On the Transformation of Surface Normals

Andrew Glassner
Frits Post

Faculty of Mathematics and Informatics
Delft University of Technology
Julianalaan 132
2628 BL Delft
The Netherlands

Technical Memo #87-1
October 1987
Revised 3 December 1987

Abstract

Modelling transformations are commonly described with 4-by-4 transformation matrices. When an object is acted upon by such a matrix, the surface normals associated with the object also change. The normals are not transformed by the same matrix which operates on the object, which is usually a surprise when one first encounters the phenomenon. A correct way to transform the normal is easy to derive. In this note we state the problem, show the solution, resolve what appears to be a contradiction, and offer some implementation hints.

1. Problem Statement

Let us say that we are given a polygon, which we define as a region of the plane bounded by edges formed from pairwise entries of a list of n sequential, coplanar, and non-colinear vertices in $\mathbf{R}^3$; $n \geq 3$. The equation of the plane in which the polygon lies may be written

$$ax + by + cz + d = \mathbf{P} \cdot \mathbf{A} = 0$$

which is satisfied for all points (x, y, z) in the plane. In the vector form, **P** is a homogeneous row vector [x y z 1] derived from the point (x, y, z), and **A** is a coefficient column vector $[a\ b\ c\ d]^t$ from which the surface normal $[a\ b\ c]^t$ and distance from the origin d can be derived.

Let **M** be a 4-by-4 transformation matrix, under which a point **P** is transformed to $\mathbf{P'} = \mathbf{P} \cdot \mathbf{M}$. We say that a polygon is transformed by applying this transformation to each of its vertices. We require that the transformed vertices be coplanar; thus the transformed polygon will also be planar.

If we transform all the vertices then we will have a new polygon in a new plane. This new plane has a coefficient vector $[a'\ b'\ c'\ d']^t$, from which we can derive a new surface normal vector $[a'\ b'\ c']^t$ and distance from the origin d'. All points **P** on this plane satisfy

$$a'x + b'y + c'z + d' = \mathbf{P'} \cdot \mathbf{A'} = 0$$

We would like to find **A'** from **A** and **M**, from which we may recover the new normal $[a'\ b'\ c']^t$.

2. A Solution

As above, we write

$$[x\ y\ z\ 1]\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \mathbf{P} \cdot \mathbf{A} = 0$$

Let's assume that when the points **P** of the polygon are transformed by $\mathbf{P'} = \mathbf{PM}$, there exists a matrix **T** such that $\mathbf{A'} = \mathbf{TA}$:

$$\mathbf{P} \cdot \mathbf{A} = \mathbf{P'} \cdot \mathbf{A'} = (\mathbf{PM}) \cdot (\mathbf{TA}) = 0$$

By inspection we can see that $\mathbf{T} = \mathbf{M}^{-1}$ satisfies the equation, since:

$$(\mathbf{PM}) \cdot (\mathbf{M}^{-1}\mathbf{A}) = \mathbf{P} \cdot \mathbf{A} = 0 \qquad (1a)$$

So we can find the new coefficient vector **A'** by the pre-multiplication of **A** with the inverse of **M**:

$$\mathbf{A'} = \mathbf{M}^{-1}\mathbf{A} \qquad (2a)$$

But what if **M** has no inverse? This can easily happen; for example let **M** be a matrix that scales by 0, or projects all of the vertices of the polygon onto the plane z=0. In fact, there is no unique solution, any more than there is a unique surface normal (any non-zero scalar multiple of a surface normal points in the same direction).

One option is to replace $\mathbf{M}^{-1}$ in the above equations with $\mathbf{M}^*$, the adjoint of **M** ($\mathbf{M}^*$ is the transpose of the matrix of cofactors of **M**) [BLINN84]. $\mathbf{M}^*$ is attractive because it always exists, even when **M** is singular. $\mathbf{M}^*$ and $\mathbf{M}^{-1}$ are related by $\mathbf{M}^{-1} = \mathbf{M}^*/\det(\mathbf{M})$ when $\det(\mathbf{M}) \neq 0$, so when $\mathbf{M}^{-1}$ exists, $\mathbf{M}^*$ and $\mathbf{M}^{-1}$ differ only by a constant factor. Thus using $\mathbf{M}^*$ instead of $\mathbf{M}^{-1}$ in the above equations affects only the magnitude of **A'**, not its direction. Since transformation (2a) doesn't (in general) conserve the magnitude of **A** anyway, nothing is lost; if we need **A'** to have unit length then we must renormalize in either case.

In addition to always existing, another nice feature of the adjoint is that it is easier to compute than the inverse. For these reasons we recommend routine use of $\mathbf{M}^*$ instead of $\mathbf{M}^{-1}$ for implementations of the equations in this note. For clarity of presentation we will continue to write $\mathbf{M}^{-1}$ when the inverse is needed, but this should be understood to be replaced by $\mathbf{M}^*$ in an implementation.

3. Efficiency

Observe that equation (2a) involves the pre-multiplication of a vector **A** by a matrix $\mathbf{M}^{-1}$. Since points are transformed by post-multiplication ($\mathbf{P'} = \mathbf{PM}$), it would appear that we need two very similar routines in our program to handle matrix transformations: one to pre-multiply coefficient vectors and another to post-multiply points. We can eliminate this duplicity by re-writing equation (2a) in a post-multiplication form. We need to recall the identity $(\mathbf{TA}) = (\mathbf{A}^t\mathbf{T}^t)^t$, for a matrix **T** and vector **A**. Then we can rewrite Equation (1a) as

$$(\mathbf{PM}) \cdot (\mathbf{M}^{-1}\mathbf{A}) = (\mathbf{PM}) \cdot (\mathbf{A}^t[\mathbf{M}^{-1}]^t)^t = 0 \qquad (1b)$$

From which we get an alternate form for **A'**:

$$\mathbf{A'} = (\mathbf{A}^t[\mathbf{M}^{-1}]^t)^t \qquad (2b)$$

So now we can use a single piece of code that post-multiplies a vector by a matrix for both coefficient vectors and points. The input to this routine consists of the vector, the matrix, and a flag. If the flag indicates that we're transforming a point then we simply post-multiply the vector by the matrix. Alternatively, if the flag indicates that we're transforming a coefficient vector, we run a transposition loop before the transformation to convert the input matrix to its transpose. We then execute the same code as before, post-multiplying the vector by the (now transposed) input matrix. If the matrix is passed by reference then you'll probably want to run the transposition loop again after the multiplication to restore the matrix to its original form.

This is why people sometimes say you need the transpose of the inverse of the transformation matrix to transform normals: such folks prefer to use form (2b). We also advocate equation (2b), but as mentioned above we suggest use of the adjoint instead of the inverse.

4. Discussion

Section 2 is pretty straightforward, but many people find the results a surprise when encountered for the first time [1]. Before we continue, though, note that there's no guarantee that the normal vector you extract from the first three components of **A** is going to have unit length. You'll need to scale your vectors to unit length after transforming them. The nice side to all this is that they don't need to be unit length coming into the transformation, so you don't need to normalize them twice.

One common difficulty with accepting equation (2a) (or its equivalent, 2b) is well described in [1]. Imagine the following thought experiment: a plane, with attached surface normal, is rotating in space. Since the normal is fixed with respect to the plane, we would expect that this normal should move *with* the plane. This would mean that the matrices transforming the plane and the normal are the same, though equation (1a) says that they're inverses! We can reconcile this apparent contradiction with some equation juggling.

Since a pure rotation matrix **R** is orthogonal, its transpose is its inverse. Thus, $(\mathbf{R}^{-1})^t = (\mathbf{R}^t)^t = \mathbf{R}$ for a pure rotation matrix **R** (also note that since the determinant of $\mathbf{R} \equiv 1$, its transpose is also its adjoint, so $(\mathbf{R}^*)^t = (\mathbf{R}^t)^t = \mathbf{R}$).

Now consider a pure rotation matrix **R** playing the part of **M** in equation (2b). Since $(\mathbf{R}^{-1})^t = \mathbf{R}$, we find $\mathbf{A}' = (\mathbf{A}^t[\mathbf{M}^{-1}]^t)^t = (\mathbf{A}^t[\mathbf{R}^{-1}]^t)^t = (\mathbf{A}^t\mathbf{R})^t$. Now we have the explanation of our thought experiment: when the transformation matrix is pure rotation, its transpose is its inverse, so the same matrix operates on both the points and the normals. So for pure rotation, the matrix transforming the points really *is* the same matrix transforming the normals! In this special case, the normal actually does move with the plane. More general types of transformations (for example, non-differential scaling as described in [1]) do not share this feature, so they require use of the inverse or adjoint.

An interesting corollary of the results in Section 2 is the following. Imagine a viewing transformation matrix **V**, which transforms points from object space to screen space. We have seen that a coefficient vector in object space may be transformed to screen space under $\mathbf{V}^{-1}$. Now imagine that you have a coefficient vector in **screen** space that you wish to transform into object space. The inverse of the viewing matrix is $\mathbf{V}^{-1}$, so the coefficient vector is transformed by $[\mathbf{V}^{-1}]^{-1} = \mathbf{V}$! So if you compute plane equations in screen space, you can easily transform them to object space simply by transforming them by the original viewing transformation **V**.

5. References

[1] Haines, Eric, "Abnormal Normals," Technical Report, 3D/Eye, Inc, September 1987.

[2] Blinn, James, "The Algebraic Properties of Homogeneous Second Order Surfaces," Siggraph '84 course notes *The Mathematics of Computer Graphics*