AnimeraMera – Local Rotation Axis

This document describes some of the problems that you have to solve during the assignment when transferring the animation from one skeleton to the other. It is provided to complement the definition of the problem already given with some mathematical formulations and visualization of the skeletons.

Consider Figure 1, where two skeletons are presented. The skeleton on the right side is the one produced by iPi Motion Capture process, and the one on the left produced by Maya's default HumanIK facility. At first glance, it can be seen that they use different orientations in some joints, and the orientations and displacements at each joint are always relative to the previous joint.

The problem posed in the assignment is to transfer the animation from the skeleton on the right to the skeleton on the left. If we were to copy the keyframes from the animation and use them with the skeleton on the left, it would not represent the same animation since the joint orientations are different in comparison to the skeleton on the right. This problem can be clearly seen in joint ***is2***, which is in a different coordinate system compared to ***s2***.

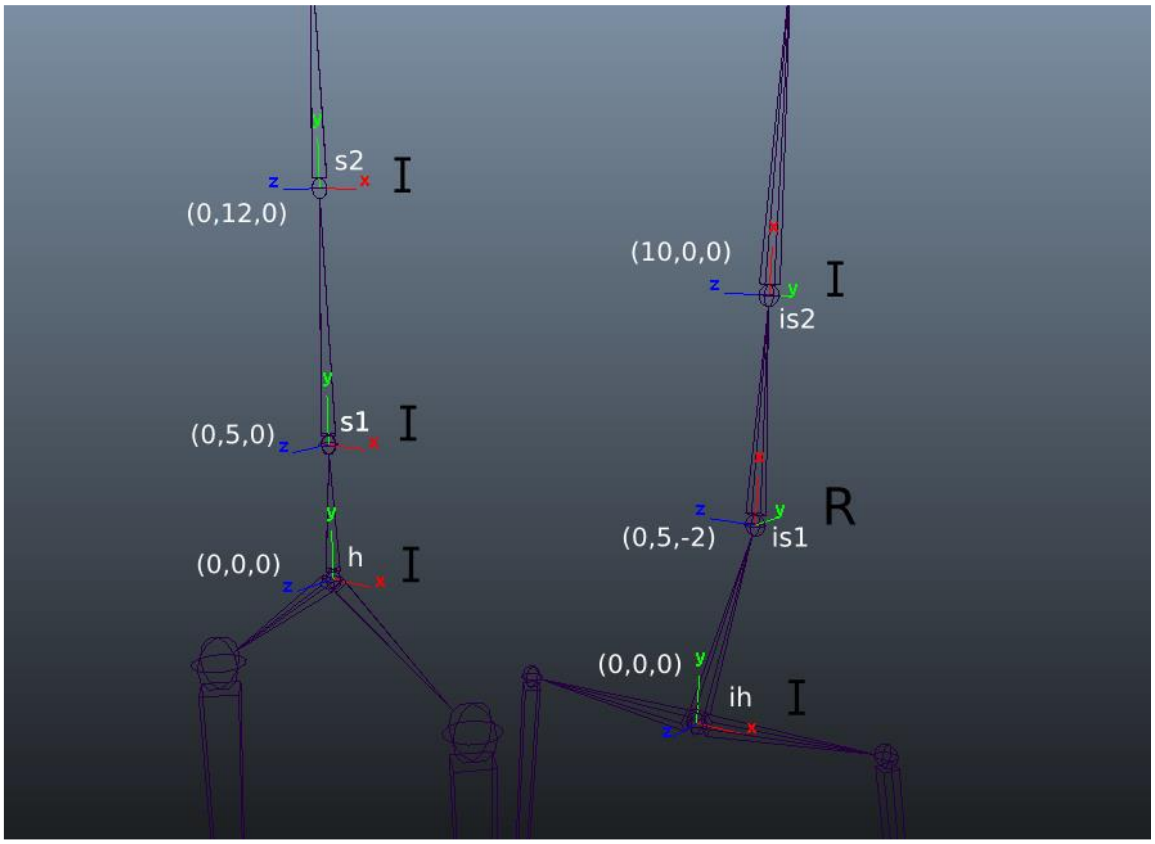


Figure 1: Skeletons from Maya (left) and iPi soft (right).

Let us assume for simplicity, that ***h*** and ***ih*** are in the **same position**, and they share the **same local rotation axis**. There is nothing to fix here, when transferring the animation from ***ih*** to ***h*** a simple copy of the keyframes is needed. When it comes to the next case, the joints ***s1*** and ***is1*** have a different orientation. Here, we will need to convert the rotation in each of the keyframes from the coordinate system of ***is1*** to the coordinate system of ***s1*** before transferring it. It is possible to do this in multiple ways, the following explanation is one of the solutions.

When the rotation is evaluated, the final rotation is composed from the rotation in the keyframe and rotation of all the joints that are between the joint in question and the root joint. If we define the rotation specified by the keyframes from ***is1*** as "**k**", in our case for the right skeleton this would be:

$$ih * is1 * k = final\ rotation$$

Thus we can see that if we simply copy the keyframe from the joint ***is1*** on the right to the joint ***s1*** on the left we would get:

$$h * s1 * k \neq final\ rotation$$

Since ***is1*** is not equal to ***s1*** this will not be the same rotation.

To solve this we start by isolating the rotation from the keyframe, this can be done by multiplying the final rotation by the inverse of the bindpose:

$$(ih\ *\ is1)^{-1} * ih * is1 * k = k$$

The rotation **k** is specified in the coordinates of the joint ***is1***, to convert this rotation to the standard coordinate space a change of basis is performed as follows:

$$ih^{-1} * k * \overset{\text{IH}}{h} = k'$$

Where **k'** is the rotation **k** in the standard coordinate space. Thus the final rotation for the joint ***s1*** would be:

$$h * s1 * k' = final\ rotation$$

As this process continues down the skeleton DAG, things get a bit more complicated. Since every joint needs to accumulate all the rotations from previous joints all the way to the root node. Let us look at the calculations needed when translating the keyframes from ***is2*** to ***s2***.

First isolating the rotation from the keyframe:

$$(ih\ *\ is1\ *\ is2)^{-1} * ih\ *\ is1\ *\ is2\ *\ k\ =\ k$$

Then the change of basis:

$$(is1\ *\ ih)^{-1} *\ k\ *\ is1\ *\ ih\ = k'$$

Here are both formulas expressed with words:

$$bindposeSourceInversed * sourceRotation = isolatedKeyframeRotation$$

$$sourceParentInversed * isolatedKeyframeRotation * sourceParent = translatedKeyframeRotation$$

You might find a solution slightly different in terms of how to define the matrices and the order of multiplication.

Remember that these **rotation** matrices are orthogonal and invertible[1], and the inverse of a rotation matrix is its transpose. Summarizing some of these mathematical properties:

$$(AB)^T = B^T A^T$$

$$A^{-1} = A^T \; if \; A \; is \; a \; pure \; rotation \; Matrix$$

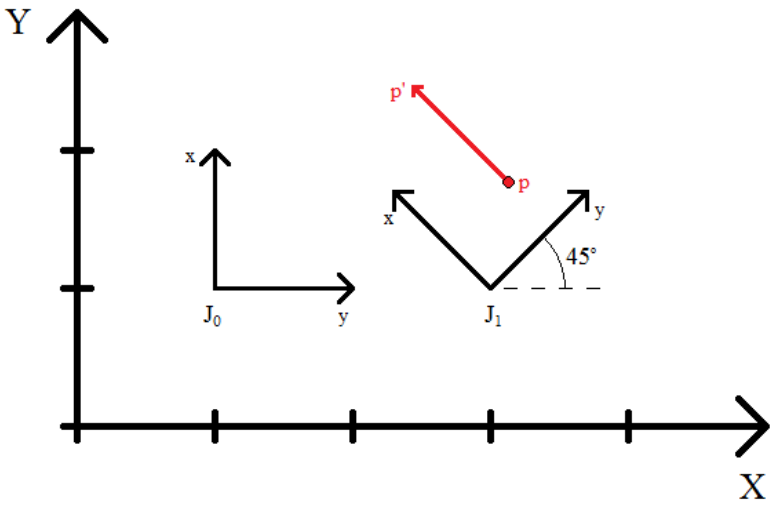
---

[1]http://en.wikipedia.org/wiki/Rotation_matrix

Here is an in depth example of how the transformation **T** is represented in standard base coordinates instead of the coordinate space of $\mathbf{J_1}$:

$$J_0: \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix} \qquad J_1: \begin{bmatrix} cos45 & -sin45 & 0 & 0 \\ sin45 & cos45 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 2 & 0 & 1 \end{bmatrix}$$

$$T \text{ with respect to} J_1 := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$



Because **T** is with respect to $\mathbf{J_1}$, it will translate any point **p** along $\mathbf{J_1}$ x axis.

How do we represent the same transform T with respect to the standard base?

$$T' = (J_1 * J_0)^{-1} * T * (J_1 * J_0)$$

$$T' = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -cos45 & cos45 & 0 & 1 \end{bmatrix}$$

Which means T' is basically a translation vector of length 1 in direction (-1, 1).