Rotational Joint Limits in Quaternion Space

Gino van den Bergen
Dtecta

Rotational Joint Limits: 1 DoF

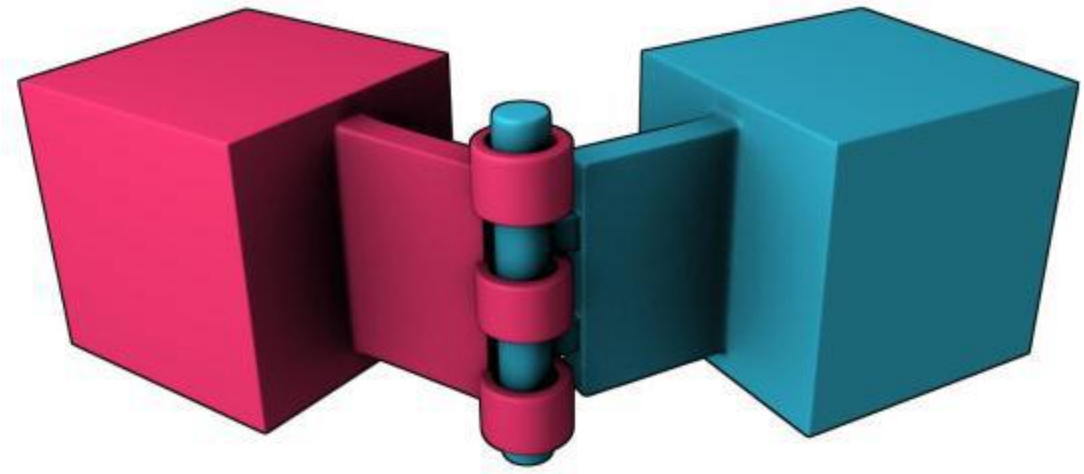


Image: Autodesk, Creative Commons

Rotational Joint Limits: 3 DoFs

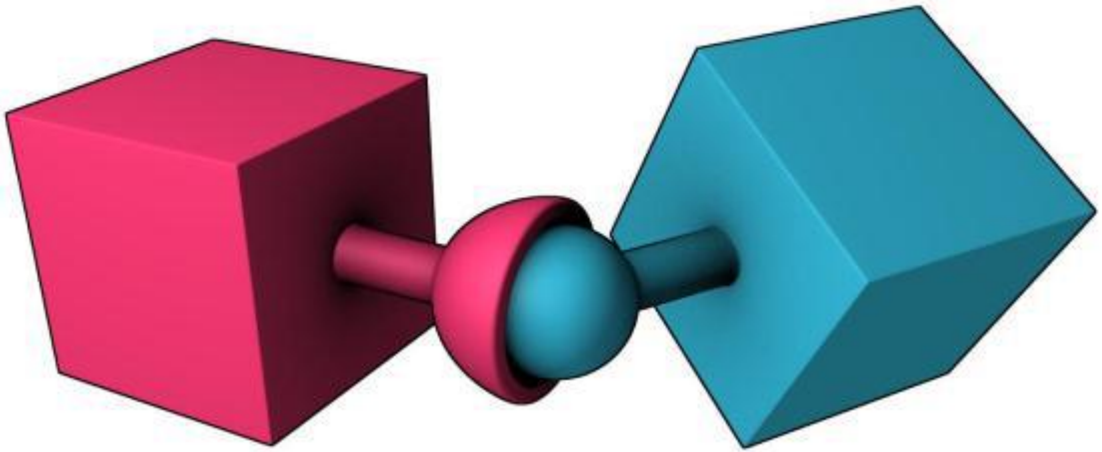
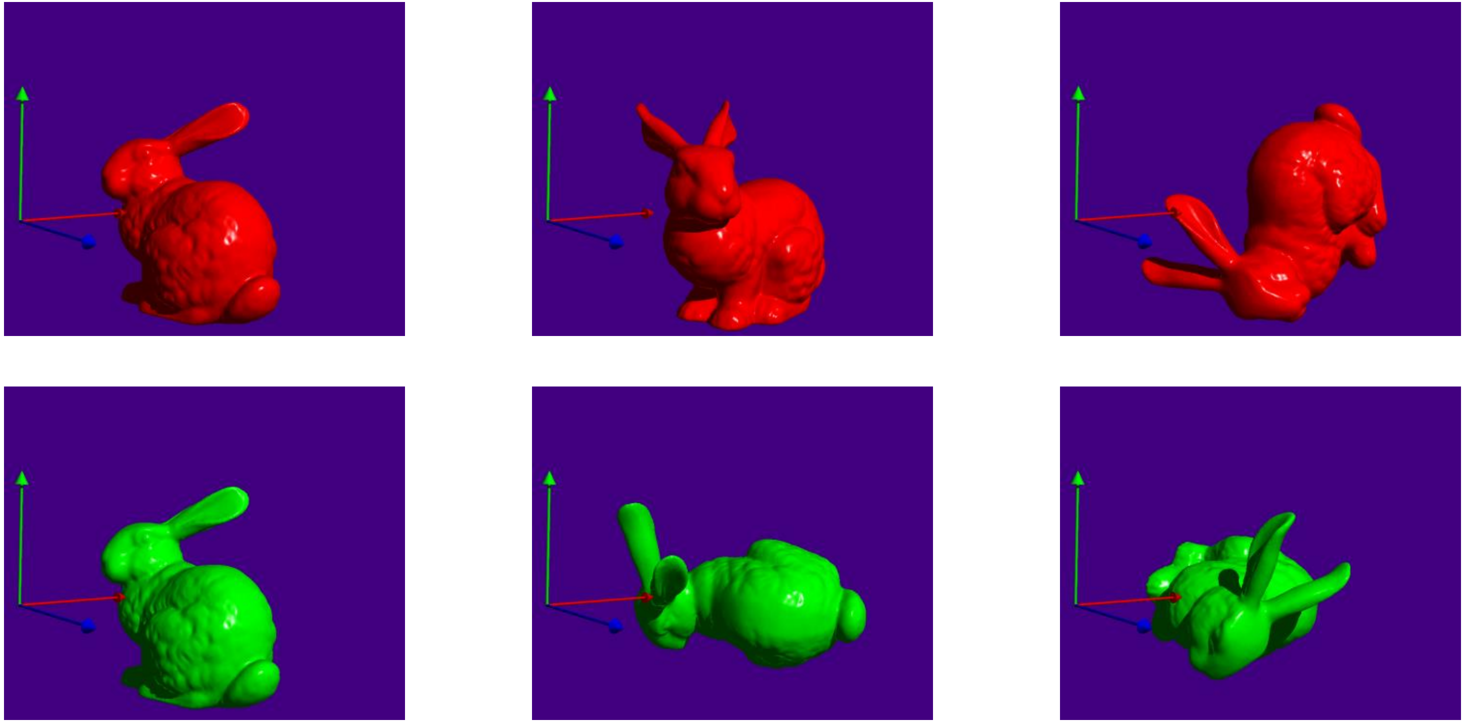


Image: Autodesk, Creative Commons

Parameterize 3-DoF Rotations

- Ideally, we want a 3D parameter space that is free of singularities, …
- … in which range bounds for shoulder and hip joints can be intuitively expressed.
- If only 3D rotations would play so nicely…

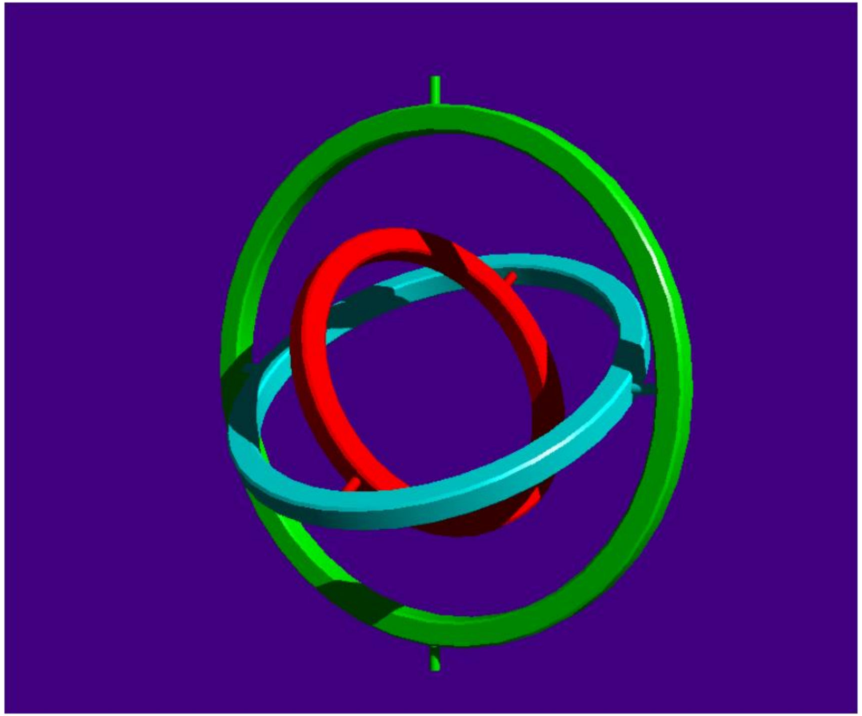
3D Rotations Do Not Commute



Euler Angles

- Parameterize 3D rotation by angles of rotation about three predefined axes.
- Choice of axes is arbitrary, as long as no two consecutive axes are the same (for example, *XYZ*, *ZYX*, *XZX*, *YXY*, …)
- Limit each angle independently.

Euler Angles (Cont'd)



Euler Angles (Cont'd)

- Euler angles space has singularities due to collapsing axes (aka *gimbal lock*).
- Only suitable for joints formed by nested gimbals.
- Totally unsuitable for shoulder and hip joints.

Euler’s Rotation Theorem

*“Any orientation of a 3D object can be reached from an initial orientation by executing a **single** rotation about a suitable 3D axis.”*

(Leonard Euler, 1775)

Axis-Angle Parameterization

- Axis is represented by a normalized 3D vector (3 parameters, 2 DoFs!).
- Zero-angle rotations form a singularity (axis is arbitrary).
- Hard to express joint limits.

Exponential Map Parameterization

- Scale axis by angle to form a 3D vector with three independent parameters.
- Zero-angle rotation is represented uniquely by the zero vector.
- Still has singularities for angles that are multiples of 2π (360˚).

Exponential Map (Cont'd)

- Limit angle range to $[0, 2\pi)$. This clears out all singularities.
- We still have a double covering. Rotating with angle θ about axis $\mathbf{u}$ results in the same orientation as rotating with angle $2\pi - \theta$ about axis $-\mathbf{u}$.

Exponential Map (Cont'd)

- Limiting the angle range to $[0, \pi]$ restricts the double covering to angles of π.
- Parameterization space is a 3D ball with radius π.
- Admissible orientations form a volume inside the 3D ball.

Quaternions

- Quaternions extend complex numbers

  $$\mathbf{q} = w + x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$$

  where w, x, y and z are real numbers
- w is the real or *scalar* part, and
- (x, y, z) is the imaginary or *vector* part.

Quaternions (cont'd)

- Quaternions behave as 4D vectors w.r.t. addition and scaling.
- In multiplications, the imaginary units resolve as: $\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1$
- In scalar-vector notation, multiplication is given by: $[w_1, \mathbf{v}_1][w_2, \mathbf{v}_2] = [w_1 w_2 - \mathbf{v}_1 \bullet \mathbf{v}_2, w_1\mathbf{v}_2 + w_2\mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2]$

Unit Quaternions

- Unit quaternions (points on sphere in 4D) form a multiplicative subgroup.
- A rotation with angle θ about unit vector $\mathbf{u}$ is represented by unit quaternion

$$\left[\cos\left(\frac{\theta}{2}\right),\ \sin\left(\frac{\theta}{2}\right)\mathbf{u}\right]$$

Unit Quaternion Parameterization

- Unit quaternions are parameterized by four dependent parameters (3 DoFs).
- 3D orientations are doubly covered: **q** and -**q** represent the same orientation.
- Yet, for procedural animation quaternions are generally the best choice.

Who Needs the Scalar Part?

- The scalar part *w*, less a sign, can be found given the vector part **v**, since

$$w = \pm\sqrt{1 - \mathbf{v} \cdot \mathbf{v}}$$

- Any orientation can be represented by a unit quaternion with nonnegative *w*.

Quaternion Vectors

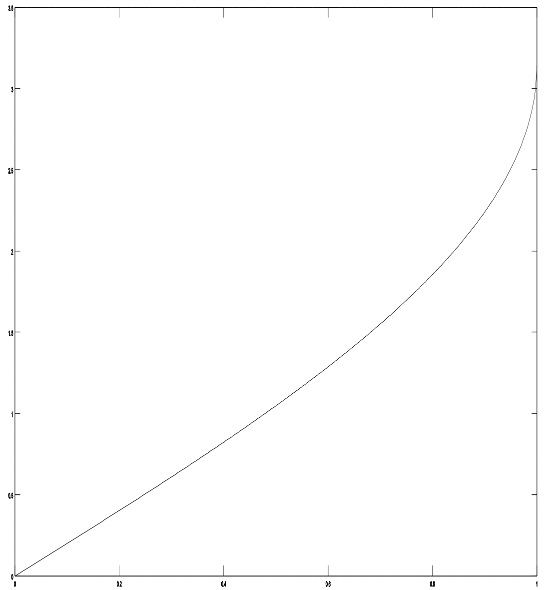
- Quaternion vectors parameterize orientations using three independent parameters.
- Zero vector is zero-angle rotation.
- Parameterization space is a unit ball.
- Only rotations over π (unit vectors) are doubly covered.

Quaternion Vectors (cont’d)

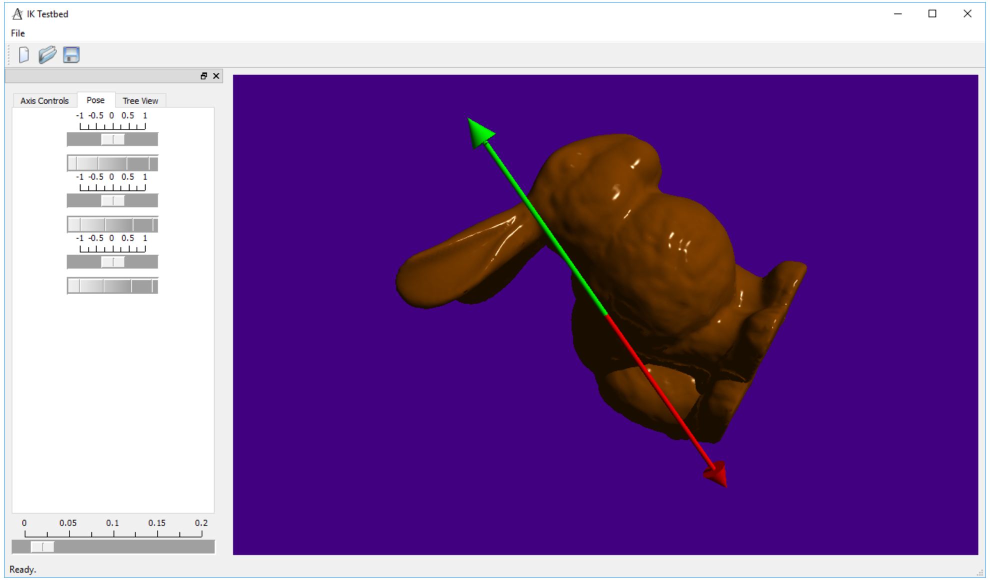
- Quaternion vectors map one-to-one to exponential map vectors.
- For a quaternion vector **v**, the corresponding exponential-map vector is

$$\frac{2\arcsin(\|\mathbf{v}\|)}{\|\mathbf{v}\|}\mathbf{v}$$

Quaternions to Exponential Map



Quaternion Vectors Demo



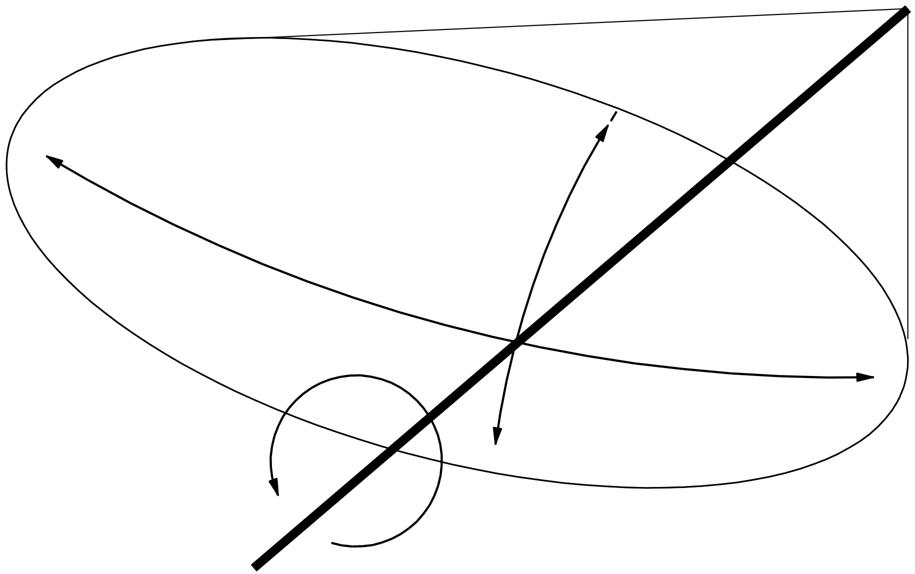
Swing-Twist Decomposition

- Decompose rotation into a swing and twist component…

$$\mathbf{q} = \mathbf{q}_{\text{swing}}\,\mathbf{q}_{\text{twist}}$$

- …and, limit each component independently.

Swing-Twist Decomposition



Swing-Twist Decomposition

- For $\mathbf{q} = w + x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$ we find that

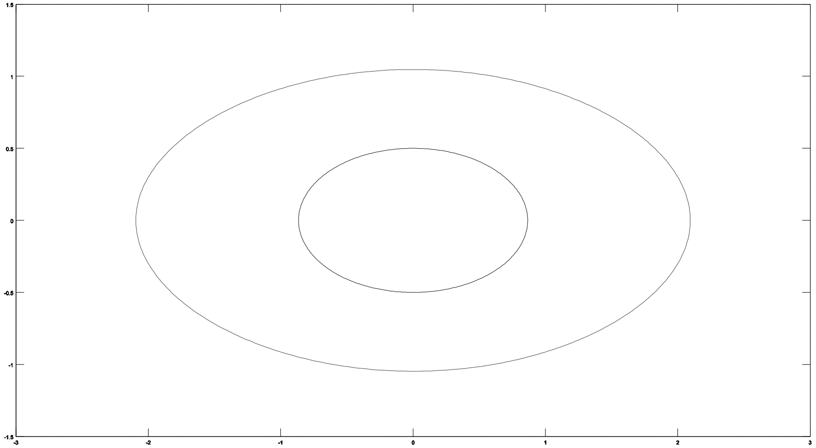
$$\mathbf{q}_{\text{swing}} = s + \frac{wy - xz}{s}\mathbf{j} + \frac{wz + xy}{s}\mathbf{k}$$

$$\mathbf{q}_{\text{twist}} = \frac{w}{s} + \frac{x}{s}\mathbf{i},$$

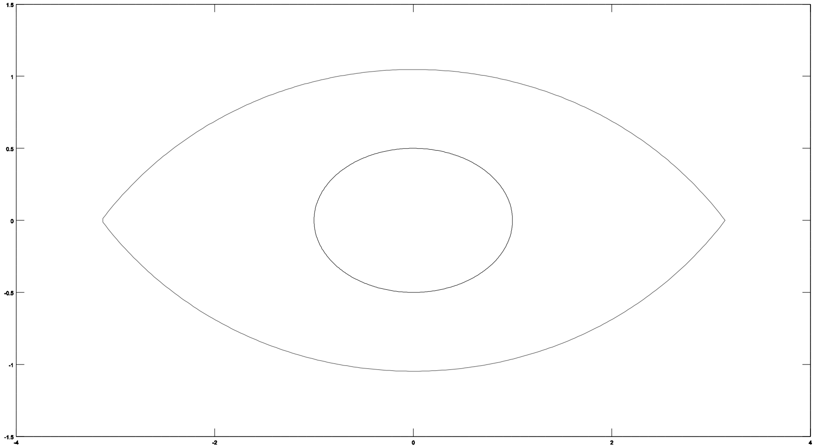
where $\quad s = \sqrt{w^2 + x^2}$

Clamp Swing to Elliptical Disk

- Quaternion Vector vs. Exponential Map



120°x 60°

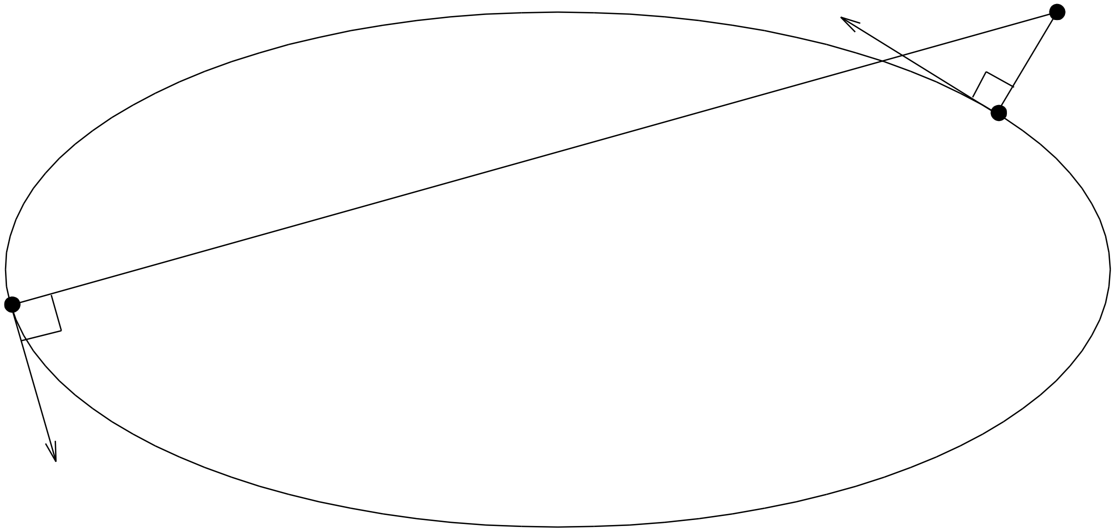


180°x 60°

Clamp Swing to Elliptical Disk

- Map 2D points (**j**,**k**) outside the ellipse to their closest point on the ellipse.
- The outside point lies on the line that is normal to the ellipse and passes through its closest point.

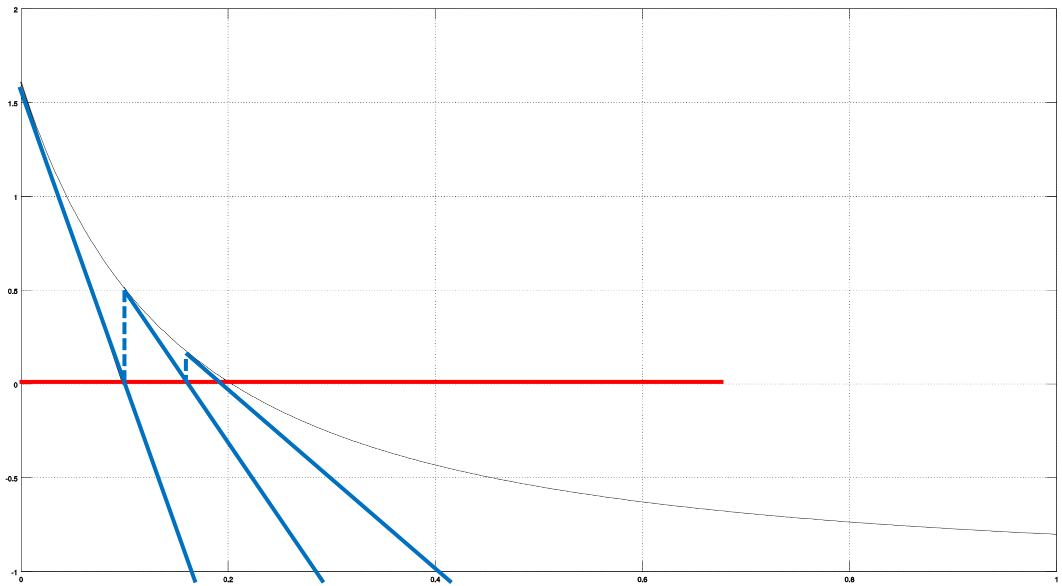
Clamp Swing to Elliptical Disk



Clamp Swing to Elliptical Disk

- A closed-form solution requires solving a quartic (4^{th} order) polynomial.
- Root finding using Newton-Raphson is more accurate, and potentially faster.
- Incremental error correction is better suited for smooth animation.

Newton-Raphson



Volume Limits

- Generalization of clamp to ellipsoid or elliptic cylinder is straightforward.
- Clamping to other convex shapes can be done using Gilbert-Johnson-Keerthi (GJK).

References

- Gino van den Bergen. “Rotational Joint Limits in Quaternion Space”. Game Engine Gems 3, Eric Lengyel (Ed.) (April 2016)
- Jim Van Verth. “Math for Game Programmers: Understanding Quaternions” GDC 2013.
- F. Sebastian Grassia. “Practical parameterization of rotations using the exponential map”. *Journal of Graphics Tools*, Vol. 3, No. 3 (March 1998), pp. 29–48.
- E. G. Gilbert, D. W. Johnson, and S. S. Keerthi. “A Fast Procedure for Computing the Distance Between Complex Objects in Three-Dimensional Space”. *IEEE Journal of Robotics and Automation*, Vol. 4, No. 2 (April 1988), pp. 193–203.

Thank You!

Check me out on

- Web: www.dtecta.com
- Twitter: @dtecta
- Sample code available in MoTo: https://github.com/dtecta/motion-toolkit