Folding Algorithm.

Marcin Kik

March 15, 2017

email: mki1967@gmail.com, www: http://cs.pwr.edu.pl/kik/

Abstract

We present analysis underlying the folding algorithm implemented in mki3d and et-editor.

The input contains four vectors A_1, A_2, B_1, B_2 from $\mathbb{R}^3$, such that their lengths $|A_1| = |A_2| = |B_1| = |B_2| = 1$. Let $O = (0, 0, 0) \in \mathbb{R}^3$. We want to find a point $V \in \mathbb{R}^3$ that is result of both: the rotation of the point B_1 around the line OA_1 and the rotation of the point B_2 around the line OA_2. Generally, we have three possible cases:

1. there is no such point, or
2. there is only single such point (on the plane OA_1A_2), or
3. there are two such points on both sides of the plane OA_1A_2.

The input also contains a point $K \in \mathbb{R}^3$ that is outside the plane OA_1A_2. Point K indicates on which side of the plane should be the point V. We present a sequence of equations that yield the method of computing V.

Assume, that we have input for which $V = (V_x, V_y, V_z) = (x, y, z)$ exists. We have to find the values of x, y, and z. First note that the length of V is $|V| = 1$ so we have scalar product $V \cdot V = 1$. This implies:

$$x^2 + y^2 + z^2 = 1. \tag{1}$$

Since $V - B_i \perp A_i$ (i.e. $V - B_i$is orthogonal to A_i), we have:

$$A_i \cdot (V - B_i) = 0. \tag{2}$$

This implies that $A_i \cdot V = A_i B_i$ which is equivalent to

$$xA_{i,x} + yA_{i,y} + zA_{i,x} = A_i \cdot B_i. \tag{3}$$

(We use notation $A_i = (A_{i,x}, A_{i,y}, A_{i,z})$.) Thus we have:

$$\begin{cases} xA_{1,x}A_{2,x} = (A_1B_1 - yA_{1,y} - zA_{1,z})A_{2,x} & \text{, and} \\ xA_{2,x}A_{1,x} = (A_2B_2 - yA_{2,y} - zA_{2,z})A_{1,x} & . \end{cases} \tag{4}$$

By equality of left sides:

$$(A_1B_1 - yA_{1,y} - zA_{1,z})A_{2,x} = (A_2B_2 - yA_{2,y} - zA_{2,z})A_{1,x}. \tag{5}$$

Let us multiply and group by z, y, and remaining components:

$$z(A_{2,z}A_{1,x} - A_{1,z}A_{2,x}) = y(A_{1,y}A_{2,x} - A_{2,y}A_{1,x}) + A_2B_2A_{1,x} - A_1B_1A_{2,x}. \tag{6}$$

Now assume that $m = A_{2,z}A_{1,x} - A_{1,z}A_{2,x} \neq 0$. (Otherwise, we could consider $\{x, y\}$or $\{x, z\}$ instead of $\{y, z\}$.) Then

$$z = y \cdot p + q, \tag{7}$$

where

$$p = \frac{A_{1,y}A_{2,x} - A_{2,y}A_{1,x}}{m} \tag{8}$$

and

$$q = \frac{A_2B_2A_{1,x} - A_1B_1A_{2,x}}{m}. \tag{9}$$

By Equation 1 we have $x^2 = 1 - y^2 - z^2$ and hence

$$x^2A_{2,x}^2 = (1 - y^2 - z^2)A_{2,x}^2. \tag{10}$$

On the other side, by Equation 3 for $i = 2$:

$$x^2A_{2,x}^2 = (A_2B_2 - yA_{2,y} - zA_{2,z})^2. \tag{11}$$

By 10 and 11 we have

$$(1 - y^2 - z^2)A_{2,x}^2 = (A_2B_2 - yA_{2,y} - zA_{2,z})^2. \tag{12}$$

Using 7 we rewrite it as a square equation on y:

$$\begin{aligned}
(1 - y^2 - (py + q)^2)A_{2,x}^2 &= (A_2B_2 - yA_{2,y} - (py + q)A_{2,z})^2 \\
(1 - y^2 - p^2y^2 - 2pqy - q^2)A_{2,x}^2 &= (A_2B_2 - qA_{2,z} - y(A_{2,y} + pA_{2,z}))^2 \\
(-(1 + p^2)y^2 - 2pqy + (1 - q^2))A_{2,x}^2 &= (A_2B_2 - qA_{2,z})^2 \\
&\quad -2(A_{2,y} + pA_{2,z})(A_2B_2 - qA_{2,z})y \\
&\quad +(A_{2,y} + pA_{2,z})^2y^2.
\end{aligned}$$

Let us move everything on the right side:

$$0 = ay^2 + by + c, \tag{13}$$

where

$$\begin{aligned}
a &= ((1 + p^2)A_{2,x}^2 + (A_{2,y} + pA_{2,z})^2, \\
b &= (2pqA_{2,x}^2 - 2(A_{2,y} + pA_{2,z})(A_2B_2 - qA_{2,z}), \\
c &= (q^2 - 1)A_{2,x}^2 + (A_2B_2 - qA_{2,z})^2.
\end{aligned}$$

Now we can solve it. First compute $\Delta = b^2 - 4ac$. If $\Delta < 0$ then there is no solution. Otherwise, let:

$$
\begin{aligned}
y_1 &= \frac{-b - \sqrt{\Delta}}{2a}, \\
y_2 &= \frac{-b + \sqrt{\Delta}}{2a}.
\end{aligned}
$$

By 7 we can compute, for each y_j, the corresponding z_j. The assumption $m \neq 0$ excludes the case $A_{1,x} = A_{2,x} = 0$. Thus, by 3, we have at least one $i \in \{1, 2\}$ that we can use for computing the corresponding x_j:

$$
x_j = \frac{A_i \cdot B_i - y_j A_{i,y} - z_j A_{i,z}}{A_{i,x}}. \tag{14}
$$

Let $V_j = (x_j, y_j, x_j)$. Finally, if $V_1 \neq V_2$, we have to decide, which solution should be selected. Recall that we have the input point K that should be used for this purpose. $V_1$and V_2 should be on different sides of the plane OA_1A_2. Let $\det[W_1, W_2, W_3]$ denote the determinant of the matrix with the columns W_1, W_2, W_3. We should select $V = V_i$, such that $\det[A_1, A_2, K] \cdot \det[A_1, A_2, V_i] > 0$.

The working JavaScript implementation of the folding algorithm can be found in the file mki3d_constructive.js. The method described in this document is implemented in the function named `mki3d.findCenteredFolding`.