# Animating and sweeping polyhedra along interpolating screw motions

**Jarek R. Rossignac**
GVU Center and College of Computing
Georgia Institute of Technology
801 Atlantic Drive, NW, Room 241
Atlanta, GA 30332-0280
Tel: +1-404-894-0671, Fax: +1-404-894-0673
jarek@cc.gatech.edu

**Jay J. Kim**
School of Mechanical Engineering
Hanyang University
Seoul, Korea
Tel: +822-2290-0447, Fax: +822-2298-4634
jaykim@email.hanyang.ac.kr

**November 4, 1999**

**Abstract**

*CAD and animation systems offer a variety of techniques for designing and animating arbitrary movements of rigid bodies. Such tools are essential for planning, analyzing, and demonstrating assembly and disassembly procedures of manufactured products. In this paper, we advocate the use of screw motions for such applications, because of their simplicity, flexibility, uniqueness, and computational advantages. Two arbitrary poses of an object are interpolated by a screw motion, which, in general, is unique and combines a minimum-angle rotation around an axis A with a translation by a vector parallel to A. We discuss the advantages of screw motions for the intuitive design and local refinement of complex motions. We present a simple and efficient algorithm for computing the parameters of a screw motion that interpolates any two or more poses and explain how to use it to produce animations of the moving objects. Finally, we introduce a geometric construction and propose a simple procedure for computing a set of faces which may be used to display the 3D region swept by a polyhedron that moves along a screw motion.*

***1.*** **Keywords**: Animation, Assembly, Screw motion, Minimal rotation, Swept region, Polyhedron

## INTRODUCTION

CAD systems provide a rich variety of techniques for designing surfaces or solid models and for combining them into large assemblies for digital mock-up or entertainment applications. The associated assembly and disassembly processes may involve complicated rigid body motions of individual components. Manufacturing plans may involve representations of the cutter motions. Manufacturing automation modules may carry representation of the hierarchical motions of articulated robots. Artistic applications[4] may call for physically correct or emotionally charged motions. In all of these applications, motions are often highly constrained and formulated in terms of abstract concepts making manual editing unnecessary or impractical.

In this paper, we focus on a different breed of applications, where engineers or animators must quickly produce series of simple motions that interpolate two or more poses of a moving subset of a model. Examples may be found in the creation of animations that explode CAD assembly and in the quick design of motions that show how parts must be handled and assembled for online maintenance manuals.

Throughout this paper, we assume that the moving shape is a rigid body called the *object*. We also assume that it is subject to a continuous movement called the *motion*. We use the term *pose* to refer to the position and orientation of the object at any given time during the motion. The terms *initial pose* and *final pose* refer respectively to the poses at the beginning and the end of the motion.

A rigid body motion is a continuous mapping from the time domain to a set of poses. To relieve the designers from the burden of specifying this mapping in abstract mathematical terms, combinations of simple rigid-body motion-primitives, such as linear translations or rotations around the principal axes, are often used. These simple motions are planar and thus ill-suited, by themselves, for approximating arbitrary motions in 3D-space. Specifying combinations of them that should occur in parallel to produce a 3D motion is difficult and error prone, because each rotation and translation is expressed in a different coordinate system, that results from the cumulative effect of the previous motion primitives in the sequence.

Furthermore, the results of these compositions of simple motions may be affected by the choice of the coordinate system, which further complicates the designer's job[17]. More detail explanation about motion design may be found in the literature[11]. *Figure 1a* shows the superposition of several sample instances of a simple object moving along a screw motion that interpolates the initial to final poses. The trajectory of the object is not surprising, given the two constraint poses. For the same initial and final poses, a very different and rather surprising trajectory, *Figure 1b*, is produced when using a motion generated by linearly interpolating the Euler angle parameters used to describe the two poses.

To eliminate these drawbacks, we advocate the use of *key-frame* animations[14], where motions of objects are specified by their initial and final poses, and possibly by an ordered series of intermediate poses that should be interpolated during the motion. Key-frame animations are commonly used in animation and robotics, but the motions they define are usually either dependent on the coordinate system, on the process through which the key poses were created, or on both.

(a) Screw motion (b) Linear interpolation

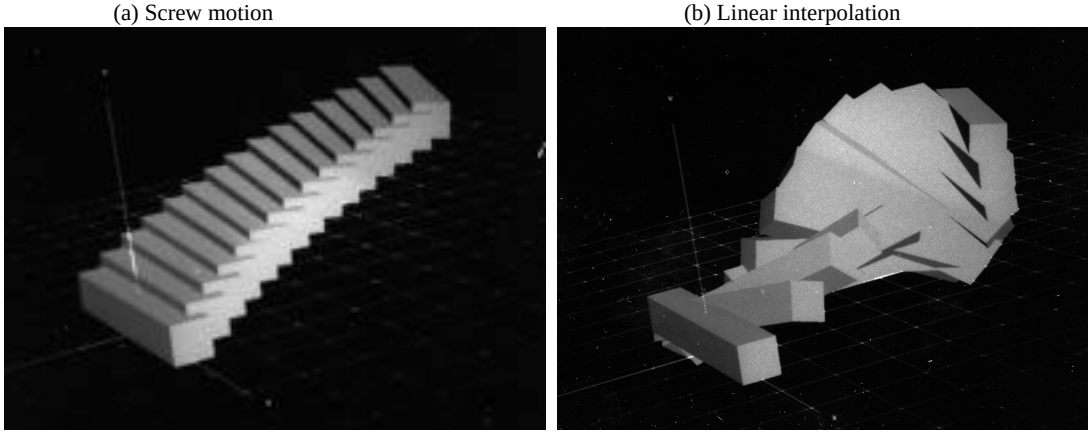

***Figure 1:*** *Screw motion (left) versus linear interpolation of pose parameters (right)*

Consider the two initial and final key-poses, ***A*** and ***B***, of *Figure 2a* and *2b*, that define the relative poses of an object (large triangle) with respect to a camera (small triangle). For clarity, the object-camera combinations are shown so that the object keeps its position and orientation throughout the illustrations of *Figure 2*. If the interpolating motion is computed as the minimum angle rotation

synchronized with a straight line distance translation, the motion will be different when computed in the coordinate system of the camera (*Figure 2c*) and when computed in the coordinate system of the object (*Figure 2d*). However, screw motion, which in the 2D case degenerates into a pure rotation or a pure translation will produce consistently the same relative trajectory (*Figure 2c*). Thus, when screw motions are used, the object always appears to rotate around its center, regardless of which coordinate system was used and of the position of the origin of the coordinate system.

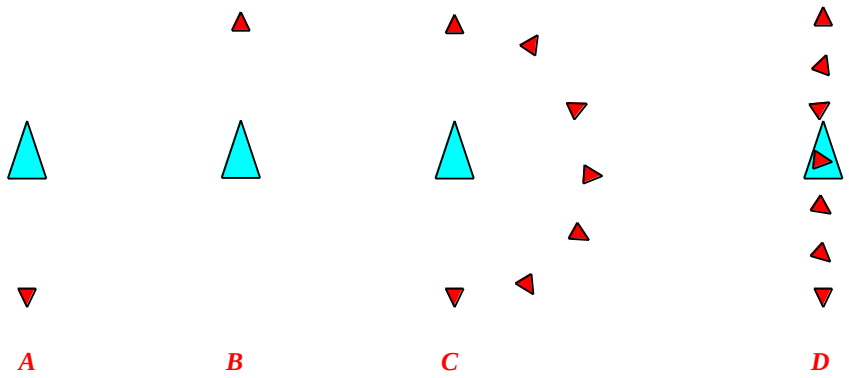


***Figure 2*** *Two different motions (**C** and **D**) defined by the same relative poses, (**A** and **B**), but expressed in different coordinate systems.*

With the approach proposed in this paper, motion designers may use direct manipulation techniques to specify the starting and ending poses, or even a sequence of key-poses to be interpolated, and need not be concerned with coordinate systems or with complex motion sequences and parameters. To be intuitive, our key-frame animation technique must satisfy two constraints:

(1) the motion that interpolates two consecutive key-poses must be intuitive and easily predicted by the designer from the two poses

(2) the designer must be able to refine a previously designed motion by changing the initial or the final key-poses, or by inserting intermediate poses for local control.

The first constraint would be clearly violated, if the interpolating motion were affected by the coordinate system, because, for given initial and final poses, the motion would depend on parameters that the user does not control, and may not even want to be aware of.

The second constraint requires that the designer be able to use, as a new key-pose, a pose that is adopted during the original motion. Clearly, the mere insertion of this new key-pose should not change the original motion, otherwise, the designer would not be able to insert such intermediate key-poses to adjust the motion during fine-tuning.

This flexibility may be illustrated by the following description of the *Meditor* (motion-editor) design interface developed by the authors. In Meditor, the designer specifies the initial and final poses, $\boldsymbol{A}$ and $\boldsymbol{B}$, of any collection of 3D objects through direct manipulation using a variety of input devices. The interpolating motion is a pose-valued expression, $\boldsymbol{P}(t,\boldsymbol{A},\boldsymbol{B})$, which is parameterized by $\boldsymbol{A}$, $\boldsymbol{B}$, and $t$. The time parameter $t$ varies between 0 and 1 and the interpolation properties of the motion imply that $\boldsymbol{P}(0,\boldsymbol{A},\boldsymbol{B})=\boldsymbol{A}$ and $\boldsymbol{P}(1,\boldsymbol{A},\boldsymbol{B})=\boldsymbol{B}$.

By simply turning a dial that represents the time, $t$, the designer changes the value of $t$ and thus animates the object along its motion. At a selected value $t'$ of $t$, the designer may create a new key-pose, $\boldsymbol{M}$, defined as $\boldsymbol{P}(t',\boldsymbol{A},\boldsymbol{B})$. The original single motion $\boldsymbol{P}(t,\boldsymbol{A},\boldsymbol{B})$ is now replaced by two motions, $\boldsymbol{P}(u,\boldsymbol{A},\boldsymbol{M})$ with $u$ in $[0, t']$ and $\boldsymbol{P}(v,\boldsymbol{M},\boldsymbol{B})$ with $v$ in $[t', 1]$. The concatenation of $\boldsymbol{P}(u,\boldsymbol{A},\boldsymbol{M})$ and $\boldsymbol{P}(v,\boldsymbol{M},\boldsymbol{B})$ produces in Meditor a motion that is identical to the original motion $\boldsymbol{P}(t,\boldsymbol{A},\boldsymbol{B})$, if we use the following mapping: for $t$ in $[0, t']$, $t$ is $u$; and for $t$ in $[t',1]$, $t$ is $v-t'$. This key-pose insertion has replaced a single motion from $\boldsymbol{A}$ to $\boldsymbol{B}$, denoted by $\boldsymbol{A}\rightarrow\boldsymbol{B}$, by a composite motion from $\boldsymbol{A}$ to $\boldsymbol{M}$ and from $\boldsymbol{M}$ to $\boldsymbol{B}$, denoted $\boldsymbol{A}\rightarrow\boldsymbol{M}\rightarrow\boldsymbol{B}$. This change has not altered the total motion and the designer is free to adjust the result by editing $\boldsymbol{A}$, $\boldsymbol{M}$, *or* $\boldsymbol{B}$ very slightly, while playing with the time knob to verify the effect of these changes. Furthermore, slight perturbations of $\boldsymbol{M}$ through translations by a relatively small vector or through rotations by a small angle will in general change the overall motion only slightly. In loose terms, the motion $\boldsymbol{A}\rightarrow\boldsymbol{M}\rightarrow\boldsymbol{B}$ is a continuous function of $\boldsymbol{M}$, and also of $\boldsymbol{A}$ and $\boldsymbol{B}$.

The user may also choose to insert three new key-frames somewhere in the $\boldsymbol{A}\rightarrow\boldsymbol{B}$ motion and produce a composite motion $\boldsymbol{A}\rightarrow\boldsymbol{L}\rightarrow\boldsymbol{M}\rightarrow\boldsymbol{N}\rightarrow\boldsymbol{B}$. Altering $\boldsymbol{M}$ will only affect the resulting motions between the time parameters associated with $\boldsymbol{L}$ and $\boldsymbol{N}$. Thus the designer has local control over the motion.

Note that these two principal characteristics (that $\boldsymbol{A} \rightarrow \boldsymbol{P}(t',\boldsymbol{A},\boldsymbol{B}) \rightarrow \boldsymbol{B}$ be identical to $\boldsymbol{A} \rightarrow \boldsymbol{B}$ and that $\boldsymbol{P}(t,\boldsymbol{A},\boldsymbol{B})$ be a continuous function of $\boldsymbol{A}$ and $\boldsymbol{B}$) are properties of interpolating screw-motions, but are not true for a variety of interpolating motions discussed in the literature.

The concept of a screw motion was developed in the 18th-19th centuries and later extensively elaborated by Ball[3] for applications to the kinematics and dynamic analysis. Recently, the screw theory has been revisited in the context of spatial mechanism and robots [6,15]. A screw motion is a special combination of two simultaneous motions: a linear translations along a vector **s** and a rotation around a constant axis (*screw axis*) parallel to **s**. During a screw motion, the amount of translation and the amount of rotation are linear functions of *t*. The trajectory of any point on the moving object is a *helix* and the velocity vector of the point is constant with respect to the object's local (and moving) coordinate system.

A screw axis can be represented with a unit vector, **s**, and an arbitrary point, **p**, on the axis. A finite screw motion, $\boldsymbol{A} \rightarrow \boldsymbol{B}$, is specified by the four parameters: **s**, **p**, *d* and *b*, where *d* denotes the total translation distance along **s** and where *b* denotes the total rotation angle. The quantity *d*/*b* is called the *pitch* of the screw. We will use the notation $\boldsymbol{M}(\mathbf{s},\mathbf{p},d,b)$ to denote a screw motion represented by these four parameters. Note that pure translations and pure rotations are special cases of screw motions: $\boldsymbol{M}(\mathbf{s},\mathbf{p},d,0)$ and $\boldsymbol{M}(\mathbf{s},\mathbf{p},0,b)$ respectively.

In this paper, we always refer to the screw motion that has minimal rotation angle, given $\boldsymbol{A}$ and $\boldsymbol{B}$. With this assumption, the screw motion $\boldsymbol{P}(t,\boldsymbol{A},\boldsymbol{B})$ is uniquely defined by the starting and ending poses, $\boldsymbol{A}$ and $\boldsymbol{B}$, and is independent of the coordinate system. Thus, the screw motion that interpolates a series of key-poses is unique (except for the degenerate cases where two consecutive key-poses correspond to a total motion that is a 180 degree rotation).

Aesthetic concerns may call for a "smooth" motion that interpolates a series of poses[16]. Although it is relatively simple to compute interpolating motions that produce a geometrically smooth trajectory for a given point on the object, it is more difficult to ensure such trajectory smoothness for all points of the moving object. We say that a trajectory is smooth when it is continuous and has a continuous velocity. Given that the interpolating motion is defined in terms of poses without reference to the moving object, to be smooth regardless of the object being moved, a motion would have to produce smooth trajectories for all points of the three-dimensional space. This is not the case for most piecewise-simple motions. Consider for instance an airplane rolling at constant speed along a 2D path made of smoothly connected circular arcs. While sliding along a single arc, the plane (and the entire space attached to it) is subject to a smooth motion (all points of that space travel along smooth curves). When switching between one arc and the next one, the cockpit may still be subject to a smooth motion, but the ends of the wings may abruptly change direction, and thus do not follow a smooth trajectory. If we were to use a motion that is a continuous and twice differentiable function of the parameter *t*, the trajectory of each point would be $C^1$, and hence smooth. We can easily construct a $C^1$ motion from a piecewise smooth motion by having the motion come gradually to a stop at each junction between two consecutive motion elements. Points on the wings of our plane would come to a halt progressively before changing directions. Such stops would produce rather unnatural motions. Although other techniques exist for producing smooth interpolating motions, they do not satisfy constraints (1) and (2) listed above. Therefore, we have focused our studies on motions that are piecewise smooth, but do not guarantee smoothness between consecutive screw motions that interpolate a series of key-poses. We have found, however, that direct manipulation of the key-poses and the local control and refinement offered by our approach enable designers to easily construct visually smooth trajectories, when required.

Let $\mathrm{W}@\boldsymbol{P}(t,\boldsymbol{A},\boldsymbol{B})$ denote the instance of an object W at time *t*, when W is moving along a screw motion $\boldsymbol{P}(t,\boldsymbol{A},\boldsymbol{B})$ from its initial instance $\mathrm{W}@\boldsymbol{A}$ to its final instance $\mathrm{W}@\boldsymbol{B}$. A moving object is an extension of $R^3$ into the four-dimensional space. A slice of this hypersolid by a hyper plane of constant t corresponds to $\mathrm{W}@\boldsymbol{P}(t,\boldsymbol{A},\boldsymbol{B})$. Thus, a moving object, W, may be represented as a hyper-solid in 4D. The projection of this hyper-solid into a three-dimensional space orthogonal to the time-axis is the 3D region, $\mathrm{S}(\mathrm{W},\boldsymbol{A},\boldsymbol{B})$, swept by W. We can formulate $\mathrm{S}(\mathrm{W},\boldsymbol{A},\boldsymbol{B})$ as the infinite union of $\mathrm{W}@\boldsymbol{P}(t,\boldsymbol{A},\boldsymbol{B})$, for all values of *t* in [0,1]. Three-dimensional renderings of $\mathrm{S}(\mathrm{W},\boldsymbol{A},\boldsymbol{B})$ provide the designer with powerful tools for analyzing the trajectory of the object and for studying its interaction with other objects and with potential obstacles. We view the possibility of computing a 3D representation of $\mathrm{S}(\mathrm{W},\boldsymbol{A},\boldsymbol{B})$ that is suitable for interactive 3D inspection as a necessary functionality in any motion design and analysis software.

For general motions, it is expensive to display $\mathrm{S}(\mathrm{W},\boldsymbol{A},\boldsymbol{B})$[1,5,9] and many systems resort to a g raphic superposition of the instances of $\mathrm{W}@\boldsymbol{P}(t,\boldsymbol{A},\boldsymbol{B})$ for a set of samples of t. For a piecewise-screw motion, however, that cost may be considerably reduced by exploiting the following two observations.

First, we can display the envelope $\mathrm{E}(\mathrm{W},\boldsymbol{A},\boldsymbol{B})$, defined below, instead of the boundary $\mathrm{bS}(\mathrm{W},\boldsymbol{A},\boldsymbol{B})$ of $\mathrm{S}(\mathrm{W},\boldsymbol{A},\boldsymbol{B})$, because it contains that boundary, i.e., $\mathrm{bS}(\mathrm{W},\boldsymbol{A},\boldsymbol{B})$ $\mathrm{E}(\mathrm{W},\boldsymbol{A},\boldsymbol{B})$, and because the excess of $\mathrm{E}(\mathrm{W},\boldsymbol{A},\boldsymbol{B})$ that is not part of $\mathrm{bS}(\mathrm{W},\boldsymbol{A},\boldsymbol{B})$ lies inside the swept region, i.e., $\mathrm{E}(\mathrm{W},\boldsymbol{A},\boldsymbol{B})$ $\mathrm{S}(\mathrm{W},\boldsymbol{A},\boldsymbol{B})$.

Thus, displaying a shaded image of E(W,***A***,***B***) will produce the correct picture for all observer positions outside of S(W,***A***,***B***).

Second, we may define the envelope, E(W,***A***,***B***), as the sweep G(W, ***A***,***B***)@***P***(*t*,***A***,***B***) of a generator G(W,***A***,***B***), which is the union of fixed line segments, called the *silhouette edges*, defined on W by the screw **A→B**. G(W,***A***,***B***) is the locus of all points on W with velocity tangential to the surface of W. Because, during a screw motion, the velocity of a point is constant in the local coordinate frame of the moving object, G(W,***A***,***B***) may be pre-computed from a description of W, ***A***, and ***B*** and then swept along the screw motion. The face swept by each silhouette edge is a subset of a ruled surface and may be easily approximated by a series of quadrilateral or triangular graphic primitives to the desired level of accuracy. A more precise definition of silhouette edges and a simple procedure for computing them are developed later in this paper.

The rest of the paper is organized as follow. We first introduce a simple and efficient technique for computing the axis of the screw and the angle and displacement for ***P***(*t*,***A***,***B***), given the initial and final key-poses ***A*** and ***B***. Then, we the explain the computation of the silhouette edges of G(W,***A***,***B***). Finally, we show how to animate an object that moves along a screw motion and how to compute the faces of E(W,***A***,***B***).

## COMPUTING THE SCREW PARAMETERS

In the area of kinematics and robotics, several methods have been developed for computing screw parameters from displacement matrices[7,13]. A variation of these methods was used in computer graphics and animation to convert 3x3 rotation matrices to quaternions[2,8,12]. The methods use the elements of a displacement matrix to compute screw parameters and thus must perform the matrix inversion or unnecessary calculations[17]. Here, we propose a different approach, which derives the screw parameters directly from the pose matrices.

A rigid body motion transformation, ***T***, may be represented by a 3x4 matrix, {**i j k o**}, formed by four column vectors, **i**, **j**, **k**, and **o**, which each have three coordinates and represent respectively the images by ***T*** of the three basis vectors and of the origin. ***T*** is a pose and will be written ***T***(**i**,**j**,**k**,**o**).

Note that when homogeneous coordinate s are used, transformations may be represented as 4 x4 matrices, padding the above {**i j k o**} matrix with a fourth row vector (0,0,0,1). This notation introduces storage and processing redundancies when dealing with rigid body transformations, but is often preferred to a 3x4 notation, because it unifies the representations of translation, rotation, scaling, and perspective transformations. Nevertheless, we use the 3x4 notation in this paper to emphasize the meaning of the four vectors, presented above.

Consequently, the two initial and final poses, ***A*** and ***B***, that define a screw motion ***M***(**s**,**p**,*d*,*b*) may each be represented by a 3x4 matrix, which defines the rigid-body transformation that takes the global coordinate system to the associated pose. Let {$\mathbf{i}_A$ $\mathbf{j}_A$ $\mathbf{k}_A$ $\mathbf{o}_A$} denote the matrix associated with pose ***A*** and let {$\mathbf{i}_B$ $\mathbf{j}_B$ $\mathbf{k}_B$ $\mathbf{o}_B$} denote the matrix associated with pose ***B*** (see *Figure 3*). Also, to simplify the notation, let $\mathbf{i}=\mathbf{i}_B-\mathbf{i}_A$, $\mathbf{j}=\mathbf{j}_B-\mathbf{j}_A$, $\mathbf{k}=\mathbf{k}_B-\mathbf{k}_A$, and $\mathbf{o}=\mathbf{o}_B-\mathbf{o}_A$.

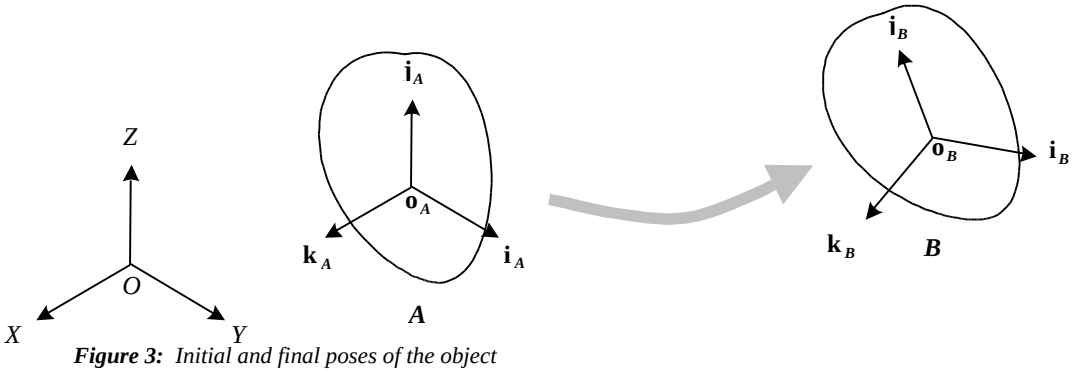


***Figure 3:*** *Initial and final poses of the object*

The four parameters **s**, **p**, *d*, and *b* may be derived from {$\mathbf{i}_A$ $\mathbf{j}_A$ $\mathbf{k}_A$ $\mathbf{o}_A$} and {$\mathbf{i}_B$ $\mathbf{j}_B$ $\mathbf{k}_B$ $\mathbf{o}_B$} by the procedure described below and comprising the following steps

1. Compute **s** from the cross-products of the three column vectors of {**i j k**} ( *Figure 4*),
2. Compute the minimal rotation angle, b, as the angle between $\mathbf{s}\times\mathbf{i}_A$ and $\mathbf{s}\times\mathbf{i}_B$, where × denotes the cross product (Figure 5),
3. Compute **p** as the center of a circular arc described by the origin, as it moves from $\mathbf{o}_A$ to $\mathbf{o}_B$ (Figure 6),
4. Compute d as **o•s**, where • denotes the dot product.

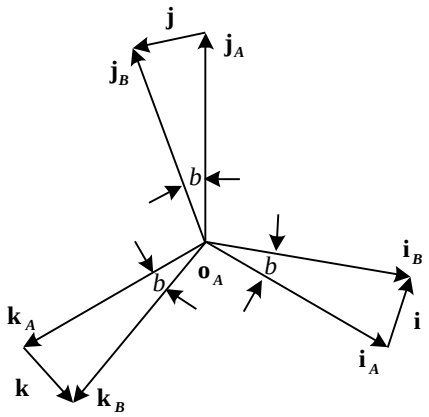


***Figure 4**: Difference vectors, as seen from the **s** direction.*

To compute the direction **s** of the screw axis, we consider only the rotational part of the screw, ignoring the images of **o** by ***A*** and ***B***. In a pure rotation, each point moves along a circle. All these circles lie in planes orthogonal to the axis of the rotation (*Figure 4*) and so to the chords joining the starting and ending point of each circular arc. Therefore, each one of the three difference vectors: **i**, **j**, and **k** is either null or orthogonal to **s**. For example, **i** represents the vector that is the chord of an arc described by the image of point (1,0,0) as it moves along a pure rotation interpolating the two orientations defined by the rotational parts of poses ***A*** and ***B***. One can easily show that when the screw motion is not a pure translation, at least two of the three vectors, **i×j**, **j×k**, and **k×i**, are non-null. Therefore, we compute **s** as:

$$\mathbf{s} = \mathbf{i}\times\mathbf{j} + \mathbf{j}\times\mathbf{k} + \mathbf{k}\times\mathbf{i}$$

When **s** is a null vector, the screw is a pure translation by vector **o**. Throughout the rest of this paper, we assume that the screw is not a pure translation and that **s** has been divided by its norm and is thus a unit vector that defines the direction of the axis of the screw .

The angle $b$ of minimal rotation is the angle between the projections of $\mathbf{i}_B$ and $\mathbf{i}_A$ on the plane orthogonal to **s** (*Figure 5*). Because these two projections have the same length, they form the two equal sides of an isosceles triangle. The third side of that triangle is **i**, which is already orthogonal to **s** (*Figure 6*). Therefore,
$\sin(b/2)=\|\mathbf{i}\|/(2\|\mathbf{s}\times\mathbf{i}_A\|)$, where $\|\mathbf{v}\|$ denotes the norm of vector **v**. Note that $b$ is always positive and less than $\pi$.

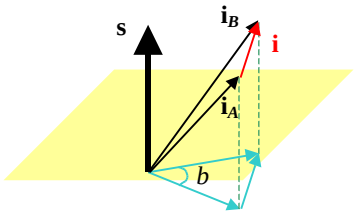


***Figure 5:** b is the angle between the projections of $\mathbf{i}_B$ and $\mathbf{i}_A$ on the plane orthogonal to **s**.*

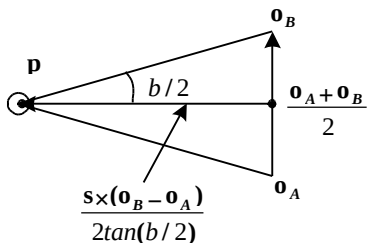


***Figure 6:** A fixed point **p** on the screw axis may be derived from the mid-point between $\mathbf{o}_A$ and $\mathbf{o}_B$.*

To assure a right handed screw, we flip the screw direction when $(\mathbf{s}\times\mathbf{i}_A)\cdot\mathbf{i}<0$.

A point **p** on the screw axis may be chosen as the center of a circular arc of angle $b$ and chord joining $\mathbf{o}_A$ and $\mathbf{o}_B$ that is orthogonal to **s**. A simple geometric construction (*Figure 6*), which starts at the mid-point between $\mathbf{o}_B$ and $\mathbf{o}_A$ and moves by the appropriate amount in the direction $\mathbf{s}\times\mathbf{o}$ leads to:

$$\mathbf{p} = (\mathbf{o}_B+\mathbf{o}_A+\mathbf{s}\times\mathbf{o}/\tan(b/2))/2$$

The translation distance $d$ is simply defined by:

$$d = \mathbf{o}\cdot\mathbf{s}$$

## COMPUTING THE SILHOUETTE EDGES

In this section, we consider that the screw motion that interpolates poses ***A*** and ***B*** is represented by its **s**, **p**, $b$, and $d$ parameters, expressed in the local coordinate system of the moving object, W. This assumption may be interpreted in three equivalent ways:

- The moving object W is already in its initial position W@***A***,
- **s** and **p** have been transformed by the inverse of ***A***,
- or ***A*** is the identity and ***B*** is the relative pose defined by the composition of the transformation associated with ***B*** followed by the transformation associated with the inverse of ***A***.

The velocity $\mathbf{v}(\mathbf{q})$ of a point **q** that is subject to a screw motion $\boldsymbol{M}(\mathbf{s},\mathbf{p},d,b)$ is the vector sum of a displacement by $d$ along **s** and a displacement by a quantity $br$ along the vector **t** that is tangent to the circle that is the projection along the **s** axis of the helix upon which **q** travels. The quantity $r$ is the radius of that circle. Thus, we have $\mathbf{v}(\mathbf{q})= d\mathbf{s}+br\mathbf{t}$.

Let **pq** stand for the vector **q**−**p**. The vector $r\mathbf{t}$ is orthogonal to both **s** and **pq** and has for magnitude the length of the projection of **pq** onto a plane orthogonal to **s**. Consequently, $r\mathbf{t}=\mathbf{s}\times\mathbf{pq}$ and we have:

$$\mathbf{v}(\mathbf{q}) = d\mathbf{s} + b\mathbf{s}\times\mathbf{pq}$$

Consider an arbitrary direction **n**, not parallel to **s**. The set of points **q** whose velocity is orthogonal to **n** is the a surface defined by $\mathbf{n}\cdot\mathbf{v}(\mathbf{q})=0$. Developing this equation yields $\mathbf{n}\cdot(d\mathbf{s}+b\mathbf{s}\times\mathbf{pq})=0$ and $\mathbf{n}\cdot d\mathbf{s}-b\boldsymbol{n}\cdot(\mathbf{s}\times\mathbf{p})+b\boldsymbol{n}\cdot(\mathbf{s}\times\mathbf{q})=0$. By exploiting the properties of the mixed product $\boldsymbol{n}\cdot(\mathbf{s}\times\mathbf{q})$, we obtain $b\boldsymbol{q}\cdot(\mathbf{n}\times\mathbf{s})=\mathbf{n}\cdot d\mathbf{s}-b\boldsymbol{n}\cdot(\mathbf{s}\times\mathbf{p})$, which is the equation of a plane, U, with normal $\mathbf{n}\times\mathbf{s}$ to which the point **q** must belong.

If we restrict **q** to lie on a specific plane, V, with normal **n**, the silhouette associated with that plane will be the line, L, that is the intersection between U and V, unless we are in the degenerate case of $\mathbf{n}\times\mathbf{s}=0$ that does not generate any silhouettes. The silhouette line separates the plane into two parts: one moving forward and one moving backward during the screw motion. The velocity vector of all points on the forward-moving part of the plane forms a positive dot-product with the outward normal to the plane. The velocity at all points of L is tangential to the plane (See *Figure 7*).

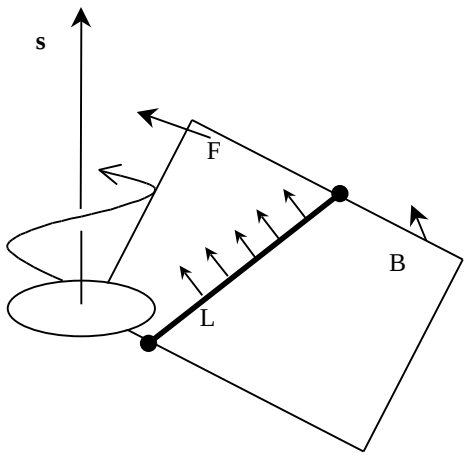


***Figure 7:*** *Silhouette line and characteristic points*

Consider a face F of a polyhedral object W. The silhouette that lies in the relative interior of F is the intersection of F with L. The tangent to L is orthogonal to **n** and to **n×s**. It may be computed as **n×(n×s)** and is the orthogonal projection of **s** onto the plane. Therefore, the intersection of the silhouette with F may be reduced to a line-face intersection in the plane V.

Consider now restricting **q** to lie on a straight line **r**+*u***t**, passing through point **r**, parallel to the unit vector **t**, and parameterized by *u*. Replacing **q** with **r**+*u***t** in $b\mathbf{q}\cdot(\mathbf{n}\times\mathbf{s})=\mathbf{n}\cdot d\mathbf{s}-b\mathbf{n}\cdot(\mathbf{s}\times\mathbf{p})$ yields:

$$b(\mathbf{r}+u\mathbf{t})\bullet(\mathbf{n}\times\mathbf{s}) = \mathbf{n}\bullet d\mathbf{s} - b\mathbf{n}\bullet(\mathbf{s}\times\mathbf{p})$$

which may be solved for u and yields:

$$u = (\mathbf{n}\bullet d\mathbf{s} - b\mathbf{n}\bullet(\mathbf{s}\times\mathbf{p}) - b\mathbf{r}\bullet(\mathbf{n}\times\mathbf{s}) ) / (b\mathbf{t}\bullet(\mathbf{n}\times\mathbf{s}) )$$

The above formula may be used for identifying the characteristic points along the edges of a polyhedron that delimit the silhouette portion of the edges.

Armed with these geometric constructions, we compute the silhouettes as the intersection of silhouette lines with faces of W and as the subsets of the edges of W that may terminate at characteristic points.

We first consider all the faces of W. For each face F, we compute the silhouette line associated with the plane that supports F and we trim it to its intersection with F.

The cost of this computation for each face is a intersection of the silhouette line with the edges that bound F, a classical line-face trimming operation that may be implemented in as a 2D calculation in the plane of F or as a series of intersections between the edges of F and the plane of points with velocity orthogonal to the normal of F.

Then, we consider each convex edge E of W. (Non-manifold edges may be treated as pseudo-manifold edges[10] and concave edges may not generate silhouettes.) Using the above equation, we compute the *u* parameters along E that correspond to characteristic points along E for the outward normals of the two faces of W that are incident upon E. These two parameters correspond to points, which¾when traversed while walking along the edge¾toggle its status between silhouette and non-silhouette. (Remember that the portion of the edge that is a silhouette is the portion which bounds a forward moving and a backward moving portion of the incident faces. The status of forward or backward moving switches at the silhouette edges for the supporting plane.)

The results of trimming the convex edges by their characteristic points and of trimming silhouette lines by the boundary of each face is the set of silhouette edges for W as it moves along the screw ***M***(**s,p**,*d*,*b*). Their union is E(W,**A**,**B**).

## ANIMATING AND SWEEPING MOVING OBJECT

Let us assume that ***M***(**s,p**,*d*,*b*) has been computed as described above. In order to display W@***P***(*t*,***A***,***B***) for a given value of *t*, we transform W by a rigid body transformation that combines the following successive primitive transformations:

5. The initial pose ***A*** which brings the object W into its starting position
6. A transformation ***K*** that brings **p** to the origin and **s** to the z-axis
7. A rotation by *tb* around the z-axis
8. A translation by *td* along the z-axis
9. The inverse $\boldsymbol{K}^{-1}$ of ***K***

A matrix representation of $\boldsymbol{K}^{-1}$ may be easily computed as {**i j s o**}, where **i** and **j** are constructed to form an orthonormal basis with **s**. The 3x4 matrix that represents the inverse of a rigid body transformation is easily obtained as a composition of a 3x3 matrix that is the transpose of the rotational part of the original matrix and a translation, which is the inverse of the original translation vector transformed by the inverted rotation. Such instances for successive values of *t* are shown *Fig. 1a*.

In order to produce an approximation for the face swept by each silhouette edge E joining two points **a** and **b**, we compute the silhouette edge for W@***A*** and compute the images **a**' and **b**' of its vertices by ***P***(ε,***A***,***B***) for ε=1/*k*, which depends on the desired accuracy of the approximation. Then we generate the quadrilateral face bounded by the four vertices (**a**, **b**, **b**', **a**'). We store all these quadrilateral faces with the associated normals in a display lists, one quadrilateral per silhouette edge.

Then, we superimpose the displays of W@***A***, W@***B***, and of instances of the display list of quadrilateral edges transformed by ***P***(ε,***A***,***B***), ***P***(2ε,***A***,***B***), ***P***(3ε,***A***,***B***),…***P***((k-1)ε,***A***,***B***).

*Figure 8* shows an example of the rendering of the area swept by a block W in motion. Note that each of the two edges shown in *Figure 8b* is cut by the characteristic point that delimit a silhouette line as indicated by an arrow. *Figure 8c* and *8d* displays the quadrilateral faces swept by each silhouette using wire-frame and shaded rendering respectively.

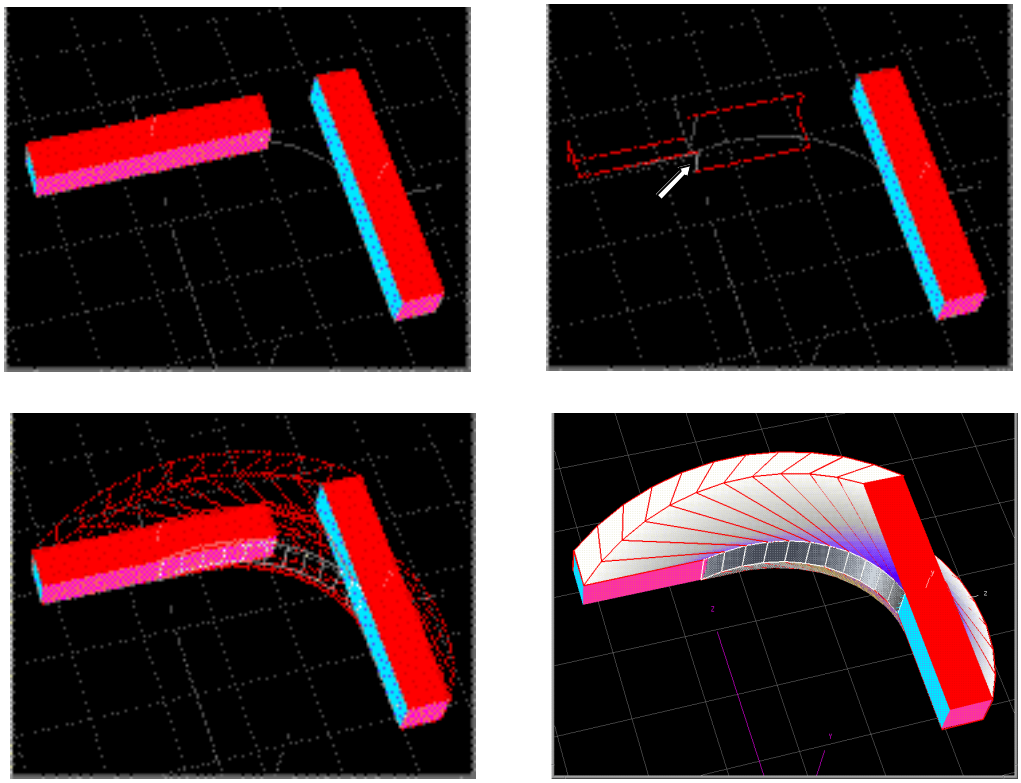

***Figure 8:*** *Envelope swept by a simple block during a screw motion. (a) The initial pose and final poses are shown top left. (b) The silhouette line (marked by a white arrow) and the silhouette edges (marked in red) are shown top right in their initial position. (c) Envelope displayed with quadrilateral faces (d) Shaded image of envelope*

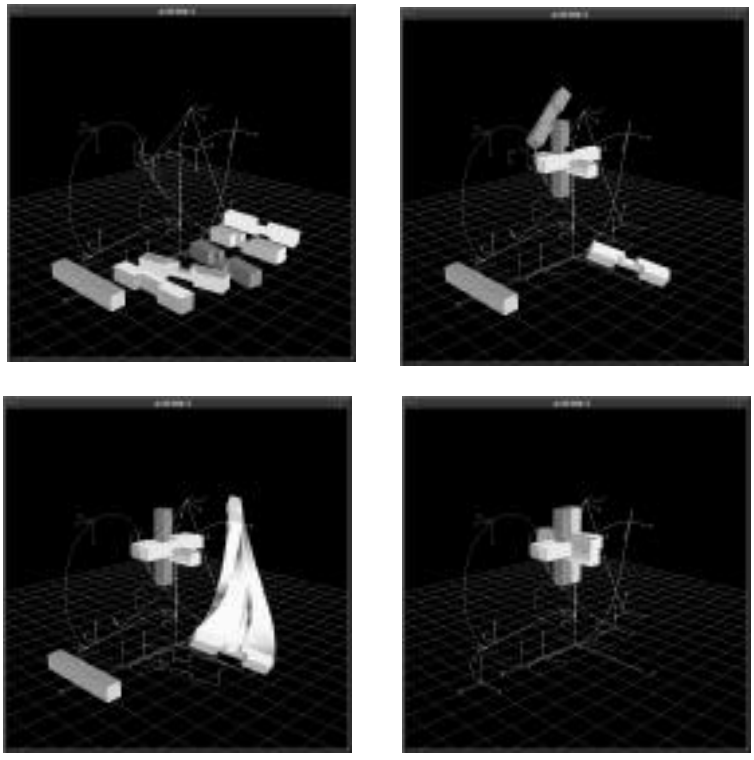

***Figure 9****: The assembly of the puzzle is explained using screw animations and swept regions. (a) The pieces are shown top left in some initial poses. (b) A step of the assembly procedure is illustrated top right showing a snapshot of the motion of a couple of pieces. (c) The region swept by one piece is shown bottom left. (d) The assembled puzzle is shown bottom right.*

*Figure 9* shows an assembly process for the puzzle composed of six pieces (*Figure 9a*). The goal is to assemble the pieces as illustrated in *Figure 9d*. The *xyz* coordinates axes in each figure shows the series of key–poses specified by a designer with the interactive pose editing facility of the Meditor system. The helix trajectories of the origin of these coordinate systems are used to visualize the paths of each piece. The user can animate the motions by turning the time dial and can control the timing of the motions by assigning the current time to the desired key-pose. *Figure 9b* is a snap shot of assembly at a given value of *t*. *Figure 9c* shows the shaded image of the region swept by one piece.

## CONCLUSIONS

We have introduced an efficient algorithm for computing a screw motion from two given poses. The algorithm is based on a geometrical approach and is simpler than commonly used methods[17].

We have also introduced a concise algorithm for computing a graphic representation of the region swept by a polyhedron moving along a piecewise screw motion. The algorithm exploits the fact that the velocity of each point of an object moving along a screw motion is constant in the coordinate system of that object. We provide simple geometric constructions for computing the silhouette of the moving object, which we define as the set of points whose velocity is tangential to the object's boundary. A superset of the faces that bound the swept region may be constructed by simply sweeping the silhouette along the screw motion. This superset does not need to be trimmed when shaded images of the swept region are produced.

## REFERENCES


1. Abdel-Malek, K., Yeh, H. J. and Othaman, S., Swept volumes: void and boundary identification. *Computer-Aided Design*, 1998, **30**(13), 1009-1018.
2. Barr, A. H., Currin, B., Gabriel, S. and Hughes, J. F., Smooth interpolation of orientations with angular velocity constraints using quaternions. In *Proceedings of SIGGRAPH '92, Computer Graphics*, 1992, 26(2), 313-320
3. Ball, R., *A Treatise on the theory of screws*. Cambridge Univ. Press. 1900.
4. Bruderlin, A. and Williams, L., Motion signal processing. In *Proceedings of SIGGRAPH '95, Computer Graphics*, 1995, 293-302
5. Martin, P. R. and Stephenson, P. C., Sweeping of three-dimensional objects. *Computer-Aided Design*, 1990, 22(4), 223-234.
6. Ohwovoriole, M. and Roth, B., An extension of screw theory. *Transaction of ASME Journal of Mechanical Design*, 1981, 103, 725-735.
7. Paul, R., *Robot manipulators*, MIT Press. 1982..
8. Pletinckx, D., Quaternion calculus as a basic tools in computer graphics. *The Visual Computer*, 1989, Vol. 5, 2-13.
9. Rossignac, J. and Requicha, A. A. G., Constant radius blending in solid modeling. *Computers in Mechanical Engineering*, July 1984, 65-73.
10. Rossignac, J. and Cardoze, Matchmaker: Manifold BReps for non-manifold r-sets. *Proceedings of the ACM Symposium on Solid Modeling*, pp. 31-41, 1999,
11. Roschel, O., Rational motion design-a survey. *Computer-Aided Design*, 1998, 30(3), 169-178
12. Shoemake, K., Animating rotation with quaternion curves. In *Proceedings of SIGGRAPH '85, Computer Graphics*, 1985, 19(3), 245-254.
13. Suh, C. and Radcliffe, C., *Kinematics & mechanisms design*. John Wiley & Sons. 1978.
14. Witkin, A. and Kass, M., Spacetime Constraints., In *Proceedings of SIGGRAPH '88, Computer Graphics*, 1988, 22(4), 159-168.
15. Yang, A. and Freudenstein, F., Application of dual-number quaternion algebra to the analysis of spatial mechanisms. *Transaction of ASME Journal of Mechanical Design*, 1964.
16. Zefran, M. and Kumar V., Interpolation schemes for rigid body motions. *Computer-Aided Design*, 1998, 30(3), 179-189.
17. Watt, A. and Watt, M., *Advanced animation and rendering techniques*, Addison-Wesley, ACM Press.1992.