One-shot Adaptation of Humanoid Whole-body Motion with Walking Priors

Supplementary Material

Collision detection plays a pivotal role in generating physically plausible poses for articulated structures, where self-intersections may occur due to intricate joint arrangements and complex motion patterns. The process entails representing the skeleton as a hierarchical tree of joints and bones, transforming local joint rotations to global coordinates through forward kinematics, and assessing collisions via geometric primitives like spheres and capsules. Intersections are quantified using penetration energies, which are subsequently minimized through optimization to yield collision-free and physically consistent configurations. This framework draws on computational geometry and optimization techniques, ensuring differentiability for an efficient gradient-based solution.

1. Forward Kinematics: Transforming Local Joint Rotations to Global Coordinates

The first step of collision detection is the computation of global joint positions from local joint rotations. Consider a skeleton with J joints, governed by a parent array $\mathbf{p} \in \mathbb{Z}^J$, where $p_j = -1$ denotes the root joint, and otherwise indicates the parent of joint j. Each joint j is parameterized by a local rotation quaternion $\mathbf{q}_j \in \mathbb{S}^3$ $\mathbf{q}_j \in \mathbb{H}_1 \subset \mathbb{R}^4$ represents the rotation of the j-th joint. Here, $\mathbb{H}_1 = \{\mathbf{q} \in \mathbb{R}^4 : \|\mathbf{q}\|_2 = 1\}$ denotes the unit quaternion group and a local translation $\mathbf{t}_j \in \mathbb{R}^3$ specified by the kinematic tree of the given robot.

Global transformations are derived recursively. The local transformation matrix is $\mathbf{T}_j^{\text{local}} = \begin{pmatrix} \mathbf{R}(\mathbf{q}_j) & \mathbf{t}_j \\ \mathbf{0}^\top & 1 \end{pmatrix}$, with $\mathbf{R}(\mathbf{q}_j)$ the rotation matrix induced by $\mathbf{q}_j$. The global matrix is $\mathbf{T}_j = \mathbf{T}_{p_j} \cdot \mathbf{T}_j^{\text{local}}$ for $p_j \neq -1$, and $\mathbf{T}_j = \mathbf{T}_j^{\text{local}}$ at the root. The global position $\mathbf{x}_j \in \mathbb{R}^3$ extracts the translation from $\mathbf{T}_j$. This yields the set $\{\mathbf{x}_j\}_{j=1}^J$, facilitating subsequent geometric evaluations.

2. Geometric Primitives for Collision Modeling

As shown in Fig. 1, joints are approximated as sphere with radii r_j, scaled by a factor ρ (typically $\rho = 0.04$) proportional to bone lengths: $r_j = \rho\|\mathbf{t}_j\|$ for the root, or $r_j = \rho\|\mathbf{x}_j - \mathbf{x}_{p_j}\|$ otherwise, where p_j represents the parent joint of the jth joint.

Bones, linking parent-child joints, are modeled as capsules—cylinders capped by hemispheres—with radius equal to the parent's sphere radius. A bone from joint i to j comprises the line segment $[\mathbf{p}, \mathbf{q}]$ where $\mathbf{p} = \mathbf{x}_i$, $\mathbf{q} = \mathbf{x}_j$, and radius r_i.

3. Distance Computation for Line Segments

Capsule intersections necessitate the minimum distance between bone line segments, a core problem in computational geometry with roots in robotics and graphics. Here, we provide a way to compute the minimum distance between bone line segments [1] (*i.e.*, the purple lines in Fig. 1), not the bone capsule. For segments $[\mathbf{p}_1, \mathbf{q}_1]$ and $[\mathbf{p}_2, \mathbf{q}_2]$, the distance $d = \min\|\mathbf{x} - \mathbf{y}\|$ is sought, with $\mathbf{x}, \mathbf{y}$ are points on respective segments.

Parametrize: $\mathbf{x} = \mathbf{p}_1 + s\mathbf{u}$, $\mathbf{y} = \mathbf{p}_2 + t\mathbf{v}$, $s, t \in [0, 1]$, $\mathbf{u} = \mathbf{q}_1 - \mathbf{p}_1$, $\mathbf{v} = \mathbf{q}_2 - \mathbf{p}_2$. Minimize $f(s,t) = \|\mathbf{x} - \mathbf{y}\| = \|\mathbf{w}_0 + s\mathbf{u} - t\mathbf{v}\|^2$, $\mathbf{w}_0 = \mathbf{p}_1 - \mathbf{p}_2$. Geometrically, configurations include intersecting (distance zero), skew (unique common perpendicular), parallel (constant separation), and degenerate (collinear) cases. Closest points categorize as interior-interior, interior-vertex, or vertex-vertex.

Expanding $f(s,t)$ (using the dot product):

$$f(s,t) = (\mathbf{w}_0 + s\mathbf{u} - t\mathbf{v}) \cdot (\mathbf{w}_0 + s\mathbf{u} - t\mathbf{v})$$
$$= |\mathbf{w}_0|^2 + s^2|\mathbf{u}|^2 + t^2|\mathbf{v}|^2 + 2s\mathbf{w}_0 \cdot \mathbf{u} - 2t\mathbf{w}_0 \cdot \mathbf{v} - 2st\mathbf{u} \cdot \mathbf{v} \ . \tag{1}$$

This is a quadratic function in s and t, and since the Hessian (second derivatives) is positive semi-definite (as we'll see via the determinant), it has a global minimum. **Deriving the linear system via critical points.** To find the minimum, compute the partial derivatives of $f(s,t)$ and set them to zero (stationarity conditions from calculus):

$$\frac{\partial f}{\partial s} = 2s|\mathbf{u}|^2 + 2\mathbf{w}_0 \cdot \mathbf{u} - 2t\mathbf{u} \cdot \mathbf{v} = 0 \ , \tag{2}$$

$$\frac{\partial f}{\partial t} = 2t|\mathbf{v}|^2 - 2\mathbf{w}_0 \cdot \mathbf{v} - 2s\mathbf{u} \cdot \mathbf{v} = 0 \ .$$

Dividing both by 2 and rearranging:

$$s|\mathbf{u}|^2 - t(\mathbf{u} \cdot \mathbf{v}) = -(\mathbf{w}_0 \cdot \mathbf{u}) \ , \tag{3}$$

$$-s(\mathbf{u} \cdot \mathbf{v}) + t|\mathbf{v}|^2 = \mathbf{w}_0 \cdot \mathbf{v} \ . \tag{4}$$

Define the scalars:
- $a = \mathbf{u} \cdot \mathbf{u} = |\mathbf{u}|^2$,
- $b = \mathbf{u} \cdot \mathbf{v}$,
- $c = \mathbf{v} \cdot \mathbf{v} = |\mathbf{v}|^2$,
- $d = \mathbf{u} \cdot \mathbf{w}_0$,
- $e = \mathbf{v} \cdot \mathbf{w}_0$.

This gives the linear system in matrix form:

$$\begin{pmatrix} a & -b \\ b & -c \end{pmatrix} \begin{pmatrix} s \\ t \end{pmatrix} \begin{pmatrix} -d \\ -e \end{pmatrix} . \tag{5}$$

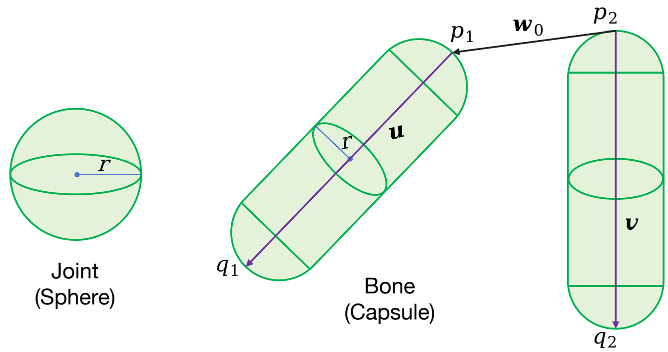

Figure 1. Illustration of spheres for joints, capsules for bones, and line segment distance between two bones.

Geometrically, these equations enforce that the vector connecting the closest points, $\mathbf{x}(s) - \mathbf{y}(t)$, is perpendicular to both direction vectors $\mathbf{u}$ and $\mathbf{v}$. This is because:

$$\frac{\partial f}{\partial s} = 2(\mathbf{x}(s) - \mathbf{y}(t)) \cdot \mathbf{u} = 0 \implies (\mathbf{x}(s) - \mathbf{y}(t)) \perp \mathbf{u} \ , \tag{6}$$

$$\frac{\partial f}{\partial t} = -2(\mathbf{x}(s) - \mathbf{y}(t)) \cdot \mathbf{v} = 0 \implies (\mathbf{x}(s) - \mathbf{y}(t)) \perp \mathbf{v} \ . \tag{7}$$

For non-parallel lines, this common perpendicular is unique, leading to a unique solution (s, t).

The determinant D and its properties. The determinant of the coefficient matrix is:

$$\det = a(-c) - (-b)b = -ac + b^2 = -(ac - b^2) \ . \tag{8}$$

By the Cauchy-Schwarz inequality:

$$|\mathbf{u} \cdot \mathbf{v}|^2 \leq |\mathbf{u}|^2 |\mathbf{v}|^2 \implies b^2 \leq ac \implies D = ac - b^2 \geq 0 \ . \tag{9}$$

- Strict inequality $D > 0$: Holds when $\mathbf{u}$ and $\mathbf{v}$ are not parallel (*i.e.*, the lines are skew or intersecting but not parallel). The system has a unique solution.
- Equality $D = 0$: Occurs when $\mathbf{u}$ and $\mathbf{v}$ are parallel ($\mathbf{u} = k\mathbf{v}$ for some scalar $k \neq 0$). The lines are parallel, and the system may be consistent (infinite solutions if the separation is constant) or inconsistent (no solution if non-coplanar, but in distance minimization, we handle via projections). In practice, for $D \approx 0$, we treat it as a special case to avoid division by zero.

The non-negativity property of D ensures the quadratic form is convex, guaranteeing a minimum (or saddle in degenerate cases).

Solving the system for $D > 0$. For the non-parallel case ($D > 0$), solve using Cramer's rule or matrix inversion. The solutions are:

$$\begin{aligned} s &= \frac{\det \begin{bmatrix} -d & -b \\ -e & -c \end{bmatrix}}{-D} = \frac{(-d)(-c) - (-b)(-e)}{-D} \\ &= \frac{dc - be}{-D} = \frac{be - cd}{D} \ , \\ t &= \frac{\det \begin{bmatrix} a & -d \\ b & -e \end{bmatrix}}{-D} = \frac{a(-e) - (-d)b}{-D} \\ &= \frac{-ae + db}{-D} = \frac{bd - ae}{D} = \frac{ae - bd}{D} \ . \end{aligned} \tag{10}$$

For finite segments, we clamp $s, t \in [0, 1]$. Then, we re-adjust these two values, *i.e.*, fix t, solve $s = (d + tb)/a$, re-clamp into $[0, 1]$ again. After getting s and t, we plug them into Eq. 1 to compute the minimum distance between two line segments.

Handling the parallel case ($D = 0$). When $D \approx 0$, *i.e.*, $D < \epsilon$ where $\epsilon \approx 10^{-9}$, the lines are parallel, and the shortest distance is the perpendicular separation between the lines (constant along their length). There is no unique (s, t); instead:

- Project $\mathbf{w}_0$ onto the common direction (normalized $\mathbf{v}/|\mathbf{v}|$).
- Choose an arbitrary s (*e.g.*, $s = 0$), then solve for $t = e/c$ (clamped if finite), or compare endpoint projections.
- The minimum distance is $|\mathbf{w}_0 - \text{proj}_{\mathbf{v}} \mathbf{w}_0|$. This ensures numerical stability and geometric correctness..

For finite segments, the distance is computed in $O(1)$ time, robust for real-time applications.

4. Penetration Energies: Quantifying Collisions

Energies penalize intersections over non-adjacent pairs, avoiding kinematic artifacts.

For (joint) spheres, the energy E_{sphere} is the sum of squared penetrations:

$$E_{\text{sphere}} = \sum_{i<j,(i,j)\in\mathcal{V}} \left(\max(0, r_i + r_j - |\mathbf{x}_i - \mathbf{x}_j|)\right)^2 \quad , \tag{11}$$

where $\mathcal{V}$ excludes adjacent or grandparent-grandchild pairs (*i.e.*, $j \neq p_i$, $i \neq p_j$, *etc.*).

For (bone) capsules, gather bones as pairs (a_m, b_m) for $m = 1, \ldots, M$ (typically $M = J - 1$). The energy E_{capsule} is defined as:

$$E_{\text{capsule}} = \sum_{i<j,(i,j)\in\mathcal{V}'} \left(\max(0, r_i + r_j - d_{ij})\right)^2 \quad , \tag{12}$$

where d_{ij} is the line segment distance between bones i and j, *i.e.*, the minimum value of Eq. 1 as discuss above, and $\mathcal{V}'$ excludes bones sharing joints.

The total energy is $E = E_{\text{sphere}} + \lambda E_{\text{capsule}}$ where we set $\lambda = 1$ in our settings.

5. Pose Optimization for Collision Resolution

The total energy E is differentiable w.r.t. the joint rotations. Thus, we minimize E over $\{\mathbf{q}_j\}$ on SO(3)^J using Riemannian gradient descent (learning rate $\eta \approx 0.05$, *e.g.*, 120 steps). Project gradient: $\mathbf{g}_{\text{proj}} = \nabla_{\mathbf{q}} E - (\mathbf{q} \cdot \nabla_{\mathbf{q}} E)\mathbf{q}$. Update $\mathbf{q} \leftarrow \mathbf{q} - \eta \mathbf{g}_{\text{proj}}$, and then normalize, *i.e.*, send back to SO(3)^J. This process is sketched in Algorithm 1 in the main text. This iteratively mitigates penetrations, producing viable poses for animation and simulation.

References

[1] Christer Ericson. *Real-time collision detection*. CRC Press, 2004. 1