Supplementary Notes: Inverse Kinematics Problems with Exact Hessian Matrices

Kenny Erleben
University of Copenhagen
kenny@di.ku.dk

Sheldon Andrews
École de technologie supérieure
sheldon.andrews@etsmtl.ca

1 Introduction

These supplementary notes are provided in an educational spirit. Section 2 shows how the cross-product formula for computing columns of the Jacobian emerges from direct calculus. Section 5 extends to arbitrary Euler angles. Our work is not limited to homogeneous coordinates or Euler angles. However, these representations are well known and chosen merely to present to a wider audience. In fact our supplementary implementations use quaternions. We refrain from using exponential maps as they in our opinion are less accessible for a general audience, although offer a nice re-parameterization [1].

We outline how to compute the Jacobian in Section 3.1 and in Section 3.2 we work on the Hessian. We also address necessary changes to add a moving root frame in Section 3.3. Our notes finalize by presenting detailed pseudo code algorithms in Section 4. Please see our paper and supplementary code for further information.

2 Mathematical Preliminaries

Let us consider a single IK joint, to be general we label this the k^{th} joint. This joint is equivalent to a rigid body transformation, which we for notational convenience may write use a homogeneous coordinate notation, $\mathbf{T}_k$,

$$\begin{bmatrix}\mathbf{p}\\1\end{bmatrix}_{k-1} = \mathbf{T}_k \begin{bmatrix}\mathbf{p}\\1\end{bmatrix}_k , \tag{1a}$$

$$= \underbrace{\begin{bmatrix}\mathbf{R}(\alpha_k,\beta_k,\gamma_k) & \mathbf{t}_k\\ \mathbf{0}^T & 1\end{bmatrix}}_{\equiv \mathbf{T}_k} \begin{bmatrix}\mathbf{p}\\1\end{bmatrix}_k . \tag{1b}$$

That transforms a vector $\mathbf{p}$ from the k^{th} joint frame into the $(k-1)^{\text{th}}$ joint frame. The rigid body transformation consist of a rotation $\mathbf{R}(\alpha_k,\beta_k,\gamma_k)$ that we have parameterized using the joint angles α_k, β_k, and γ_k, and a joint vector $\mathbf{t}_k$ (also called the link vector). We consider the link vector to be constant in this work, although one can generalize this to be variable to model prismatic joints. One may define the rotation part in many different ways. To be specific in this work we adopt a ZYZ Euler angle convention,

$$\mathbf{R}(\alpha_k,\beta_k,\gamma_k) \equiv \mathbf{R}_Z(\alpha_k)\mathbf{R}_Y(\beta_k)\mathbf{R}_Z(\gamma_k) \tag{2}$$

where the individual rotation matrices $\mathbf{R}_Z$ and $\mathbf{R}_Y$ are defined as

$$\mathbf{R}_Z(\theta) \equiv \begin{bmatrix}\cos(\theta) & -\sin(\theta) & 0\\ \sin(\theta) & \cos(\theta) & 0\\ 0 & 0 & 1\end{bmatrix}, \tag{3a}$$

$$\mathbf{R}_Y(\theta) \equiv \begin{bmatrix}\cos(\theta) & 0 & \sin(\theta)\\ 0 & 1 & 0\\ -\sin(\theta) & 0 & \cos(\theta)\end{bmatrix} \tag{3b}$$

where θ is used as a generic placeholder in above definitions.

If we consider the derivative of the rotation matrices with respect to their respective joint angle argument then we find

$$\frac{\partial \mathbf{R}_Z}{\partial \theta} \equiv \begin{bmatrix}0 & 1 & 0\\ -1 & 0 & 0\\ 0 & 0 & 0\end{bmatrix} \mathbf{R}_z = \mathbf{C}(\mathbf{k})\,\mathbf{R}_Z , \tag{4a}$$

$$\frac{\partial \mathbf{R}_y}{\partial \theta} \equiv \begin{bmatrix}0 & 0 & -1\\ 0 & 0 & 0\\ 1 & 0 & 0\end{bmatrix} \mathbf{R}_y = \mathbf{C}(\mathbf{j})\,\mathbf{R}_y \tag{4b}$$

where we defined the cross product matrix as follows

$$\mathbf{C}(\mathbf{p}) \equiv \begin{bmatrix}0 & z & -y\\ -z & 0 & x\\ y & -x & 0\end{bmatrix} \tag{5}$$

where $\mathbf{p} = \begin{bmatrix}x & y & z\end{bmatrix}^T$ is just a generic placeholder vector used to make the definition. We also make use of the unit-axis vectors

$$\mathbf{i} \equiv \begin{bmatrix}1 & 0 & 0\end{bmatrix}^T , \tag{6a}$$

$$\mathbf{j} \equiv \begin{bmatrix}0 & 1 & 0\end{bmatrix}^T , \tag{6b}$$

$$\mathbf{k} \equiv \begin{bmatrix}0 & 0 & 1\end{bmatrix}^T . \tag{6c}$$

The cross product matrix simply allows us to rewrite a cross product in terms of a matrix multiplication. One may show algebraically that given two 3D vectors $\mathbf{a}$ and $\mathbf{b}$ then

$$\mathbf{a} \times \mathbf{b} = \mathbf{C}(\mathbf{a})\,\mathbf{b} = -\mathbf{C}(\mathbf{b})\,\mathbf{a} . \tag{7}$$

If $\mathbf{R}$ denotes some arbitrary rotation then we recall that the cross product obeys the rule

$$\mathbf{R}\,\mathbf{a} \times \mathbf{R}\,\mathbf{b} = \mathbf{R}\ (\mathbf{a} \times \mathbf{b})\ , \tag{8a}$$

$$\mathbf{R}\,\mathbf{C}(\mathbf{a})\,\mathbf{b} = \mathbf{C}(\mathbf{R}\,\mathbf{a})\,\mathbf{R}\,\mathbf{b} . \tag{8b}$$

In particular, the last version will become important when we work with simplifying our IK transformations. We now have all the necessary components to derive the derivatives of the bone transforms with respect to the joint angles,

$$\frac{\partial \mathbf{T}_k}{\partial \alpha_k} = \begin{bmatrix}\mathbf{C}(\mathbf{k})\mathbf{R}_Z(\alpha_k)\mathbf{R}_Y(\beta_k)\mathbf{R}_Z(\gamma_k) & \mathbf{0} \\ \mathbf{0}^T & 0\end{bmatrix} , \tag{9a}$$

$$\frac{\partial \mathbf{T}_k}{\partial \beta_k} = \begin{bmatrix}\mathbf{R}_Z(\alpha_k)\mathbf{C}(\mathbf{j})\mathbf{R}_Y(\beta_k)\mathbf{R}_Z(\gamma_k) & \mathbf{0} \\ \mathbf{0}^T & 0\end{bmatrix} , \tag{9b}$$

$$\frac{\partial \mathbf{T}_k}{\partial \gamma_k} = \begin{bmatrix}\mathbf{R}_Z(\alpha_k)\mathbf{R}_Y(\beta_k)\mathbf{C}(\mathbf{k})\mathbf{R}_Z(\gamma_k) & \mathbf{0} \\ \mathbf{0}^T & 0\end{bmatrix} . \tag{9c}$$

Using the rule of working with rotation matrices in equation (8b) we rewrite this as

$$\frac{\partial \mathbf{T}_k}{\partial \alpha_k} = \begin{bmatrix}\mathbf{C}(\mathbf{k})\mathbf{R} & \mathbf{0} \\ \mathbf{0}^T & 0\end{bmatrix} , \tag{10a}$$

$$\frac{\partial \mathbf{T}_k}{\partial \beta_k} = \begin{bmatrix}\mathbf{C}(\mathbf{j}')\mathbf{R} & \mathbf{0} \\ \mathbf{0}^T & 0\end{bmatrix} , \tag{10b}$$

$$\frac{\partial \mathbf{T}_k}{\partial \gamma_k} = \begin{bmatrix}\mathbf{C}(\mathbf{k}')\mathbf{R} & \mathbf{0} \\ \mathbf{0}^T & 0\end{bmatrix} \tag{10c}$$

where we omitted writing the explicit angle arguments and we defined the instantaneous joint axis vectors wrt. frame $(k-1)$ as

$$\mathbf{j}' = \mathbf{R}_Z(\alpha_k)\,\mathbf{j} , \tag{11a}$$

$$\mathbf{k}' = \mathbf{R}_Z(\alpha_k)\,\mathbf{R}_Y(\beta_k)\,\mathbf{k} . \tag{11b}$$

Now we may put our results together to compute the derivative of a vector that is transformed by a IK bone. That is

$$\frac{\partial}{\partial \alpha_k}\begin{bmatrix}\mathbf{p} \\ 1\end{bmatrix}_{k-1} = \frac{\partial \mathbf{T}_k}{\partial \alpha_k}\begin{bmatrix}\mathbf{p} \\ 1\end{bmatrix}_k = \begin{bmatrix}\mathbf{k} \times \Delta\mathbf{p} \\ 0\end{bmatrix} , \tag{12a}$$

$$\frac{\partial}{\partial \beta_k}\begin{bmatrix}\mathbf{p} \\ 1\end{bmatrix}_{k-1} = \frac{\partial \mathbf{T}_k}{\partial \beta_k}\begin{bmatrix}\mathbf{p} \\ 1\end{bmatrix}_k = \begin{bmatrix}\mathbf{j}' \times \Delta\mathbf{p} \\ 0\end{bmatrix} , \tag{12b}$$

$$\frac{\partial}{\partial \gamma_k}\begin{bmatrix}\mathbf{p} \\ 1\end{bmatrix}_{k-1} = \frac{\partial \mathbf{T}_k}{\partial \gamma_k}\begin{bmatrix}\mathbf{p} \\ 1\end{bmatrix}_k = \begin{bmatrix}\mathbf{k}' \times \Delta\mathbf{p} \\ 0\end{bmatrix} \tag{12c}$$

where we defined $\Delta\mathbf{p} \equiv \mathbf{R}\left[\mathbf{p}\right]_k$. These equations offer us the general insight that we may compute the derivative of a single degree of freedom (DOF) rotation by transforming the joint axis into the $(k-1)^{\text{th}}$ frame and take the cross product of this instantaneous axis with the vector $\mathbf{p}$ rotated from the k^{th} frame into the $(k-1)^{\text{th}}$ frame.

3 Building the Matrices

Last section build up mathematical preliminaries and made a mathematical derivation showing that "cross-product" between the joint origin to end-effect arm and angular velocity vectors gives the positional derivative of the end-effector position with respect to the rotation angle. The "cross-product" interpretation is quite well known from rigid body physics, and quite independent of whichever way one parameterizes the rotation axes. With the mathematics in place we can know show how to compute the Jacobian, the Hessian and extend to a moving root frame too.

3.1 Computing the Jacobian

Now let us consider that the i^{th} index of θ can be either the α, β or γ angle of the k^{th} joint. That is,

$$\theta_i \in \{\alpha_k, \beta_k, \gamma_k\}\ . \tag{13}$$

Let us define the homogeneous matrix concatenation as $\mathbf{X}_a^b \equiv \mathbf{T}_a, \mathbf{T}_{a+1} \cdots \mathbf{T}_{b-1}\mathbf{T}_b$ for $a \leq b$. Using this notation we can write compactly the Jacobian with respect to the i^{th} joint as

$$\frac{\partial \mathbf{F}(\theta)}{\partial \theta_i} = \begin{cases}\mathbf{X}_0^{k-1}\frac{\partial \mathbf{T}_k}{\partial \theta_i}\mathbf{X}_{k+1}^N\begin{bmatrix}\mathbf{t} \\ 1\end{bmatrix}_N & \text{if } \theta_i \in \{\alpha_k, \beta_k, \gamma_k\} \\ \mathbf{0} & \text{otherwise}\end{cases} . \tag{14}$$

From this we realize that

$$\begin{bmatrix}\mathbf{t} \\ 1\end{bmatrix}_k = \mathbf{X}_{k+1}^N\begin{bmatrix}\mathbf{t} \\ 1\end{bmatrix}_N \tag{15}$$

is simply the tool vector in the $(k+1)^{\text{th}}$ joint frame. Hence, when applying results from (10) we find

$$\frac{\partial \mathbf{F}(\theta)}{\partial \alpha_k} = \mathbf{X}_0^{k-1} \frac{\partial \mathbf{T}_k}{\partial \alpha_k} \begin{bmatrix} \mathbf{t} \\ 1 \end{bmatrix}_k = \mathbf{X}_0^{k-1} \begin{bmatrix} \mathbf{k} \times \Delta \mathbf{p} \\ 0 \end{bmatrix}, \tag{16a}$$

$$\frac{\partial \mathbf{F}(\theta)}{\partial \beta_k} = \mathbf{X}_0^{k-1} \frac{\partial \mathbf{T}_k}{\partial \beta_k} \begin{bmatrix} \mathbf{t} \\ 1 \end{bmatrix}_k = \mathbf{X}_0^{k-1} \begin{bmatrix} \mathbf{j}' \times \Delta \mathbf{p} \\ 0 \end{bmatrix}, \tag{16b}$$

$$\frac{\partial \mathbf{F}(\theta)}{\partial \gamma_k} = \mathbf{X}_0^{k-1} \frac{\partial \mathbf{T}_k}{\partial \gamma_k} \begin{bmatrix} \mathbf{t} \\ 1 \end{bmatrix}_k = \mathbf{X}_0^{k-1} \begin{bmatrix} \mathbf{k}' \times \Delta \mathbf{p} \\ 0 \end{bmatrix} \tag{16c}$$

where $\Delta\mathbf{p}$ is the tool vector with respect to the k^{th} joint frame and $\mathbf{k}$, $\mathbf{k}'$ and $\mathbf{j}'$ are the instantaneous joint axes of our ZYZ Euler rotation expressed in the $(k-1)^{\text{th}}$ frame.

We notice that the homogeneous coordinates in all above equations have become zero. Hence the effect of $\mathbf{X}_0^{k-1}$ is to rotate the directional cross-product vector from frame $(k-1)$ into the world coordinate frame. This important observation allows us to make the shortcut of simply computing the cross product using world coordinate system quantities. This is application of the rule from equation (8b). Using this observation, and dropping homogeneous coordinates, we finally obtain

$$\frac{\partial \mathbf{F}(\theta)}{\partial \alpha_k} = \left[\mathbf{k}\right]_{WCS} \times \Delta \mathbf{p}_{WCS}\,, \tag{17a}$$

$$\frac{\partial \mathbf{F}(\theta)}{\partial \beta_k} = \left[\mathbf{R}_Z(\alpha_k)\mathbf{j}\right]_{WCS} \times \Delta \mathbf{p}_{WCS}\,, \tag{17b}$$

$$\frac{\partial \mathbf{F}(\theta)}{\partial \gamma_k} = \left[\mathbf{R}_Z(\alpha_k)\mathbf{R}_Y(\beta_k)\mathbf{k}\right]_{WCS} \times \Delta \mathbf{p}_{WCS} \tag{17c}$$

where $\Delta\mathbf{p}_{WCS} = \left(\mathbf{e} - \left[\mathbf{t}_k\right]_{WCS}\right)$ is the vector difference between the end-effector position and the position of the joint origin in the world coordinate system.

3.2 Computing the Hessian

The $\mathbf{K}$-tensor is defined as

$$\mathbf{K}_{ij} = \frac{\partial}{\partial \theta_j}\left(\frac{\partial \mathbf{F}}{\partial \theta_i}\right), \tag{18a}$$

$$= \mathbf{X}_0^{h-1} \frac{\partial \mathbf{T}_h}{\partial \theta_j} \mathbf{X}_{h+1}^{k-1} \frac{\partial \mathbf{T}_k}{\partial \theta_i} \mathbf{X}_{k+1}^{N} \begin{bmatrix} \mathbf{t} \\ 1 \end{bmatrix}_N \tag{18b}$$

where we assume $\theta_i \in \{\alpha_k, \beta_k, \gamma_k\}$ and $\theta_j \in \{\alpha_h, \beta_h, \gamma_h\}$. We notice this is simply a recursive application of the transformation rules we used for computing the Jacobian in the first place. The first derivative will change the position into a vector direction and apply a cross-product with the instantaneous joint axis from bone k. The second derivative will add a cross-product with the instantaneous joint axis from bone h. Hence, we may now immediately write up a world-coordinate system equation for computing $\mathbf{K}_{ij}$.

Let the instantaneous joint axis of bone k be given by

$$\mathbf{v} = \begin{cases} \mathbf{k} & \text{if } \theta_i = \alpha_k \\ \mathbf{R}_Z(\alpha_k)\mathbf{j} & \text{if } \theta_i = \beta_k \\ \mathbf{R}_Z(\alpha_k)\mathbf{R}_Y(\beta_k)\mathbf{k} & \text{if } \theta_i = \gamma_k \\ \mathbf{0} & \text{otherwise} \end{cases} \tag{19}$$

and similarly for bone h by

$$\mathbf{w} = \begin{cases} \mathbf{k} & \text{if } \theta_j = \alpha_h \\ \mathbf{R}_Z(\alpha_h)\mathbf{j} & \text{if } \theta_j = \beta_h \\ \mathbf{R}_Z(\alpha_h)\mathbf{R}_Y(\beta_h)\mathbf{k} & \text{if } \theta_j = \gamma_h \\ \mathbf{0} & \text{otherwise} \end{cases}. \tag{20}$$

Finally, we have

$$\mathbf{K}_{ij} = \mathbf{w}_{WCS} \times \mathbf{v}_{WCS} \times \Delta \mathbf{p}_{WCS} \tag{21a}$$

$$= \mathbf{w}_{WCS} \times \mathbf{J}_i\,, \tag{21b}$$

where $\mathbf{v}_{WCS} = \mathbf{R}_{WCS}^{k}\mathbf{v}$ and $\mathbf{w}_{WCS} = \mathbf{R}_{WCS}^{h}\mathbf{w}$ are the instantaneous joint axes transformed in the coordinates of the world coordinate system.

3.3 Moving Root Bone

For character animation it may be desirable to have support for a moving root bone. In principle one can make the character walk and jump simply by controlling the end-effectors. Here we outline how to extend the joint angle parameter vector to include the translational parameters of the root bone. First we extend the joint parameter vector θ with $\mathbf{t}_o = \begin{bmatrix} t_x & t_y & t_z \end{bmatrix}^T$ to give us the extended version θ'

$$\theta' \equiv \begin{bmatrix} t_x & t_y & t_z & \theta^T \end{bmatrix}^T . \tag{22}$$

The root bone has the nice property that the bone vector lives in the world coordinate system and will be independent of any rotational degree of freedom in the system. Hence, the first three columns of the IK Jacobian will be

$$\frac{\partial \mathbf{F}(\theta')}{\partial \mathbf{t}_x} = \mathbf{i}\,, \tag{23a}$$

$$\frac{\partial \mathbf{F}(\theta')}{\partial \mathbf{t}_y} = \mathbf{j}\,, \tag{23b}$$

$$\frac{\partial \mathbf{F}(\theta')}{\partial \mathbf{t}_z} = \mathbf{k}\,. \tag{23c}$$

Further, we trivially have from above equations that

$$\mathbf{K}_{ij} = \mathbf{0} \quad \text{if } i \in \{0,1,2\} \text{ or } j \in \{0,1,2\}\,. \tag{24}$$

Hence, we can add the extension trivially to the unmodified $\mathbf{J}$ and $\mathbf{H}$ to get the root-translation extensions $\mathbf{J}'$ and $\mathbf{H}'$, by letting $\mathbf{I}_{3\times3} \equiv \begin{bmatrix}\mathbf{i} & \mathbf{j} & \mathbf{k}\end{bmatrix}$ and define

$$\mathbf{J}' \equiv \begin{bmatrix}\mathbf{I}_{3\times3} & \mathbf{J}\end{bmatrix}\,, \tag{25a}$$

$$\mathbf{H}' \equiv \begin{bmatrix}\mathbf{I}_{3\times3} & \mathbf{0} \\ \mathbf{0} & \mathbf{H}\end{bmatrix}\,. \tag{25b}$$

4 Pseudo Code

In this section we provide pseudo code for initialization and updating of the inverse kinematics (IK) data structures, as well as algorithms for building the Jacobian and Hessian matrices.

Serial chains are easily created by traversal over the bones as shown in Algorithm 1. This should be run once at initialization, or whenever the tree-structure changes its connectivity or becomes re-rooted.

A recursive traversal can be used to perform forward kinematics and compute the world orientations and positions of all bones, as shown in Algorithm 2. It should be invoked prior to any Jacobian or Hessian computation. The Jacobian computation is outlined in Algorithm 3, and the Hessian computation in Algorithm 4.

Performance could be improved by realizing a parallelized implementation of Algorithm 3 and Algorithm 4.

```
   Data: $\mathcal{S}$: All bones in a tree IK structure
   Result: $\mathcal{C}$: Set of all IK chains of the IK structure
1  $\mathcal{L} \leftarrow$ empty set of bones;
2  foreach $\mathcal{B} \in \mathcal{S}$ do
3      if $\mathcal{B}$ has no children then
4          add $B$ to $\mathcal{L}$;
5      end
6  end
7  $\mathcal{C} \leftarrow$ empty set of chains;
8  foreach $\mathcal{B} \in \mathcal{L}$ do
9      $H \leftarrow$ empty chain;
10     while $\mathcal{B}$ exist do
11         add $\mathcal{B}$ to front of $H$;
12         $\mathcal{B} \leftarrow$ Parent bone of $\mathcal{B}$;
13     end
14     add $H$ to $\mathcal{C}$;
15 end
```

Algorithm 1: MakeChains: Chains of a tree-branched kinematic structure are generated by iterating over all bones to identify end-effector bones and then followed by back-traversal from each end-effector to the single root bone of the tree structure. Running time is $\mathcal{O}(N + L\,H)$ where N is number of bones in structure, $L \ll N$ is the number of leaves/chains, and $H < N$ is the height of the tree-structure. Hence, we have $\mathcal{O}(N)$.

```
   Data: $\mathcal{B}_k$: Current bone
1  $(\alpha, \beta, \gamma) \leftarrow$ current Euler angles of $\mathcal{B}_k$;
2  $\mathbf{R}_k \leftarrow \mathbf{R}_Z(\alpha)\,\mathbf{R}_Y(\beta)\,\mathbf{R}_Z(\gamma)$;
3  $\mathbf{t}_k \leftarrow$ local position of $\mathcal{B}_p$;
4  $\mathbf{R}_p \leftarrow$ identity;
5  $\mathbf{t}_p \leftarrow \mathbf{0}$;
6  if Parent bone of $\mathcal{B}_k$ exist then
7      $\mathcal{B}_p \leftarrow$ Parent bone of $\mathcal{B}_k$;
8      $\mathbf{R}_p \leftarrow$ world orientation of $\mathcal{B}_p$;
9      $\mathbf{t}_p \leftarrow$ world position of $\mathcal{B}_p$;
10 end
11 world orientation of $\mathcal{B}_k \leftarrow \mathbf{R}_p\,\mathbf{R}_k$;
12 world position of $\mathcal{B}_k \leftarrow \mathbf{t}_p + \mathbf{R}_p\,\mathbf{t}_k$;
13 foreach $\mathcal{B}_c \in$ children of $\mathcal{B}_k$ do
14     update($\mathcal{B}_c$);
15 end
```

Algorithm 2: ForwardKinematicsUpdate: Recursive forward sweep through a hierarchical kinematics structure quickly updates all bones with their current orientation $\mathbf{R}$ and position $\mathbf{t}$ of joint frames in the WCS. Running time is $\mathcal{O}(N)$, where N is number of bones in the structure.

5 Other Euler Conventions

Although we used the ZYZ Euler convection in our derivations our theory is general and supports any Euler angle convention. For instance imagine we used ZYX then we make the modifications

$$\mathbf{R}_k \leftarrow \mathbf{R}_Z(\alpha)\,\mathbf{R}_Y(\beta)\,\mathbf{R}_X(\gamma) \tag{26}$$

to line 2 in Algorithm 2. Further the instantaneous joint axes in lines 15-17 of Algorithm 3 and lines 12-14 in Algorithm 4 will be modified accordingly

$$\mathbf{u} \leftarrow \mathbf{R}_p\mathbf{k}\,, \tag{27a}$$

$$\mathbf{v} \leftarrow \mathbf{R}_p\mathbf{R}_Z(\alpha)\,\mathbf{j}\,, \tag{27b}$$

$$\mathbf{w} \leftarrow \mathbf{R}_p\mathbf{R}_Z(\alpha)\,\mathbf{R}_Y(\beta)\,\mathbf{i}\,. \tag{27c}$$

To make this even more general, we may denote any Euler convention by the ordered triplet ABC. Each letter

```
   Data: $\mathcal{C}$: set of chains, $\mathcal{S}$: All bones in a tree IK structure
   Result: $\mathbf{J}$: The IK Jacobian
 1 ForwardKinematicsUpdate(root bone of $\mathcal{S}$);
 2 $\mathcal{C} \leftarrow$ MakeChains($\mathcal{S}$);
 3 $E \leftarrow$ number of chains;
 4 $N \leftarrow$ number of bones;
 5 $\mathbf{J} \leftarrow$ zero matrix of $3\,E \times 3\,N$;
 6 foreach $H_i \in \mathcal{C}$ do
 7     $\mathbf{e} \leftarrow$ world position of end-effector bone in $H_i$;
 8     foreach $\mathcal{B}_k \in \mathcal{H}$ do
 9         $\mathbf{R}_p \leftarrow$ identity;
10         if Parent bone of $\mathcal{B}_k$ exist then
11             $\mathbf{R}_p \leftarrow$ world orientation of parent bone of $\mathcal{B}_k$;
12         end
13         $(\alpha, \beta, \gamma) \leftarrow$ current Euler angles of $\mathcal{B}_k$;
14         $\Delta\mathbf{p} \leftarrow \mathbf{e} -$ world position of $\mathcal{B}_k$;
15         $\mathbf{u} \leftarrow \mathbf{R}_p\,\mathbf{k}$;
16         $\mathbf{v} \leftarrow \mathbf{R}_p\,\mathbf{R}_Z(\alpha)\,\mathbf{j}$;
17         $\mathbf{w} \leftarrow \mathbf{R}_p\,\mathbf{R}_Z(\alpha)\,\mathbf{R}_Y(\beta)\,\mathbf{k}$;
18         $\mathbf{J}_{i,3\,k} \leftarrow \mathbf{u} \times \Delta\mathbf{p}$;
19         $\mathbf{J}_{i,3\,k+1} \leftarrow \mathbf{v} \times \Delta\mathbf{p}$;
20         $\mathbf{J}_{i,3\,k+3} \leftarrow \mathbf{w} \times \Delta\mathbf{p}$;
21     end
22 end
```

Algorithm 3: ComputeJacobian: Building the IK Jacobian uses blocked indexing into $\mathbf{J}$, which consists of $\mathbf{R}^3$ blocks. The running time is $\mathcal{O}(N\,L)$, where $L < N$ is the number of chains in the structure. Tool-vectors are ignored in this algorithm, but line 7 can be modified to include non-zero tool-vectors if needed.

```
   Data: $\mathcal{C}$: set of chains, $\mathbf{J}$: Jacobian matrix
   Result: $\mathbf{H}$: The IK Hessian matrix
 1 $\mathbf{H} \leftarrow$ symmetric matrix $\mathbf{J}^T\mathbf{J}$;
 2 foreach $H_c \in \mathcal{C}$ do
 3     $\mathbf{e} \leftarrow$ world position of end-effector bone in $H_c$;
 4     $\mathbf{r} \leftarrow$ goal of $H_c$ - $\mathbf{e}$ ;
 5     foreach $\mathcal{B}_k \in \mathcal{H}$ do
 6         foreach $\mathcal{B}_h \in \mathcal{H}$ with $h \leq k$ do
 7             $\mathbf{R}_p \leftarrow$ identity;
 8             if Parent bone of $\mathcal{B}_h$ exist then
 9                 $\mathbf{R}_p \leftarrow$ world orientation of parent of $\mathcal{B}_h$;
10             end
11             $(\alpha, \beta, \gamma) \leftarrow$ current Euler angles of $\mathcal{B}_h$;
12             $\mathbf{u} \leftarrow \mathbf{R}_p\,\mathbf{k}$;
13             $\mathbf{v} \leftarrow \mathbf{R}_p\,\mathbf{R}_Z(\alpha)\,\mathbf{j}$;
14             $\mathbf{w} \leftarrow \mathbf{R}_p\,\mathbf{R}_Z(\alpha)\,\mathbf{R}_Y(\beta)\,\mathbf{k}$;
15             foreach $i \in (3\,k, 3\,k+2)$ do
16                 $\mathbf{H}_{i,3\,h} \leftarrow \mathbf{H}_{i,3\,h} - (\mathbf{u} \times \mathbf{J}_i)^T\,\mathbf{r}$;
17                 $\mathbf{H}_{i,3\,h+1} \leftarrow \mathbf{H}_{i,3\,h+1} - (\mathbf{v} \times \mathbf{J}_i)^T\,\mathbf{r}$;
18                 $\mathbf{H}_{i,3\,h+2} \leftarrow \mathbf{H}_{i,3\,h+2} - (\mathbf{w} \times \mathbf{J}_i)^T\,\mathbf{r}$;
19             end
20         end
21     end
22 end
```

Algorithm 4: ComputeHessian: The numerical method for computing the Hessian only fills in the upper triangular part of the Hessian $\mathbf{H}$-matrix. Notice that the tool vectors are assumed to be the zero vectors in line 3. The running time is $\mathcal{O}(N^2L + L^2)$, or $\mathcal{O}(N^2)$ overall.

referes to a rotation around some coordinate axis. We label the respective axes **a**, **b**, **c**. Now we have,

$$\mathbf{R}_k \leftarrow \mathbf{R}_A(\alpha)\,\mathbf{R}_B(\beta)\,\mathbf{R}_C(\gamma) \tag{28}$$

and

$$\mathbf{u} \leftarrow \mathbf{R}_p\mathbf{a}\,, \tag{29a}$$
$$\mathbf{v} \leftarrow \mathbf{R}_p\mathbf{R}_A(\alpha)\,\mathbf{b}\,, \tag{29b}$$
$$\mathbf{w} \leftarrow \mathbf{R}_p\mathbf{R}_A(\alpha)\,\mathbf{R}_B(\beta)\,\mathbf{c}\,. \tag{29c}$$

References

[1] F. Sebastin Grassia. Practical parameterization of rotations using the exponential map. *J. Graph. Tools*, 3(3):29–48, March 1998.