Errors and Omissions in Marc Alexa's "Linear Combination of Transformations"

Charles Bloom*
OddWorld Inhabitants, Inc.

Jonathan Blow†
Bolt-Action Software, Inc.

Casey Muratori‡
RAD Game Tools, Inc.

Abstract

The affine transform is a fundamental computer graphics tool. While there are many situations which require the linear combination of such transformations, most practioners find these operations troublesome due to the non-linear nature of rotations. Alexa [2002] proposed a methodology for treating transforms as linear, and although this is a worthwhile goal, we feel that the techniques presented were sorely lacking, both in their practical implementation and in their theoretical underpinnings. This paper points out these serious flaws, and demonstrates how alternative techniques, already in limited use since the late 1990s, are much more suited to adoption as standard practices than those advocated by Alexa [2002].

1 Introduction

Rotation has long been the troublesome part of the 3D transformation. While translation, scale, and shear all admit obvious, plausible linear combination operators, rotation does not. This has lead to a variety of rotation representations and operators, and the computer graphics community has yet to settle on a standard.

Alexa [2002] boldly tries to attack the general problem of linear combination of any type of transformation. Since only the rotation portion of a typical 3D transformation matrix is combinationally non-linear to begin with, the majority of Alexa [2002] revolves around how to map rotation matrices to and from a space where the rotation portion can be treated linearly.

In this paper, we will show that Alexa's proposed transformation interpolation is seriously flawed. Rotation in particular has many different flavors of interpolation, and restricting the method of interpolation is often a hindrance, not a help. Furthermore, Alexa's chosen interpolant lacks many important qualities that are essential to a versatile representation for the linear combination of transformations.

In addition, we put forward, for the first time, a concise presentation of the properties of rotation interpolation as they apply to computer graphics. It is our hope that this will set the record straight, such that meaningful progress in standardization on rotation combination operators can be made.

The rest of this paper is organized as follows.

In section 2, we introduce the qualities one might look for in a rotation interpolant. We explain why a single interpolant cannot have all of them at the same time, and assign the appropriate qualities to the common methods of rotation interpolation.

In section 3, we discuss the topology of rotation interpolation. We point out the important aspects of this topology, and isolate specific ways in which the methods of Alexa [2002] do not adequately address problems or take advantage of possibilities presented by the topology.

In section 4, we correct Alexa's erroneous assertion that the non-geodesic nature of the exponential map implies that it is not constant speed. We then demonstrate that it actually *is* constant speed, but that it does not minimize torque, and we discuss the downsides of this tradeoff.

In section 5, we point out a very serious problem with exponential mappings that Alexa [2002] dismisses. We show where Alexa's math is erroneous, and why it is a serious problem for the exponential map to have a non-injective inverse.

In section 6, we provide examples of real-world performance criteria for transformation representations. Using Alexa's own timings, we then demonstrate that even if there was some benefit to using Alexa's proposed implementation, it would be too slow to be used in any serious application.

In section 7, we discuss a constructive approach to the rotation interpolation problem. We point out practical compromises for the central problems of rotation interpolation, and demonstrate how a thoughtful implementation can usually have the best of both worlds when it comes to rotation interpolation.

2 Interpolant Properties

Interpolation of translations and scales is relatively straightforward. Interpolation of rotations, however, has many complex subtleties. In particular, there are three primary qualities of interpolation that are of interest: torque minimization, constant angular velocity, and commutivity.

If a rotation interpolation is torque-minimal, it means that the path taken between two interpolated rotations takes the shortest path through the space of rotations. In a sense, if one considered there to be actual force involved in spinning a coordinate frame, torque-minimal interpolation "uses the least amount of torque" to rotate from one orientation to another.

If a rotation interpolation has constant angular velocity, it means that a uniform change in the interpolation parameter produces a uniform change in the resulting rotation. The

*e-mail: cbloom@cbloom.com
†e-mail: jon@bolt-action.com
‡e-mail: cmu@funkytroll.com

speed of the rotation during interpolation is constant if the rate of change of the interpolation parameter is also constant. Put a different way, the interpolation speed depends only on the derivative of the interpolation parameter, and not the parameter's actual value.

If a rotation interpolation is commutative, it means that for multiple sequential interpolations, the order of interpolation is unimportant. If you interpolate part way from A to B, then part way from there to C, the resulting orientation will be the same as if you went from C to B, and then to A, provided you still used the same coefficients.

Sadly, there is no known direct interpolation scheme which can provide all three of these (generally desirable) qualities at the same time. Out of the three primary contenders, each lacks one.

Spherical linear interpolation of quaternions (typically abbreviated "SLERP") provides torque minimization and constant angular velocity, but it does not commute. This is because the operation depends on the angle between the two rotations being interpolated. Changing the order changes the relative angles, thus you get a different result.

Linear interpolation of quaternions provides torque minimization, and it commutes, but it does not interpolate with constant angular velocity (for those unfamiliar with this operation, see **??**). This is because the linearly interpolating rotation is effectively "tunnelling" through the unit hypersphere, and then being projected back onto its surface. The projection necessarily introduces the non-linear angular velocity.

Linear interpolation of exponential maps is constant angular velocity, and it commutes, but it does not follow the torque-minimal path. This is because composition or rotations is not commutative. Therefore, addition in their log-space is not the same as multiplying them in rotation space, so they mathematically cannot follow the same path as a quaternion SLERP or LERP, which are proven to be torque-minimal. The exception to this is, of course, when the rotations do commute, which is whenever one rotation is the identity, or both rotations are occuring along the same axis, in which case the formulations are equivalent, which can be proven the same way.

It is generally assumed that there is no way to construct a single interpolant that satisfies all three of these requirements. This is due to the non-Euclidean topology of the space of rotations; ie., there is no way to unwrap a sphere onto a plane while preserving linearity, commutivatity, and the distance metric. The following section discusses this in more detail.

3 The Topology of Interpolation

Alexa [2002] does not address the global structure of transformations. Although he does not present it as such, Alexa's approach boils down to one well-known prescription: interpolate transformations in their Lie space.

All transformations are elements of a group. That group has Lie generators which define a local manifold in each neighborhood of the group. Alexa is suggesting that you take the local neighborhood around the identity operation, use the Lie basis to define a flat Reimann topology for the transformations, and then to use that topology globally. This works fine for groups that have globally flat topology, such as the Translation and Scale groups (ie the Lie algebra is commutative). Because Translation and Scale are locally flat, and we can write our transformation as a group product, the only problem with interpolation between transformations comes from the rotation part. The rotation group (in dimensions greater than R^2) is a group that does not have globally flat topology. Alexa's approach is akin to making a flat projection of the group topology, and then working in that projected space.

The rotation group has topology S^3/Z^2. S^3 is the 3-dimensional surface of a sphere (embedded in 4 dimensions); the division by Z^2 refers to the fact that opposite rotations are identical. That is, rotation around N by theta and rotation around $-N$ by $-\theta$ are the same rotation. Note that the rotation around N by theta and the rotation around N by $(2\pi - \theta)$ lead to the same orientation, but they are not the same rotation. The topology S^3 means that any flat projection (ala Alexa) is going to cause distortion. Failure to account for the Z^2 (double-cover) will cause global neighborhood errors.

Thus, two major problems arise from trying to use Alexa's method in general: distortion and global neighborhood errors. Distortion means that distances in the exponential map (Lie space) are not proportional to distances in the group. This can cause you to take unnatural curved paths through the space of transformations. Close to the identity, this distortion is small; infinitesimally close, the exponential map is a perfect coverage; the error gets worse the farther you get from the identity operation. Global neighborhood errors typically manifest by taking an unnecessary long path between two transformations. In a flat Reimann space (like R3) there is only one straight line between two points. On a sphere, there are many geodesics between two points; if you try to make an unwrapping of the surface of a sphere (to a disc), you must also consider the paths which go around the boundary of the disc. In general, there may be many ways that two different points on the unwrapping have zero distance between them; for example, a torus may be unwrapped to a rectangle, with opposite edges representing the same points in the group.

The problem with Alexa's technique in general is that it produces more distortion than rival techniques, and it doesn't account for the global topology problems at all. The only way to handle the topology problems is to explicitly know what group you are dealing with, and to account for that space. Thus, generic interpolation of transformations is impossible and undesirable.

4 Constant Speed

Now that we have covered the relevant theory of rotation interpolation, we are ready to discuss the first specific error in Alexa [2002]:

> The exponential map, on the other hand, has some drawbacks. Essentially, a straight line in parameter space doesn't necessarily map to a straight line (i.e. a geodesic) in the space of transformations. This means the linear interpolation between two transformations as defined above could have non-constant speed. Furthermore, also spline curves, which could be thought of as approximating straight lines as much as possible, are minimizers of a geometrically doubtful quantity. Nevertheless, we found the results pleasing.

Sadly, although Alexa correctly observes (or transcribes from Grassia [1998]) the concept that exponential map interpolation is not geodesic, he apparently fails to understand what this actually means. The geodesic interpolation refers

to taking the torque-minimal path through rotation space. This has nothing to do with constant angular velocity, which is an entirely separate quality. In fact, contrary to Alexa's assertion, the exponential map *is* constant speed. The fact that it is not always geodesic means that it is not torque minimal.[1] See Appendix A if you are interested in an intuitive presentation of geodesy.

5 The Non-injective Inverse

Obviously, the exponential map must be inverted to go from one of its values back to the corresponding rotation. Because the exponential map is not periodic, and rotations are, this implies that there will be multiple values in the space of the exponential map that all become the same rotation when the mapping is inverted. If you think about exponentially mapped values as $v\theta$, then it's clear that a family of values in the exponential map, which all correspond to the same rotation, is given by $v(\theta + n2\pi)$, where n is any integer.

This means that there are an infinite number of values in the exponential map space that correspond to a single rotation. While this non-injectivity causes no problems for simple operations, it causes serious problems for interpolation, which Alexa mentions:

> We would also like to point at an interesting difference to quaternions: The log-matrix representation allows angles of arbitrary degree. Computing the logarithm of a rotation by π and then multiplying this log-matrix leads to a representation of rotations more than 2π. While this could be useful in some applications it might be disadvantageous in others. For example, the interpolation between the two rotations of $\pm(\pi - \epsilon)$ results in a rotation by almost 2π rather than by a rotation by 2ϵ. However, using the tools presented in the following section this could be easily avoided.

In a moment, we will take issue with the words "this could be easily avoided". But before we do, we would first like to point out that non-injectivity of the inverse mapping is not a feature that the exponential map has which quaternions do not (as might be misconstrued from the ambiguous first sentence in the quoted paragraph above). Unit quaternions also have a non-injective inverse. However, instead of infinitely many unit quaternions corresponding to a single rotation, there are in fact only two: the family is defined by q and its inverse, $-q$. [2]

> To interpolate along the shortest path from $R_1 to R_2$ one chooses for each of the factors x_1, y_1, z_1 and x_2, y_2, z_2 the shorter path on the circle. Specifically, if the difference between two corresponding factors is larger than $|r|$, then the shorter interpolation path is via $\pm r$ rather than via 0.

Unless Alexa knows something he isn't telling us, relying on this kind of alignment for interpolation of the exponential map is far from "easy". This "neighborhooding" operation is central to all animation work, be it a single interpolation between two rotations or a cubic b-spline that interpolates four rotations at once. It is therefore essential that it be simple and efficient to take any number of rotations and put them in the proper neighborhood for interpolation.

To demonstrate why Alexa's neighborhooding operation does not make things easy, we begin by looking at the first paragraph in 6.2. Alexa states that the angle of rotation is

$(x + y + z)/\phi$

Actually, it is

$|(x, y, z)|/\phi$

which is to say that it is the magnitude of the vector (x, y, z).

The value "r" that he talks about here is (pi * phi). This "phi" factor is very confusing, so I'll drop it from now on, and just use phi = 1. The choice of phi=1 means that my rotations are :

$Rx = e^(Jx)$ $log(Rx) = x_h at$

and

$T = e^(J * ntheta)$

T is rotation around n by $theta$

And

$x =< logRx, logT >= n_x theta$

In particular,

$x, y, z = n * theta$

Are the parameters in the exponential map space.

Now, Alexa's r is just pi. Alexa then considers each parameter separately. He says that if you look at a parameter x, then values pi and $-pi$ are the same rotation. In the exponential map, x is a segment; by equating the ends, it's a circle. He's saying that the rotations around X have the topology S^1. This is true when you only do a rotation around X, but it is WRONG for a 3D rotation. Alexa is treating the global topology as $S^1 * S^1 * S^1$, which is just wrong, that's not the same as S^3, which is what he should be using. In particular, the rotations :

x, y, z and $2pi - x, y, z$

are NOT the same rotation. They are only the same if y and z are zero. The correct relation is that :

$n * thetaandn * (theta - 2pi)$

are the same rotation. If we take that theta starts in $[0, pi]$, then this means that opposite points on the surface of the sphere of vectors of magnitude pi are equal rotations.

> It seems that implementations of quaternion splines or other elaborated techniques are hardly available in common graphics APIs. Note how simple the implementation of interpolating or approximating transformation curves is with the approach presented here. One simply plugs the transform object into existing implementations for splines in Euclidean space.

As Alexa suggests, we note "how simple the implementation" is, which is, in this case, not simple at all. It is not simple because each combination of b-spline control points must occur between matrix logarithms that are in the same neighborhood, as described previously. This means that you do not simple write the addition operator and "plug the transform object in". You must instead write a large preamble that ensures the proper neighborhood. Furthermore, assuming you only want to perform the neighborhood operation when interpolating, this leaves you with two different operators (or one operator and a neighborhood operator), and thus the code for splines in Euclidean space must be made aware of this.

[1] At this point, the reader may wonder which of these two properties is more desirable. See section 7.

[2] One may ask oneself at this time, precisely how many values does one want to map to a single rotation in the ideal representation for rotation in computer graphics? See section 7.

6 A Realistic Notion of Performance

In Alexa [2002], Alexa suggests that the performance penalty involved with his method for interpolation is not an issue:

> Our current implementation needs $3 \cdot 10^{-5}$ sec to construct a transform object, which is essentially the time needed to compute the matrix logarithm. The conversion to standard matrix representation (i.e. exponentiation) requires $3 \cdot 10^{-6}$ sec. Timings have been acquired on a 1GHz Athlon PC under normal working conditions. Note that for most applications transform objects are created at the initialization of processes, while the conversion to standard representation is typically needed in in [sic] every frame. However, we have found the 3μs necessary for this conversion to be negligible in practice.

CROWD SIMULATION EXAMPLE

Unless Alexa intends for his method to be used solely for the purpose of rotating cows or a single 17-bone human animation cycle, 3μs most certainly is not negligible. A typical game engine today must run on machines less powerful that the 1GHz Athlon Alexa uses (even the XBox has only a 733mHz Pentium III), but even so, the typical game load of fifty or more characters with thirty bones each would make this method prohibitively expensive. Even assuming a 20the animation system, Alexa's method would eat a full 22of that just to convert the results. For a 60Hz app, this means you'd be spending half you entire animation budget doing nothing but conversion!

But at the same time Alexa overestimates the performance of his own method, he manages to vastly underestimate the performance of others:

> We have compared the computation times of this approach with standard techniques. A SLERP based on quaternions between two rotation matrices is about 10 times faster than our approach. However, this is only true for the linear interpolation between two transformations. Quaternion splines are substantially slower. They typically do not allow interactively adjusting the key transformations.

By "quaternion splines", we can only assume that Alexa is restricting this classification to apply to solvers or energy minimization approaches to quaternion splining. If one simply treats quaternions as linear during splining, then the performance is actually vastly superior to Alexa's proposed method. In fact, that method is currently used by one of the author's character animation systems, whose total CPU budget (including all spline evaluation) weighs in at less than just the conversion required by Alexa's method.

7 Alternatives

To practically interpolate rotations, we must get away from the difficult (S3/Z2) topology. Quaternions do this using a double-cover; the topology of quaternions is S3, which means that for each rotation there are two quaternions. Linear interpolation of quaternions is a mapping to R4, simply by allowing the quaternions in S3 to leave the surface of their sphere. (Note that there are many ways to boost rotations from S3 to R4; Casey's "Lerp" is one, but see also the Rational Map papers). The exponential map is a projection/unwrapping of the sphere to R3.

At this point, the reader may wonder which of these two properties is more desirable. Is it a good idea to trade minimal torque interpolation for constant speed? While the answer likely depends on your application, it is, as far as is known, much easier to correct the speed of an interpolation than it is to correct its path. This is because correcting the speed of interpolation is a matter of changing the rate of change of the interpolation parameter, which can be done accurately (as in the way SLERP operates) or which can be approximated for visually equivalent but less computationally expensive results. On the other hand, attempts at changing the path of interpolation typically involve solvers, as in the case of torque-minimal quaternion splines.

One may ask oneself at this time, precisely how many values does one want to map to a single rotation in the ideal representation for rotation in computer graphics? The answer, luckily for quaternions, is "two". While it may have seemed intuitively obviously that the answer was "one", it turns out that that is overly restrictive. For a free rotation, where interpolation is always intended to take the shortest path, injectivity would suffice. However, if one considers that computer graphics is concerned both with free rotation and constrained rotation, representing exactly 4π radians is essential.

The canonical example is with a constrained joint whose range of motion is more than π radians. In this situation, one wishes the joint to remember that the shortest path always passes through zero, regardless of whether or not it would be "shorter" to traverse through 2π instead. Thus, having the option to select, at any time, whether we would like to allow interpolation across 2π or not turns out to be quite a valuable tool.

C) I would take this chance to explicitly describe the "Lerp" technique (just letting quats stray from S3 into R4); maybe in an appendix. You refer to "Lerp" in section 2, but haven't really described it.

D) The bit at the end of page 2, beginning of page 3, starting with "one may ask oneself" : I think this bit is a little questionable. It is correct, but there are a few problems. For one thing, I would make it clear that we are talking about both "operations" and "states" here; that is, a rotation R represents the "operation" of rotating by R; it also represents the orientation which you get by applying R to the identity. Without this, I think it's confusing whether we are talking about "rotations" or "orientations" - really we're talking about both. Really, I think I might leave this section out of the main paper, like in an appendix, because it doesn't really address any problem in Alexa. If you're going to mention it, you should mention that orientations in the real world actually have the topology of quaternions (SU(2)) not rotations (SO(3)). The classic example of this is of course Dirac's "belt trick"; you can also find web sites of Feynman's "Candle Dances", and knot theoretical issues. That is, a rigid body in space rotations like SO(3) - that is, a 2pi rotation takes the object to itself. If, however, you attach that object to three strings that extend to infinity, a 2pi rotation is no longer the identity, but a 4pi rotation still is!

It's useful to compare this to what you have to do with quats. With quats, you start with two rotations q1,q2. You need to consider the mirrors -q1 and -q2. You have to consider

q1-¿q2, q1-¿(-q2), q2-¿(-q1), and (-q1)-¿(-q2)

Neatly enough, there are actually only two cases here,

since

q1-¿(-q2) is the same as q2-¿(-q1)

and (-q1)-¿(-q2) is the same as q1-¿q2

So, to get the shortest path, all you have to do is :

z = q1 DOT q2

if z ¿= 0 , use q1-¿q2 else , use q1-¿(-q2)

Note that this is also a better distance metric. The proper distance metric for rotations is the length of the geodesic between them on the surface of S3. This is just acos(z) for quats, so using "z" means you are correctly choosing the shorter geodesic. In the exponential map, it's not even clear that you *want* to choose the shorter geodesic, since interpolation doesn't go across geodesics on S3.

8 Conclusion

We close with the following statement from Alexa:

> The main feature of this approach is the simplicity and fle xibility [sic] in developing graphics software. We believe that many of the possible results of this approach might be generated by other means, though with considerably more progrmaming effort and use of complex numerical techniques. We hope that the simple iterative implementations of matrix exponential and logarithm find their way [into] in every graphics API.

Hopefully, after reading this paper, you will agree that the above paragraph is completely false. Alexa's approach is not the simplest, it is not the most flexible, and it is not the most efficient. It requires more complex numerical techniques than equivalent or superior methods, not less. And we hope that you will join us in ignoring Alexa's request that these techniques "find their way into every graphics API".

A Linear Interpolation of Quaternions

It is useful to point out the one can intuitively understand the geodesy of various interpolations by thinking of the familiar 3D sphere (instead of the more complicated 4D one). The sphere unwrapping analogue of the exponential map (Alexa's chosen interpolation method) is to peel the sphere back from its pole and lie it flat onto a plane (FIGURE!). By contrast, linear interpolation of quaternions is simply a projection of the sphere (FIGURE!). Spherical interpolation of quaternions operates on the sphere itself, and as such, is not an unwrapping at all (and this is why it does not commute or permit linear combinations).

B Exponential Map Neighborhood Operator

I just worked out the correct relations for choosing the shortest path in the exponential map.

Consider two vectors in the exponential map :

r1 = n1 * theta1 r2 = n2 * theta2

Here the "n" are unit vectors an the theta are angles in [0,pi]. Each rotation has a "mirror" that must be considered :

r1' = n1 * (theta1 - 2pi) r2' = n2 * (theta2 - 2pi)

We must consider four different paths. You can interpolate r1-¿r2, r1-¿r2', r1'-¿r2, or r1'-¿r2'. I used just the Euclidean distance between the vectors in the exponential map. That's actually questionable, but it's the best we can do. I call these paths "A","B","C", and "D", respectively.

Put theta1 = pi * u theta2 = pi * v

so u is in [0,1] and v is in [0,1] Also put

z = n1 * n2

So z is in [-1,1]

We find these terms :

d(A) = 0 d(B) = 1 - v + z*u d(C) = 1 - u + z*v d(D) = 2*(z+1) + (z-1)*(u+v)

You should choose the path that has the smallest "d". The first thing we see is that

d(D) ¿= Min d(A), d(B), d(C)

So that the "D" path (r1'-¿r2') is never the best one. We also see that

if z ¿= 0 , then A is the best path. That is, the rotation axes must be opposing for the "strange" paths to be favorable.

Also, if (u+v) ¡= 1, then A is the best path. The sum of angle must be ¿= pi for the strange paths to be preferred.

It's fun to plot the change-over point for when the two strange paths B and C become shorter. If you make a plot of u vs. v and draw the lines of change-over for various values of "z", you get this triangle in (u+v) ¡= 1 where the basic path A is always preferred. At z = -1, both B and C are equally good, and they start at u+v = 1. As z gets bigger, up to "1", the B lines rotate back towards v = 1, and the C lines rotate up to u = 1.

I think all this material is appropriate for an appendix. It's a good record of how to correctly handle the exponential map, and it's a good demonstration that it's definitely not "simple".