Highly adaptable real-time Kinematic Chain-Based Motion Retargeting for Virtual Characters

Mick Körner Thomas Kronfeld Guido Brunnett

GI VR/AR Workshop 2025, TU Chemnitz

16. September 2025

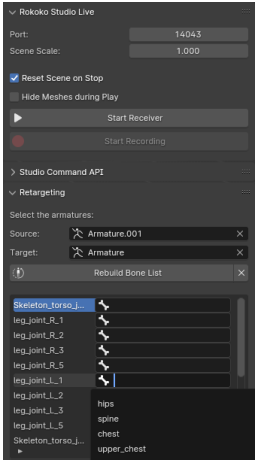
Introduction

- Motion Retargeting: Transferring motion between virtual characters.
- Style should be kept intact.
- Crucial for many applications: VR, games, film animation.
- The Problem:

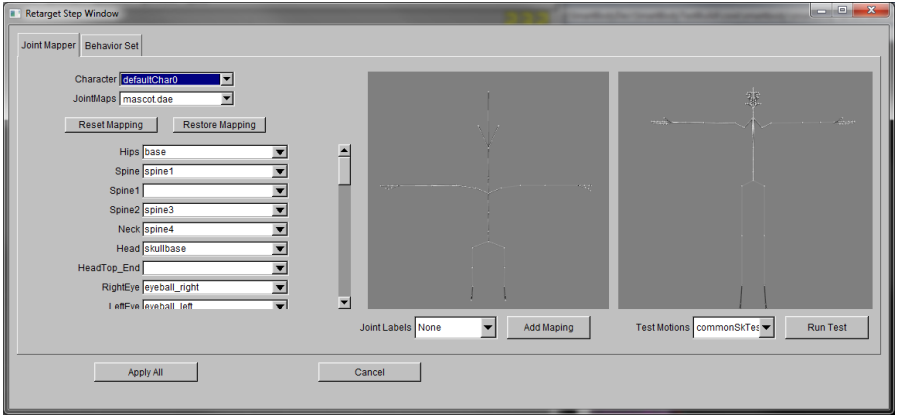
Introduction

- Motion Retargeting: Transferring motion between virtual characters.
- Style should be kept intact.
- Crucial for many applications: VR, games, film animation.
- The Problem:
  - Closed-Source Solutions:
    - Rely on standardized Skeletons.
    - No insight on Functionality.
    - Harder to Adopt into Projects.
  - Open-Source Solutions:
    - Require similar skeletons.
    - Require substantial user input.
    - Many rely on outdated approaches.
  - Retargeting quality often subjective depending on usecase.

Introduction



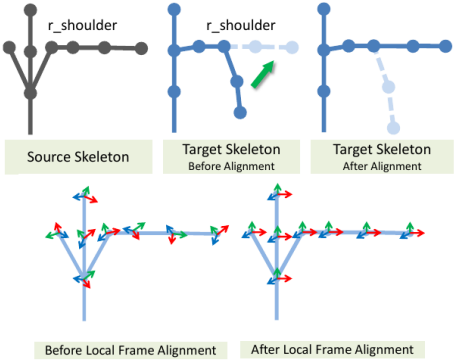
rokoko blender



smartbody

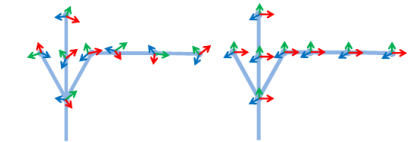
Related Work Excerpt

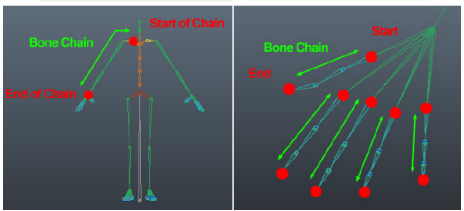
- Problem motion transfer not trivial due to BCS (Base Coordinate System) differences:
- Intermediate Skeleton:
  - Ming-Kai Hsieh et al. [MBM05] precompute correction.
  - Align BCS of Skeletons.
  - Tang et al. [Ta12] adjust Skeletons using adjustment Matrix.
  - Joint correspondence manually defined.
- Many methods use Inverse Kinematics (IK) for artifact cleanup.



Related Work Excerpt

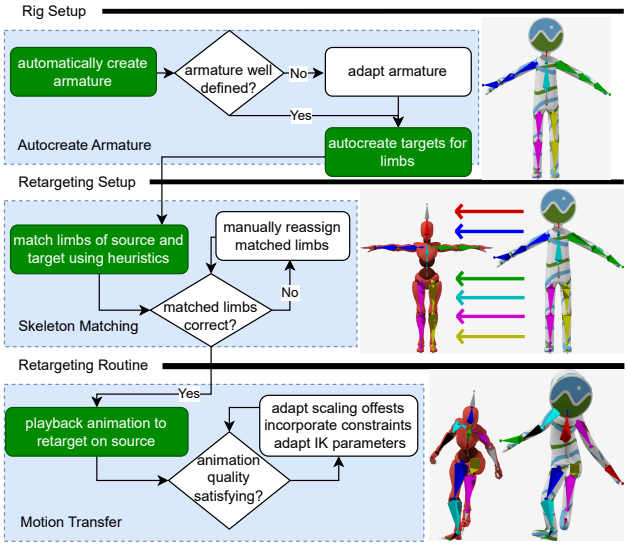
- Problem motion transfer not trivial due to BCS (Base Coordinate System) differences:
- Intermediate Skeleton:
  - Ming-Kai Hsieh et al. [MBM05] precompute correction.
  - Align BCS of Skeletons.
  - Tang et al. [Ta12] adjust Skeletons using adjustment Matrix.
  - Joint correspondence manually defined.
- Many methods use Inverse Kinematics (IK) for artifact cleanup.
- Limb abstraction:
  - Used for simplified matching or IK.
  - Du Sel et al. [DCR15] define Chains on Skeleton.
  - Abdul-Massih et al. [AYB17] propose Groups of Body Parts.
  - Allows Retargeting between largely different Skeletal Structures.





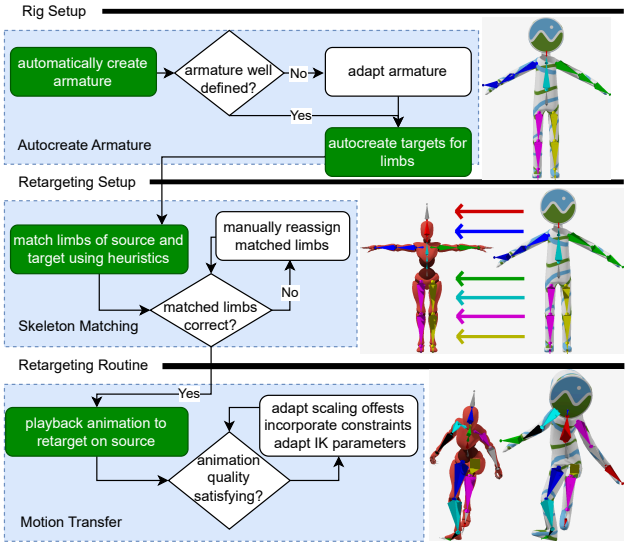
Proposed Framework

- A new adaptable kinematic chain-based Retargeting approach.



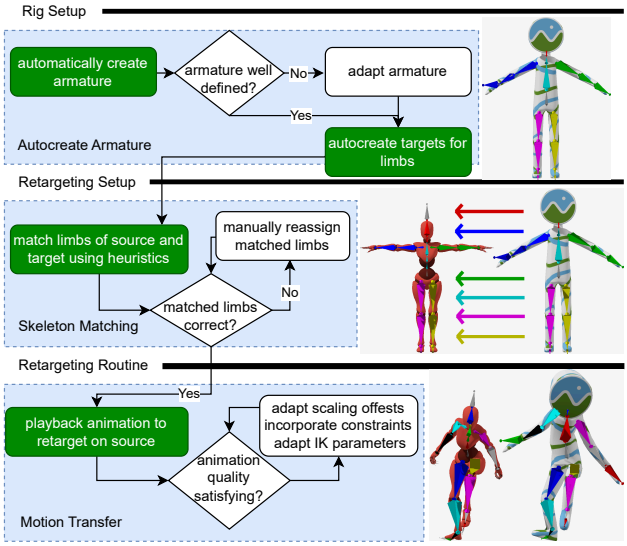
Proposed Framework

- A new adaptable kinematic chain-based Retargeting approach.
- **Key Contributions:**
  1. Automated Armature Setup
  2. Heuristic Skeleton Matching
  3. Adaptive Motion Retargeting
  4. Advanced Kinematic Control
  5. Dynamic Kinematic Chain Scaling



Proposed Framework

- A new adaptable kinematic chain-based Retargeting approach.
- **Key Contributions:**
  1. Automated Armature Setup
  2. Heuristic Skeleton Matching
  3. Adaptive Motion Retargeting
  4. Advanced Kinematic Control
  5. Dynamic Kinematic Chain Scaling
- Contents:
  1. Armature creation & matching.
  2. Joint angle Imitation based on matched Chains.
  3. IK cleanup and target scaling.

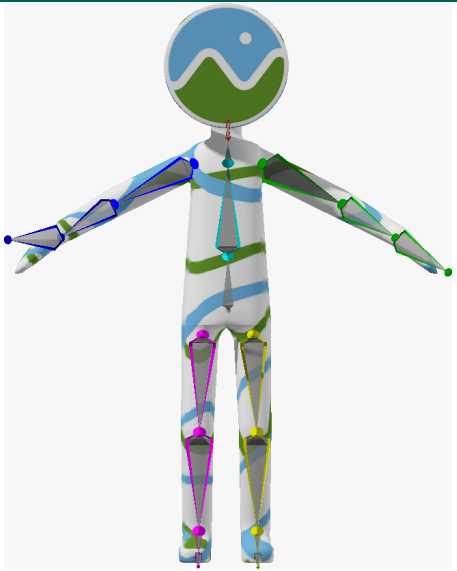


Automated Armature creation

- Create Chains for armature automatically:

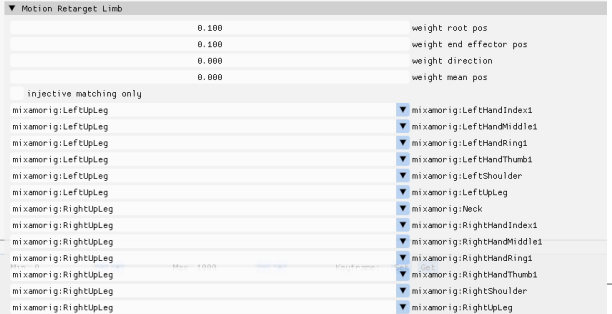
1. DFS starting from root joint.
2. Name Chain same as starting Joint.
3. On ≥ 2 child Joints:
   - End current Chain definition.
   - Start new definition for each Child.

- Opt. ignore Control Bones (no skinning weights).
- Can rename and adapt chains manually.



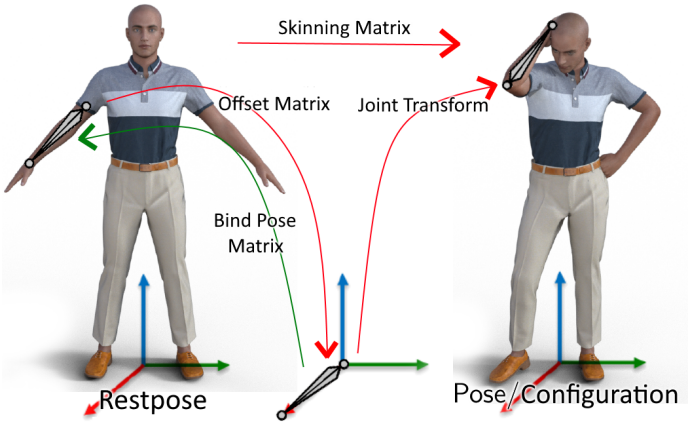
Armature Matching

- Joint Correspondence Problem simplified through matching Chains.
- Calculate closeness rating using distance between root joints, end-effector joints, direction and median joint position.
- Higher score indicates closer relation.
- Interactively adapt by weighting parameters.
- Hungarian algorithm used for injective mapping.



Joint angle Imitation: Recap of Skeletal Skinning

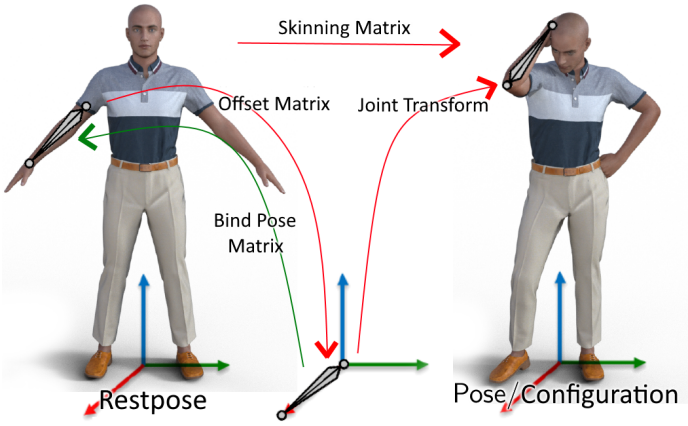
- Skinning Matrices used for Blend Skinning (LBS etc.).
- Used by various 3D File formats.
- **Bind Pose Matrix**:
  - transforms local Joint BCS (Base Coordinate System) to Rest Pose configuration.
- **Offset Matrix**:
  - transforms Joint from Rest Pose to BCS.
- **Joint Transform**: $M_{parent} \cdot M_{local}$



Joint angle Imitation: Recap of Skeletal Skinning

- Skinning Matrices used for Blend Skinning (LBS etc.).
- Used by various 3D File formats.
- **Bind Pose Matrix**:
  - transforms local Joint BCS (Base Coordinate System) to Rest Pose configuration.
- **Offset Matrix**:
  - transforms Joint from Rest Pose to BCS.
- **Joint Transform**: $M_{parent} \cdot M_{local}$
- **Skinning Matrix**:

$$SM = M_{parent} \cdot M_{local} \cdot OM$$



Single Joint angle Imitation

- Intuitive derivation of the formula:

$$\overset{s}{SM} = \overset{s}{M}_{parent} \cdot \boxed{\overset{s}{M}_{local}} \cdot \overset{s}{OM}$$

Single Joint angle Imitation

- Intuitive derivation of the formula:

$$\overset{s}{SM} = \overset{s}{M}_{parent} \cdot \boxed{\overset{s}{M}_{local}} \cdot \overset{s}{OM}$$

$$\overset{t}{M}_{newLocal} = \overset{t}{M}_{parent}^{-1} \cdot \overset{s}{M}_{parent} \cdot \overset{s}{M}_{local} \cdot \overset{s}{OM} \cdot \overset{t}{OM}^{-1} = \overset{t}{M}_{parent}^{-1} \cdot \overset{s}{SM} \cdot \overset{t}{OM}^{-1}$$

Single Joint angle Imitation

- Intuitive derivation of the formula:

$$\overset{s}{SM} = \overset{s}{M}_{parent} \cdot \boxed{\overset{s}{M}_{local}} \cdot \overset{s}{OM}$$

$$\overset{t}{M}_{newLocal} = \overset{t}{M}_{parent}^{-1} \cdot \overset{s}{M}_{parent} \cdot \overset{s}{M}_{local} \cdot \overset{s}{OM} \cdot \overset{t}{OM}^{-1} = \overset{t}{M}_{parent}^{-1} \cdot \overset{s}{SM} \cdot \overset{t}{OM}^{-1}$$

- Therefore when evaluating a target Skinning Matrix:

$$\overset{t}{SM} = \overset{t}{M}_{parent} \cdot \boxed{\overset{t}{M}_{parent}^{-1} \cdot \overset{s}{SM} \cdot \overset{t}{OM}^{-1}} \cdot \overset{t}{OM} = \overset{s}{SM}$$

- Resulting in restpose relative imitation.

Single Joint angle Imitation

- Intuitive derivation of the formula:

$$\overset{s}{SM} = \overset{s}{M}_{parent} \cdot \boxed{\overset{s}{M}_{local}} \cdot \overset{s}{OM}$$

$$\overset{t}{M}_{newLocal} = \overset{t}{M}_{parent}^{-1} \cdot \overset{s}{M}_{parent} \cdot \overset{s}{M}_{local} \cdot \overset{s}{OM} \cdot \overset{t}{OM}^{-1} = \overset{t}{M}_{parent}^{-1} \cdot \overset{s}{SM} \cdot \overset{t}{OM}^{-1}$$

- Therefore when evaluating a target Skinning Matrix:

$$\overset{t}{SM} = \overset{t}{M}_{parent} \cdot \boxed{\overset{t}{M}_{parent}^{-1} \cdot \overset{s}{SM} \cdot \overset{t}{OM}^{-1}} \cdot \overset{t}{OM} = \overset{s}{SM}$$

- Resulting in restpose relative imitation.
- Extendable to support mapping multiple source to one target joint:

$$\overset{t}{M}_{newLocal} = \overset{t}{M}_{parent}^{-1} \cdot \overset{s}{M}_{parent} \cdot \overset{s}{M}_{local} \cdot (\overset{s_2}{M}_{local} \cdot ... \cdot \overset{s_N}{M}_{local}) \cdot \overset{s}{OM} \cdot \overset{t}{OM}^{-1}$$

Imitating Joint angles

1. The kinematic chain of the target armature covering the current examined joint is identified.

imitate(J,M_{parent})

- Find target Chain C which contains J.

Imitating Joint angles

1. The kinematic chain of the target armature covering the current examined joint is identified.
2. A corresponding matched kinematic chain from the source armature is retrieved.

imitate(J,M_{parent})

- Find target Chain C which contains J.
- if C exists
  - $CS := corr(C)$

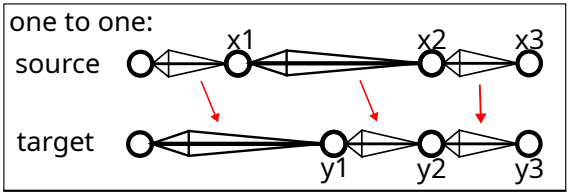
Imitating Joint angles

1. The kinematic chain of the target armature covering the current examined joint is identified.
2. A corresponding matched kinematic chain from the source armature is retrieved.
3. The joint indexing function is then used to determine which joints of the source Character are being imitated.

imitate(J,M_{parent})

- Find target Chain C which contains J.
- if C exists
  - $CS := corr(C)$
  - set $s := jointIndexing(J.id, CS, C)$

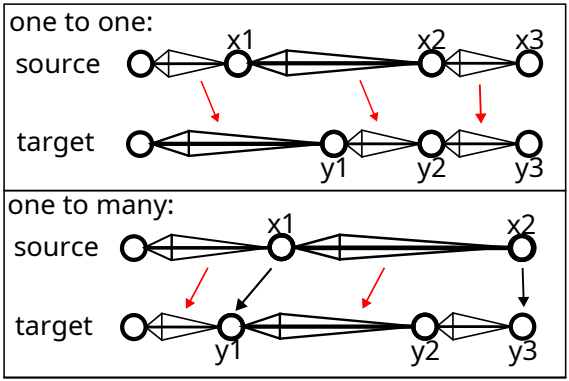
Joint Indexing Function



- Determines which source joints get mapped inside a chain.
- Indexing function choosen freely.
- one-to-one: Map directly.

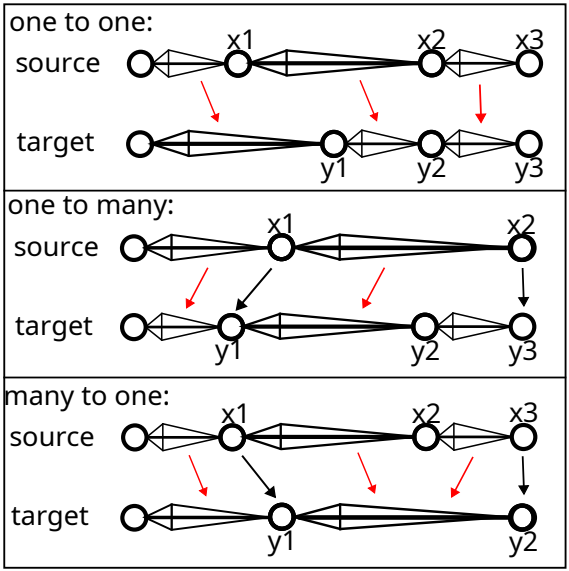
Joint Indexing Function

- Determines which source joints get mapped inside a chain.
- Indexing function choosen freely.
- one-to-one: Map directly.
- Mapping based on normalized cumulative lengths:
  1. Straighten Chains and compute absolute distance to root.
  2. Normalize all lengths $[0, 1]$
  3. Greedy one-to-one map $\forall i, j : min(|x_i - y_j|)$ pairs first.



Joint Indexing Function

- Determines which source joints get mapped inside a chain.
- Indexing function choosen freely.
- one-to-one: Map directly.
- Mapping based on normalized cumulative lengths:
  1. Straighten Chains and compute absolute distance to root.
  2. Normalize all lengths $[0, 1]$
  3. Greedy one-to-one map $\forall i, j : min(|x_i - y_j|)$ pairs first.
  4. Remaining source nodes mapped to previous target node.

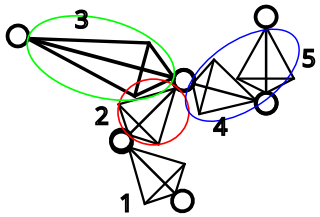


Imitating Joint Angles

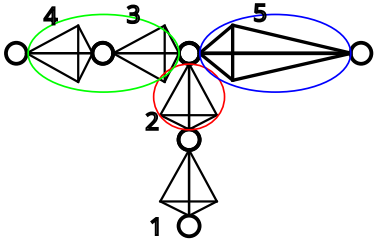
imitate(J,M_{parent})

- Find target Chain C which contains J.
- if C exists
  - set $s := jointIndexing(J.id, corr(C), C)$
- if $s.size > 0$:
  - $\overset{t}{M}_{newLocal} := {M_{parent}}^{-1} \cdot \overset{s}{SM} \cdot {\overset{t}{OM}}^{-1}$
  - $M_{parent} := M_{parent} \cdot \overset{t}{M}_{newLocal}$
- else: (no match)
  - $M_{parent} := M_{parent} \cdot \overset{t}{M}_{local}$
- For $child$ **in** J.children:
  - imitate($child$,M_{parent})

source



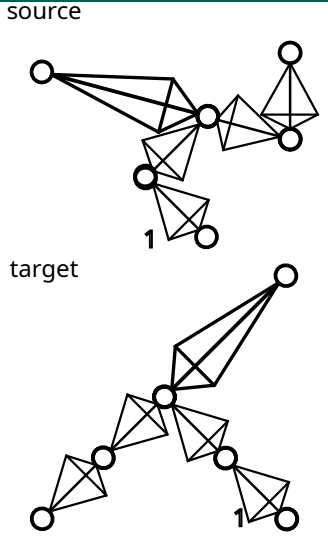
target



Imitating Joint Angles: Step 1

imitate(J,M_{parent})

- Find target Chain C which contains J.
- if C exists
  - set $s := jointIndexing(J.id, corr(C), C)$
- if $s.size > 0$:
  - $\overset{t}{M}_{newLocal} := {M_{parent}}^{-1} \cdot \overset{s}{SM} \cdot {\overset{t}{OM}}^{-1}$
  - $M_{parent} := M_{parent} \cdot \overset{t}{M}_{newLocal}$
- else: (no match)
  - $M_{parent} := M_{parent} \cdot \overset{t}{M}_{local}$
- For $child$ **in** J.children:
  - imitate($child$,M_{parent})

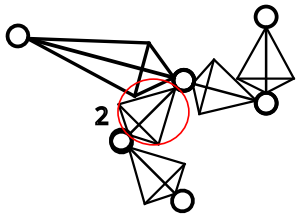


Imitating Joint Angles: Step 2

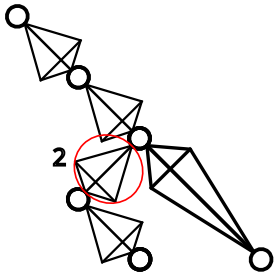
imitate(J,M_{parent})

- Find target Chain C which contains J.
  - if C exists
    - set $s := jointIndexing(J.id, corr(C), C)$
- if $s.size > 0$:
  - $\overset{t}{M}_{newLocal} := {M_{parent}}^{-1} \cdot \overset{s}{SM} \cdot \overset{t}{OM}^{-1}$
  - $M_{parent} := M_{parent} \cdot \overset{t}{M}_{newLocal}$
- else: (no match)
  - $M_{parent} := M_{parent} \cdot \overset{t}{M}_{local}$
- For $child$ **in** J.children:
  - imitate($child$,M_{parent})

source



target

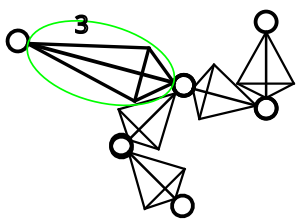


Imitating Joint Angles: Step 3

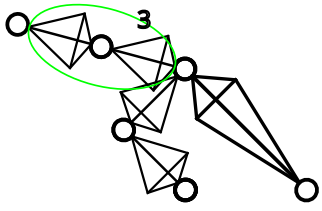
imitate(J,M_{parent})

- Find target Chain C which contains J.
- if C exists
  - set $s := jointIndexing(J.id, corr(C), C)$
  - if $s.size > 0$:
    - $\overset{t}{M}_{newLocal} := {M_{parent}}^{-1} \cdot \overset{s}{SM} \cdot \overset{t}{OM}^{-1}$
    - $M_{parent} := M_{parent} \cdot \overset{t}{M}_{newLocal}$
- else: (no match)
  - $M_{parent} := M_{parent} \cdot \overset{t}{M}_{local}$
- For $child$ **in** J.children:
  - imitate($child$,M_{parent})

source



target

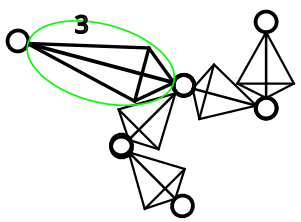


Imitating Joint Angles: Step 4

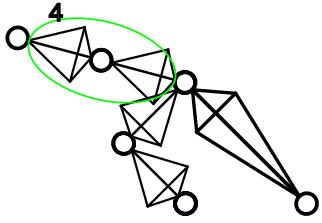
imitate(J,M_{parent})

- Find target Chain C which contains J.
- if C exists
  - set $s := jointIndexing(J.id, corr(C), C)$
- if $s.size > 0$:
  - $\overset{t}{M}_{newLocal} := {M_{parent}}^{-1} \cdot \overset{s}{SM} \cdot {\overset{t}{OM}}^{-1}$
  - $M_{parent} := M_{parent} \cdot \overset{t}{M}_{newLocal}$
- else: (no match)
  - $M_{parent} := M_{parent} \cdot \overset{t}{M}_{local}$
- For $child$ **in** J.children:
  - imitate($child$,M_{parent})

source



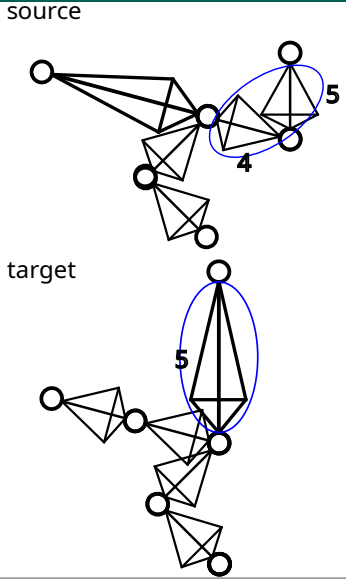
target



Imitating Joint Angles: Step 5

imitate(J,M_{parent})

- Find target Chain C which contains J.
- if C exists
  - set $s := jointIndexing(J.id, corr(C), C)$
  - if $s.size > 0$:
    - $\overset{t}{M}_{newLocal} := {M_{parent}}^{-1} \cdot \overset{s}{M}_{global} \cdot (\overset{s_2}{M}_{local} \cdot ... \cdot \overset{s_N}{M}_{local}) \cdot \overset{s}{OM} \cdot \overset{t}{OM}^{-1}$
    - $M_{parent} := M_{parent} \cdot \overset{t}{M}_{newLocal}$
- else: (no match)
  - $M_{parent} := M_{parent} \cdot \overset{t}{M}_{local}$
- For $child$ **in** J.children:
  - imitate($child$,M_{parent})

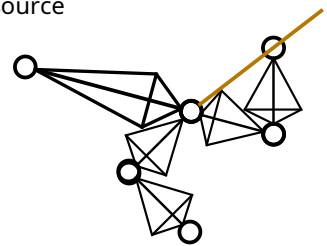


Imitating Joint Angles: Apply IK

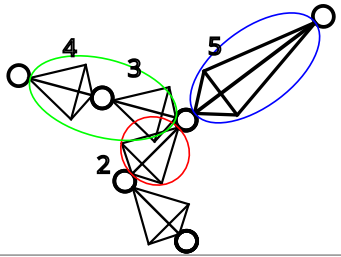
imitate(J,M_{parent})

- Find target Chain C which contains J.
- if C exists
  - set $s := jointIndexing(J.id, corr(C), C)$
- if $s.size > 0$:
  - $\overset{t}{M}_{newLocal} := {M_{parent}}^{-1} \cdot \overset{s}{SM} \cdot {\overset{t}{OM}}^{-1}$
  - $M_{parent} := M_{parent} \cdot \overset{t}{M}_{newLocal}$
- else: (no match)
  - $M_{parent} := M_{parent} \cdot \overset{t}{M}_{local}$
- For $child$ **in** J.children:
  - imitate($child$,M_{parent})

source



target



IK and Chain Scaling

- Use IK on defined Chains to correct pose.
- Desire to take different goals into Consideration e.g. task height.
- Scale IK Target Position T using Chain Length $|C|$.

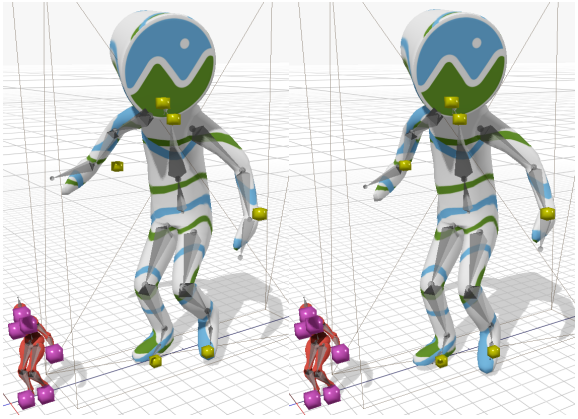
$$T_{new} = \frac{(T - R) \cdot (d(||C_{tar}|| - ||C_{src}||) + ||C_{src}||)}{||C_{src}||} + R$$

- Attach scale on Root position R of target.
- $d = 0 \rightarrow$ source pos.
- $d = 1 \rightarrow$ target pos.
- Adaptable in real-time using UI.
- Decouples height from retargeted motion.

Retargeting Routine

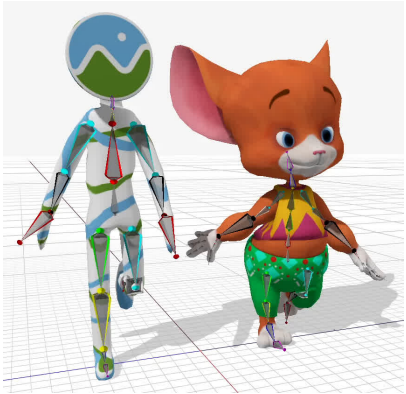
Retargeting occurs implicitly during source character animation playback.

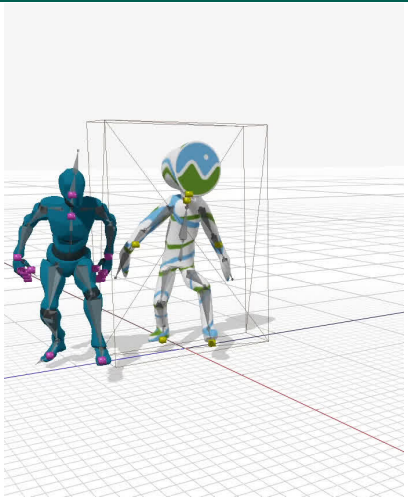
- Initialize Armature + establish Retarget state.
- Loop each frame:
  1. Imitate joint angles of source character.
  2. Rescale temporary target positions of Chains.
  3. Solve all IK chains to clean up artifacts.
- Interactively change:
  - IK parameters and solvers per limb
  - scale target offsets
- Bake and Export Animation.



Evaluation

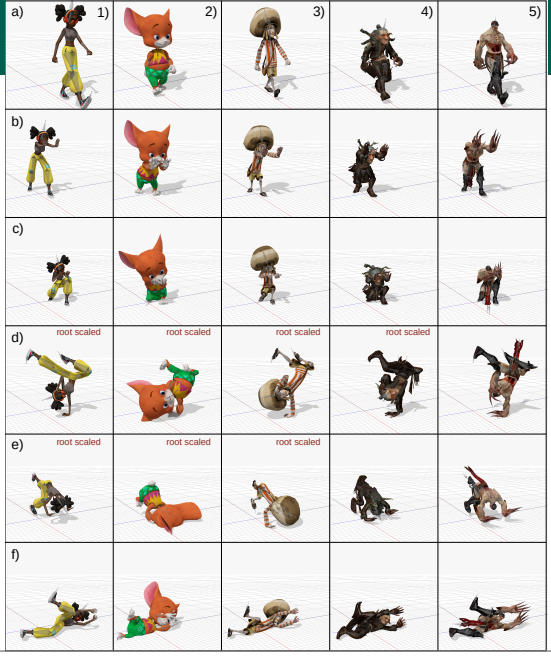
- Implemented in C++ using 3D FOSS Framework CrossForge.
- Realized CCD IK, FABRIK and Jacobian IK.
- Jacobian IK look best, but more costly.
  - → Option to only choose when needed.
- Independence from any Skeletal Structure.
- Minimal manual effort.





Conclusion

- Novel motion retargeting framework.
  - Streamlined Matching process.
  - Real-time interactive.
  - Open for adaptation of various tasks.
  - Easier to implement. (I hope)
- Future Work:
  - Implement IK Target priorities.
  - Easing Func. / F-Curves on scaling parameters.
  - Integrating physics-based simulations.
  - Incorporate other Retargeting methods into Framework.
  - Make Blender / Godot Addon.
- Source Code Available at:
  - https://github.com/nSkade/VRAR25-MotionRetarget



Thank you for your attention!

Any questions?