf{q}_1$ defines a quaternion rotation operator L_{q2q1} which represents a sequence of operators L_{q1} followed by L_{q2}. This property can be generalized to the composition of any number of rotations.

The dual operation of a rotation of a point within a fixed frame, is the rotation of the frame where the point is inertially fixed. The coordinate transformation of the fixed point with respect to the rotating frame is given by [3]

$$\mathbf{v}' = \mathbf{q}^* \mathbf{v} \mathbf{q} = L_q^{-1}(\mathbf{v}) = L_{q^*}(\mathbf{v}). \tag{19}$$

The representation of rotations by quaternions has several advantages over the representation by Euler angles. The parameterization of rotations using quaternions involve only the angle and the axis (vector) of rotation, while Euler angles define a rotation as a composition of three independent rotations about coordinate axes. Further, the mutually independent characteristic of Euler angle rotations breaks down when the second Euler angle becomes 90 degrees, resulting in a loss of one degree of freedom. This phenomenon is called the gimbal lock [15]. Even different formulations of the Euler angles in the rotation matrix (13) does not remove this singularity. The condition of the gimbal lock could lead to frustrating results in a graphics animation.

The parameterization of orientation using Euler angles also requires more number of arithmetic operations, compared to a quaternion based approach. Table.1 summarizes a few important points.

	Quaternions	**Euler Angles**
Representation of Rotations	4 Elements	9 Elements (16 Elements if homogeneous coordinates are used)
Composition of Rotations	16 Multiplications and 12 Additions	27 Multiplications and 18 Additions

Table 1. Computational Complexity: Quaternions vs. Euler Angles

Several computer graphics algorithms perform incremental small-angle rotations. In order to speed-up the process, the cosine and sine terms in the direction cosine matrix (13) are often approximated as follows:

$$\begin{aligned} &\text{Cos } \psi \approx 1, \quad \text{when } \psi \text{ is small.} \\ &\text{sin } \psi \approx \psi, \quad \text{when } \psi \text{ is small.} \end{aligned} \tag{20}$$

When a sequence of such rotations are performed, the accumulated errors in a direction cosine matrix M, may cause the matrix to become non-orthogonal, and hence the transformation may not represent a pure rotation. The normalization of a matrix M is done by replacing it by the matrix $M(M^{\mathrm{T}}M)^{-½}$. This operation is computationally more expensive than the corresponding operation in the quaternion domain, which involves only a division by $\|\mathbf{q}\|$. Quaternions are also used in virtual reality systems to describe rotations [18].

In [4], Shoemake proposed a new method using quaternions, for interpolating two key frames in an animation sequence. Given two orientations in the 3-space, graphics animators conventionally used linear interpolation between the corresponding Euler angles to obtain the in-between key frames. However, such an interpolation algorithm has many limitations: (i) the interpolation does not give a natural transition between the key frames, since Euler angle parameterization combines the interpolated rotations along the coordinate axes, to form the resulting orientation, (ii) the extraction of Euler angles from the transformation matrix can cause singularities, and (iii) problems such as gimbal lock may be encountered, which would severely affect the smoothness of the animation. On the other hand, an interpolation between the quaternions of the two key frames generates a more realistic animation. We have already seen that a quaternion rotation operator is defined with respect to unit quaternions (16). The set of all unit quaternions form a unit sphere within the quaternion group. By representing the quaternions of two key frames as points on the unit sphere, a *spherical linear interpolation* (SLERP) defines the intermediate sequence of rotations as a path along the great circle between the two points on the sphere (Fig.3). SLERP and the associated quaternion operations have become increasingly popular among animators and games programmers in the recent past [16,17].

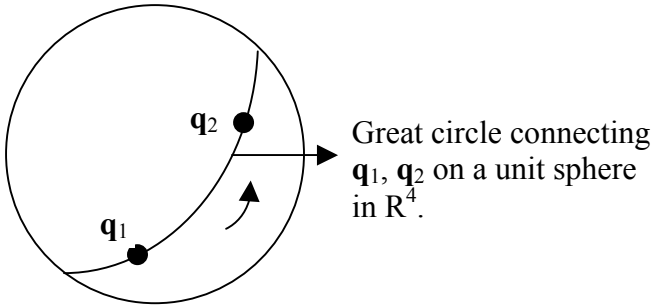

Fig. 3 Spherical linear interpolation between two unit quaternions

The spherical linear interpolation between $\mathbf{q}_1$ and $\mathbf{q}_2$ is given by

$$\text{Slerp}(\mathbf{q}_1, \mathbf{q}_2; u) = \mathbf{q}_1 (\mathbf{q}_1^{-1} \mathbf{q}_2)^u, \qquad 0 \le u \le 1. \tag{21}$$

The above formula, with exponentiation as defined in (12), requires a significant number of quaternion multiplications. The following equation is therefore generally preferred over (21).

$$\text{Slerp}(\mathbf{q}_1, \mathbf{q}_2; u) = \mathbf{q}_1 \frac{\sin(1-u)\theta}{\sin\theta} + \mathbf{q}_2 \frac{\sin u\theta}{\sin\theta}, \qquad 0 \le u \le 1. \tag{22}$$

Extensions of the above algorithm based on spherical Bezier curves for in-betweening a sequence of (more than two) key frames can be found in [4,5].

Modeling fractal geometries using quaternions is another well researched topic in comptuer graphics [19,20]. In the two dimensional complex domain, a Julia set is produced by an iterative algorithm $z_{n+1} \leftarrow z_n^2 + c$, for a given complex number c. The Julia set contains the initial complex values z_0 for which the system converges. Extending this idea to the vector space of quaternions, the set of pure quaternions $\mathbf{q}_0$ for which the system $\mathbf{q}_{n+1} \leftarrow \mathbf{q}_{n+1}^2 + \mathbf{c}$ converges for a given pure

quaternion **c**, is a three-dimensional fractal in R^3. The problem of generating a 4D Mandelbrot set is addressed in [20].

6 Conclusions

This paper has attempted to provide a very broad overview of the various applications of quaternions, particularly in the context of its increasing importance in the fields of computer graphics, animation and virtual reality. The fundamental properties of quaternions have been outlined, and the mathematical preliminaries for using quaternions as a rotation operator have been presented. The paper also described some of the applications in computer vision and robot vision where quaternions could be effectively used in extracting three dimensional orientation/relative attitude information. The paper is intended to convey the importance of quaternions to graphics and games programmers, and also to stimulate further research towards extending/improving current applications.

References

1. Sir William R. Hamilton, Elements of Quaternions (Third Edition), Chelsea Publishing Co., New York (1963).

2. G. Birkhoff and S. MacLane, A Survey of Modern Algebra, A.K.Peters (1997).

3. Jack B. Kuipers, Quaternions and Rotation Sequences, Princeton University Press (1999).

4. Ken Shoemake, Animating Rotation with Quaternion Curves, SIGGRAPH 85, Proc. Computer Graphics Vol 19 No. 3, (1985), pp. 245-254.

5. S.G. Hoggar, Mathematics for Computer Graphics, Cambridge University Press (1992).

6. Funda J, On Homogeneous Transforms, Quaternions and Computational Efficiency, IEEE Trans. on Robotics and Automation, Vol. 6, No. 3 (1990), pp. 383-388.

7. Klumpp A.R, Singularity Free Extraction of a Quaternion From a Direction Cosine Matrix, Journal of Spacecraft and Rockets, Vol. 16, No.1 (1979), pp.1-9.

8. Yuan J.S.C, Closed Loop Manipulator Control With Quaternion Feedback, IEEE Trans. on Robotics and Automation, Vol. 4, No. 4 (1988), pp. 434-439.

9. Wie B and Barba P.M, Quaternion Feedback for Spacecraft Large Angle Maneuvers, Journal of Guidance Control and Dynamics, Vol. 8, No. 3 (1985), pp 360- 365.

10. R. Mukundan, Estimation of Quaternion Parameters Two-Dimensional Image Moments, Graphical Models and Image Processing, Vol. 54, No. 4 (1992), pp. 345-350.

11. R. Mukundan and N.K. Malik, Attitude Estimation Using Moment Invariants, Pattern Recognition Letters, Vol 14 (1993), 199-205.

12. R. Mukundan and K.R. Ramakrishnan, A Quaternion Based Solution to the Pose Determination Problem for Rendezvous and Docking Simulations, Mathematics and Computers in Simulation, Vol. 39 (1995), 143-153.

13. Horn B.K.P, Closed Form Solution of Absolute Orientation Using Unit Quaternions, Journal of the Optical Society of America, Vol. 4, No. 4 (1987), pp 629- 642.

14. Pervin E and Webb J.A, Quaternions in Computer Vision and Robotics, Intnl. Conf. on Computer Vision and Pattern Recognition, (1983), 382-383.

15. Alan Watt and Mark Watt, Advanced Animation and Rendering Techniques, Addison Wesley (1992).

16. Dante Treglia and Mark Deloura, Games Programming Gems 3, Charles River Media (2002).

17. Eric Lengyel, Mathematics for 3D Programming and Computer Graphics, Charles River Media (2001).

18. John Vince, Virtual Reality Systems, Addison-Wesley (1999).

19. C.A.Pickover, Chaos and Fractals, Elsevier (1998).

20. Yan Ke and E.S. Panduranga, A Journey into the Fourth Dimension, Proc. IEEE Conf on Visualization (1990), pp. 219-229.