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

Mick Körner [1], Thomas Kronfeld [1], and Guido Brunnett [1]

Fig. 1: Our novel kinematic chain-based approach demonstrates significant extensibility and supports various settings, showcasing real-time motion retargeting with complete independence from skeletal structure and out-of-the-box support for detailed motion transfer, including hand movements. The figure highlights these capabilities through examples of retargeting between characters that differ in size, height, and proportions. Three character pairs demonstrate this: in each pair, the character on the left provides the motion and the character on the right is the recipient.

Abstract: Transferring motion from one character to another is a crucial operation in character animation. This procedure, also known as motion retargeting, is essential for many applications, including computer-generated films, virtual reality and video games. We present a novel kinematic chain-based approach for real-time animation transfer between virtual characters of diverse skeletal structures and proportions. Our method automates key setup processes, including armature matching by extracting non-branching joint sequences, and employs a novel heuristic skeleton matching to probabilistically establish limb correspondences based on joint positions and chain alignment. For motion transfer, a joint angle imitation technique intelligently adapts source animation to the target, even with rest pose differences. To reduce artifacts like foot skating or self-penetration we use a dynamic chain scaling method that adjusts target positions based on limb length ratios. Our framework is characterized by its high adaptability and extensibility, allowing distinct inverse kinematics solvers for individual limbs, providing a flexible open-source framework for animators and researchers.

Keywords: Motion Retargeting, animation, real-time.

[1] Technische Universität Chemnitz, GDV, Straße der Nationen 62, 09111 Chemnitz, Germany,
mick.koerner@s2019.tu-chemnitz.de, https://orcid.org/0009-0004-8092-1885;
thomas.kronfeld@informatik.tu-chemnitz.de, https://orcid.org/0000-0002-6399-8352;
guido.brunnett@informatik.tu-chemnitz.de, https://orcid.org/0000-0002-8224-015X

1 Introduction

Creating realistic character animation through automatic motion synthesis for articulated characters remains a difficult topic. Although motion capture technology has somewhat made it easier to manually create expressive and organic human motion, there aren't good open source tools available for modifying already existing motion to adapt to changing environments or diverse characters. Utilizing previously collected motion capture data requires the use of motion retargeting. Motion Retargeting is the process of transferring motion data generated for a skeleton with specific hierarchy and bone lengths to another skeleton that differs in one or both of these characteristics. The transferred motion should imitate the motion of the source skeleton as closely as possible.

The naive retargeting approach entails the process of transcribing the motion applied to a joint from the source character to the target character. This is achieved by defining joint correspondences between the source and target character. For each joint, one-to-one, many-to-one, or no mapping needs to be defined manually or in a semi-automatic approach. After that, the initial poses are aligned through the recursive computation of rotation offsets between corresponding joints, from the root to the leaf. Finally, the motion data is transferred via the conversion of rotation angles from local to global coordinates of the source, and back to the local coordinates of the target skeleton.

Unlike traditional sequential techniques that adjust limbs in isolation, numerical optimization methods consider interdependence between all body segments within a unified mathematical framework. The employment of numerical optimization facilitates the integration of multiple objectives simultaneously, including motion semantics preservation and constraint satisfaction.

However, all of these approaches are susceptible to various issues, including but not limited to ground penetration, self-interpenetration, incorrect directions due to differences in rest pose, foot sliding, and missing target objects.

Numerical motion retargeting methods approach the problem through simultaneous optimization of the entire armature's motion, similar to the classification of numerical inverse kinematics in opposition to heuristic approaches that handle individual body parts independently.

Current commercially available motion retargeting software, for instance, Mixamo [Ad25], Autodesk MotionBuilder [Au25], Maya HumanIK or AccuRIG [Re25], yields high-quality results but remains closed source. Hence, detailed insights into their processing pipeline and, consequently, adoption are not possible. On the other hand, existing open-source implementations such as SmartBody [Te25] or the popular Rokoko Blender Plugin [Ro25] are often limited to a specific method or skeleton format, require substantial user input, or rely on outdated methods. There is also a lack of open-source frameworks that allow the integration of complex motion retargeting algorithms and thus enable a comparison and optimization of such algorithms.

The proposed framework addresses these challenges by presenting a highly adaptable real-time Kinematic Chain-Based Motion Retargeting solution.

Our main contributions include:

1. **Automated Armature Setup** for extracting body parts and defining kinematic chains from arbitrary skeletal structures.
2. **Heuristic Skeleton Matching** for semi-automatically establishing limb correspondences between source and target characters quickly.
3. **Adaptive Motion Retargeting:** An intuitive joint angle imitation technique that intelligently adapts motion data to the target character.
4. **Advanced Kinematic Control** by integrating various Inverse Kinematic solvers on a per-body-part basis.
5. **Dynamic Kinematic Chain Scaling**, rescaling end-effector target positions to account for user-defined tasks.
6. **Open-Source Framework** in which all proposed methods are implemented in an adaptable, and extensible manner, enabling real-time interactive motion retargeting.

The subsequent sections of the paper are organized as follows. In Section 2, the relevant literature is reviewed. Thereafter, a broad overview of the proposed solution is provided in the opening section of Section 3. Armature generation and matching are addressed in section 3.1, and subsequent motion transfer is discussed in section 3.2. Finally, we present the achieved results in section 4 and conclude our work in section 5.

2 Related Work

The field of computer animation has intensively researched motion retargeting methods for character animation transfer. To overcome the difficulties of transmitting motion between characters with various morphologies, researchers have put forth a variety of strategies. An overview of related work for each step of the motion retargeting process is given in this section.

2.1 Armature Creation and Skeleton Matching

Due to the abundance of different skeleton hierarchies, the initial process of determining correspondences between joints remains a challenging problem. Therefore, many methods require animators to manually set the joint correspondences between the two hierarchies [MBM05; Ta12], or allow different proportions but require identical hierarchies [CK00; Gl98].

Hecker et al. [He08] for example, proposed special tools that support animators in defining joint correspondences. Feng et al. [Fe12] extended this method by incorporating automated

matching by exploiting joint naming conventions and the identification of body parts by their topology. The utilization of naming conventions to differentiate between left and right joints is a common practice. However, there are instances where these conventions are not followed, which can lead to complications, particularly in scenarios where motion capture systems employ numerical joint identifiers as the sole form of naming.

Monzani et al. [Mo00] introduce an Intermediate Skeleton as a bridge between the Performer and End User Skeletons. This Skeleton is achieved by using Inverse Rate Control to enforce spatial constraints. Similarly, Du Sel et al. [DCR15] employed an intermediate skeleton representation called Golaem Skeleton for both the source and target skeletons. This approach enabled the conversion of motions to the intermediate representation for playback on any target. However, as with other methods, the user is required to establish a one-to-one correspondence between joints.

In recent years, a kinematic chain-based abstraction has introduced a simplified approach, wherein the definition is limited to joints within a kinematic chain, thereby reducing the complexity of the problem. Abdul-Massih et al. [AYB17] proposed the utilization of Groups of Body Parts. Similarly, Tang et al. [Ta12] semi-automatically align groups of body parts based on their representative vectors' starting point and direction. Their definition is less restrictive but requires more user intervention.

Aberman et al. [Ab20] describe Skeletal Pooling, collapsing degree two joints, which often results in a graph similar to Groups of Body Parts. However, the alignment of limbs remains a subject that necessitates further consideration, particularly in the context of automated solutions.

2.2 Motion Transfer

Despite the presence of analogous rest poses among two characters, non-redundant components of the skinning matrix continue to exert an influence on the application of motion data to a rig. Consequently, the straightforward application of motion data from one rig to another is not feasible. The incorporation of Bind Pose Matrices becomes imperative for the determination of the Base Coordinate Systems of joints, thereby accounting for discrepancies. For example, Tang et al. [Ta12] outline the utilization of adjustment matrices for this purpose but they do not offer a detailed explanation of the mathematics thereby leaving readers uncertain about the practical implementation of this approach. The rotation of each joint is then copied from the source to the target and adjusted in accordance with the initial posture alignment.

Most early methods for motion retargeting perform a constrained optimization problem over the skeleton of a character. Gleicher [Gl98] optimized the output motion sequence guided by kinematic constraints. This approach is aimed at minimizing the magnitude of changes and restricting their frequency content to preserve the qualities of the transferred motion.

Several authors include physical plausibility constraints in the optimization [Al18; PW99; TK05], including ground contact constraints.

Another approach is to first solve an inverse kinematics (IK) problem for each motion frame, and then apply spacetime smoothing to the result [AL11; CK00; Ha16; LS99; Un08]. Several proposed methods combined inverse kinematics for instance with prioritized constraints [CB04], end-effector importance [Sh01] or intermediate/normalized skeletons [KMA05; Mo00] to handle both motion adaptation and retargeting.

Many of these methods produce high-quality results for their considered use cases but fail on others. Our method overcomes these issues by allowing the user to interactively switch between constraints as well as to change the inverse kinematic solver used for each limb individually. Thus, our system can enforce multiple constraints and use different inverse kinematics solvers simultaneously, delivering high-quality results for a wide range of use cases.

Prior to motion transfer, Abdul-Massih et al. [AYB17] separated motion features of individual user-defined Groups of Body Parts (GBP) into two distinct categories: positional and angular. Positional features, in this context, denote a trajectory of representative vectors from the base to the leading joint. These vectors encapsulate fundamental components of the motion and are subsequently employed in positional constraints within the framework of an inverse kinematics solver. The angular amplitude features delineate the contraction or extension of body parts in relation to the neutral input motion, thereby capturing the motion style. The angular amplitude constraints are defined as the limits of rotation. Motion transfer is then computed on a per GBP basis, similar to the approach described by Du Sel et al. [DCR15]. Our method adapts and extends these approaches by also transferring motion in a per limb-based fashion.

Retargeting human motion using neural networks is an emerging research area that focuses on transferring motion patterns from one character to another. These methods rely on various types of input data to accurately capture and transfer movement patterns.

Methods operating on a skeletal level try to abstract out the dynamics of the source sequence and to reproduce it on a target character with a different morphology, that is to a skeleton with different bone lengths and possibly a different topology. In the seminal work of Villegas et al. [Vi18], a recurrent neural network architecture with a Forward Kinematics layer and cycle consistency based adversarial training objective is proposed. This method captures high-level properties of an input motion and adapts them to a target character with different bone lengths. It utilizes cycle consistency to learn the motion transfer problem in an unsupervised manner, working online to adapt the motion sequence frame-by-frame on-the-fly. Aberman et al. [Ab20] extended the scope of retargeting to skeletons with different topologies, but with the restriction that all topologies considered are homeomorphic.

In contrast to skeleton-based methods, Wang et al. [Wa23] proposed a motion retargeting method taking character meshes as input, considering motion retargeting as pose deformation

transfer. Their method uses a hierarchical coarse-to-fine approach resulting in an average reduction of point-wise mesh distances compared to the state-of-the-art methods while also preserving motion semantics for low-resolution meshes. Similarly, Ye et al. [Ye24] focuses on skinned mesh motion retargeting but directly models correspondences on the dense geometric mesh.

Biswas et al. [Bi21] try to combine the advantages of skeleton-based motion retargeting and shape deformation modeling to handle context and geometric detail. They employ a posed skeleton and the point cloud of the posed mesh for training. Similarly, the method proposed by Villegas and colleagues [Vi21] uses the human motion sequence and a target skeleton and character geometry as input. Their goal is to identify self-contacts and ground contacts in the input movement and thus to optimize the movement of the target skeleton so that these contacts are maintained and mutual penetration is reduced.

These machine learning approaches have demonstrated efficacy in the domain of retargeting, offering enhanced quality outcomes. However, limitations persist in the realm of interactivity and the modification of retargeted motion. Moreover, the training process of these models is characterized by a substantial time investment including retraining of models when new features are required. Additionally, many referenced works state that the results of motion retargeting are subjective and depend on the goal of corresponding application tasks at hand.

2.3 Motion Cleanup Approaches

Most of the methods described above leave it up to the user to manually remove any artifacts that occur. Motion Cleanup Approaches are methods that try to fix the artifacts generated during motion transfer. For example, Tang et al. [Ta12] utilize inverse kinematics as a post-process to rectify the resultant motion and enforce the alignment with Cartesian constraints. Similarly, Feng et al. [Fe12] utilize Jacobian-based inverse kinematics to enforce constraints in a post processing step. Ming-Kai Hsieh et al. [MBM05] directly splits the retargeting process into two steps. Firstly, they transfer and concatenate of motions, and secondly, they generate smooth transitions and cleanup artifacts.

Only machine learning based methods that directly use mesh data try to minimize artifacts during the transfer [Bi21; Vi21]. Ye et al. [Ye24] uses semantically consistent sensors to establish dense mesh correspondences between characters and develops a dense mesh interaction field to capture both contact and non-contact interactions between body geometries. This method aims to preserve motion semantics, prevent self-interpenetration, and ensure contact preservation.

3 Proposed Framework

Recent machine learning approaches have demonstrated high efficacy in retargeting, producing excellent results. However, limitations remain in terms of interactivity and the

ability to modify retargeted motion. Training these models is also time-consuming and requires retraining when new features are needed. Numerous referenced works also assert that the outcomes of motion retargeting are subjective and contingent upon the specific application task.

Conversely, open-source implementations are often lacking in non-machine learning methods, which limits the comparison and optimization of such algorithms. Automated or semi-automated methods of matching joints or kinematic chains between two given skeletons are also often not considered.

Whilst inverse kinematics (IK) is already a common tool that allows animators to quickly obtain a desired pose, a well-implemented, accessible and adaptable motion retargeting method could further improve an animator's workflow by providing a basis for a desired sequence that can then be refined using other motion editing tools. Therefore it is worth considering motion retargeting as an adaptable and extensible tool, akin to IK.

Figure 2 shows the flowchart of our proposed motion retargeting approach.

The process begins with the rig setup, during which the skeletons and animations of the target and source characters are imported. We define an armature as a kinematic chain abstraction representing individual body parts. After character alignment, our method automatically creates an armature and subsequently generates targets for the limbs. The next phase is the retargeting setup. Here, we match the limbs of the source and target characters using heuristics. Manual reassignment is an option if adjustments are needed.

Finally, the retargeting routine commences. This involves executing real-time motion retargeting on animations played back from the source character to the target character. Our novel method imitates motion style using armatures and uses IK for artifact clean-up. While evaluating the quality of the animation, various options are provided to refine it. These include adapting IK target scaling offsets, incorporating constraints, and using different inverse kinematics solvers with various settings for different limbs to accommodate various tasks. If the animation is satisfactory, it can be baked and exported.

We begin with armature creation and matching in Section 3.1, detailing our method for representing characters' skeletal structures as armatures and how to create them automatically in Section 3.1.1. Section 3.1.2 then describes a skeleton matching heuristic that establishes limb correspondences.

Subsequently, Section 3.2 elaborates on core mechanisms for transferring motion. Presenting our novel Joint Angle Imitation technique in Section 3.2.1, and Kinematic Chain Scaling to dynamically adjust target positions in Section 3.2.2. Finally, in Section 3.3, we integrate these components into a Retargeting Routine, describing the real-time editor routine including initialization, state establishment, and the per-frame retargeting loop.

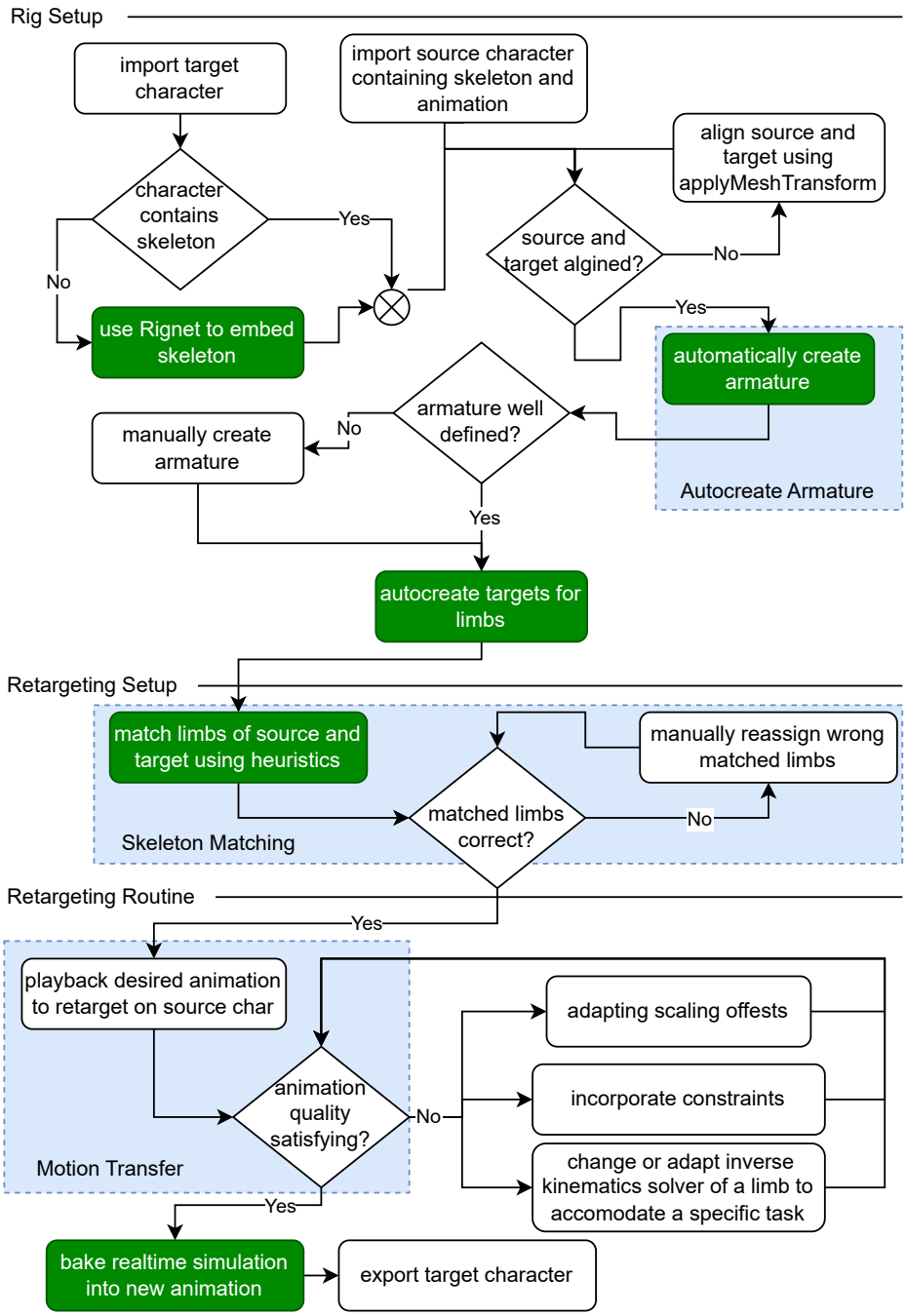

Fig. 2: User workflow of the implemented Motion Retargeting Editor. Green indicates automated processes, while not many, most of the manual tasks are minor or checks to assure motion quality. Blue boxes represent our proposed methodologies, with motion transfer incorporating joint angle imitation and kinematic chain scaling, both applied in real time.

3.1 Armature creation and matching

In our proposed framework, an armature is defined as a collection of non-branching, non-intersecting joint chains representing individual body parts, which is specific to each character. It handles multiple inverse kinematics constraints and access for follow-up methods. Solving an armature involves solving each chain in its collection with designated solvers. This can be done either in a specific order or by utilising priority weighting to determine the importance of the constraints. Figure 3 shows an example of a user-definable joint chain to which an inverse kinematics constraint has been applied.

The ability to import and export an Armature for a character in a structured data format (like JSON), including all mentioned members, for later retrieval and use allows for faster testing and adaptation.

While the Joint Chains inside an Armature can be user defined and edited, similar to other existing methods, it remains a difficult process. Additionally, manually matching created joint chains between two Characters is tedious. The following two sections detail methods on how to automatically generate Armatures and match Joint Chains more quickly.

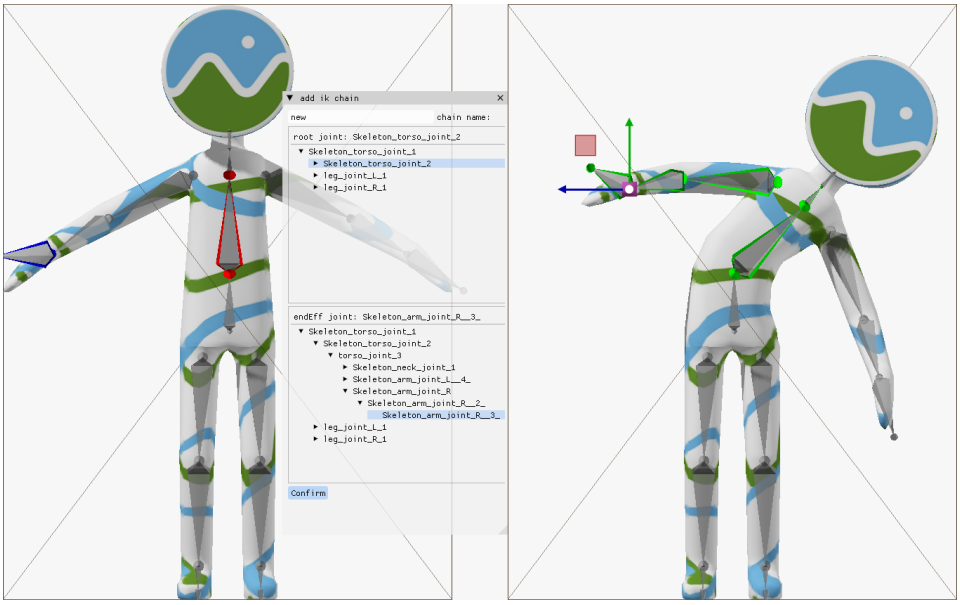

Fig. 3: Visualization of a user-defined joint chain added to the character's armature. Inverse kinematics constraints can be applied to each chain using inverse kinematics solvers that are definable and configurable for each chain.

3.1.1 Autocreate Armature

Algorithm 1 aims to automatically generate an armature from any skeleton using the branches of the joint tree to create kinematic chains. Traversal of the skeleton commences from the root bone, and the definition and appending to a new chain continues until a joint with two or more children is encountered. Upon reaching a branch, the definition of the current Chain is finalized, thereby initiating the creation of a new chain accordingly. In order to identify chains, the first joint name is used as the chain name, which can later be renamed using the chain editor.

Algorithm 1 Automated IK Chain Generation from Armature.

```
1:  procedure GenerateIKChains(ArmatureController)
2:      Initialize empty JointChains map
3:      function PropagateJoint(CurrentJoint, ChainID)
4:          JointChains[ChainID].insert(CurrentJoint)
5:          if |CurrentJoint.Children| > 1 then
6:              ▷ Branching point; each child starts a new chain.
7:              for ChildJoint in CurrentJoint do
8:                  PropagateJoint(ChildJoint,ChildJoint.Name)
9:              end for
10:         else
11:             if |CurrentJoint.Children| == 1 then
12:                 ▷ Continue current chain.
13:                 PropagateJoint(Child,CurrentJoint.Name)
14:             end if
15:         end if
                                          ▷ End effector (ChildCount == 0) terminates chain.
16:     end function

17:     PropagateJoint(RootJoint, ).
18:     Output: Populated JointChains for inverse kinematics.
19: end procedure
```

While this method is particularly effective for rudimentary rigs and can even be applied to fingers, it has drawbacks. Compared to skeletal pooling, it doesn't utilize a strategic collapse of joints, keeping key joints such as knees, elbows, and ankles intact, leading to the oversight of potentially crucial motion characteristics. However, for similar characters and in our proposed motion imitation approach, this can be neglected.

Another potential issue concerns rigs that contain control bones. Rather than deforming the mesh directly, control bones influence other bones or mechanisms in the rig, enabling animators to create complex movements efficiently. Utilizing Importers that convert these control bones into joints results in the generation of additional branches, thereby initiating new chains at branches containing control joints. However, these bones can be easily identified and flagged, as they have no skinning weight influences on any vertices.

3.1.2 Skeleton Matching

A significant number of motion retargeting tools, both past and present, demand the establishment of a joint-to-joint correspondence between two characters. Our kinematic chain-based abstraction has introduced a simplified approach, wherein the definition is limited to joints within a kinematic chain, thereby reducing the complexity of the problem. However, the alignment of limbs remains a subject that necessitates further consideration, particularly in the context of automated solutions.

Equation 1 calculates the probability factor P for a matched chain pair c. T_{root} and T_{eef} represent the root and end effector position of the chain pairs from source and target character s and t. The calculation incorporates inverse distances between end effector and root joint pairs (α and β), the direction the chain is pointing (γ), as well as the mean joint position (δ). Each of these predefined weights can be optionally adjusted interactively to further refine the matching if desired.

$$
\begin{aligned}
P_c =& \alpha \frac{1}{1 + ||\overset{t}{T}_{eef} - \overset{s}{T}_{eef}||} + \beta \frac{1}{1 + ||\overset{t}{T}_{root} - \overset{s}{T}_{root}||} \\
&+ \gamma \left(\frac{1}{2} \left\langle \frac{\overset{t}{T}_{eef} - \overset{t}{T}_{root}}{||\overset{t}{T}_{eef} - \overset{t}{T}_{root}||}, \frac{\overset{s}{T}_{eef} - \overset{s}{T}_{root})}{\overset{s}{||T}_{eef} - \overset{s}{T}_{root}||} \right\rangle + \frac{1}{2}\right) \\
&+ \delta \frac{1}{1 + ||\frac{1}{\overset{s}{|T|}} \sum_{k=0}^{\overset{s}{|T|}} \overset{s}{T}_k - \frac{1}{\overset{t}{|T|}} \sum_{k=0}^{\overset{t}{|T|}} \overset{t}{T}_k||}
\end{aligned}
\tag{1}
$$

In general, the height of humanoid skeletons is known to vary, thereby incorporating a larger γ compared to α and β poses to be advantageous.

Equation 1 is then used to calculate a score for each source-target pair of chains. A higher value of P compared to other pairs indicates a closer relationship between the chains in question. The pair with the highest probability is added to the list of matched chains. This process continues until either the target armature runs out chains to match. The proposed formula has been found to be effective for similar poses, including A and T poses, which are commonly seen in humanoid resting positions. Corrections to initial guess parameters and individual matches can be made rapidly and easily, substantially streamlining the matching process.

Nonetheless, the presented formula requires the initial alignment of the two characters to be correct. Although automated alignment through the Iterative Closest Point (ICP) method is

a possibility by interpreting joints of both characters as point clouds, it is not a beneficial approach in most cases.

3.2 Motion Transfer

3.2.1 Joint angle Imitation

As previously mentioned, due to the presence of discrepancies in rest pose and misalignment of Base Koordinate Systems of Joints, direct application of motion data from one rig to another is not feasible. The subsequent explanation is an intuitive derivation that elucidates the process of imitating joint angles and computing the necessary adjustments. This is achieved by utilising the standardised skinning matrix decomposition, which is widely used for blend skinning in engines, file formats, and research.

Offset matrices (also called Inverse Bind Pose Matrix) transform the corresponding joint to the object space center of the character, thereby representing its actual base coordinate system transformation. Skinning Matrices on the other hand describe the transformation of each joint from rest pose to a set pose. This transformation ensures that local rotations of a joint are applied correctly. The Skinning Matrix SM of single Joint is computed as follows:

$$SM = M_{parent} \cdot M_{local} \cdot OM \tag{2}$$

where M_{local} is the Local Joint transformation, M_{parent} are all parent transformations up to the root joint and OM is the Offset matrix of the individual joint.

Let overset s be the source and overset t be target components. We want the Rotational component of target Skinning Matrix to be equal to the source Skinning Matrix in object space, but independently of parent joints. Given that the Skinning Matrix encodes global transformation, it is advantageous to utilize it as an optimal method for directly extracting local orientation for the desired target armature.

In order to compute the adjusted local transformation, M_{parent} and OM need to be removed relative to the target joint:

$$\overset{t}{M}_{newLocal} = \overset{t}{M}_{parent}^{-1} \cdot \overset{s}{M}_{parent} \cdot \overset{s}{M}_{local} \cdot \overset{s}{OM} \cdot \overset{t}{OM}^{-1} \tag{3}$$

$$\overset{t}{M}_{newLocal} = \overset{t}{M}_{parent}^{-1} \cdot \overset{s}{SM} \cdot \overset{t}{OM}^{-1} \tag{4}$$

Algorithm 2 details the imitation of joint angles. The procedure is initiated at the root joint, and each joint in the target skeleton is traversed down to the end effectors of the armature. For each joint, the following steps are taken:

Algorithm 2 Kinematic chain-based joint angle imitation. It involves the imitation of joint transformations through the identification of corresponding limbs, the transfer of motion, and the maintenance of posture in the absence of a match, with these processes being recursively applied to child joints.

procedure IMITATE
  Input: J ▷ current joint
  Input: M_{parent} ▷ parent joint transform
  Input: $corr$ ▷ chain correspondences between CS and CT
  Find target limb CT which contains J.
  if CT exists **then**
    ▷ Use defined limb correspondence to find matched limb CS from source chains.
    $CS := corr(CT)$
    ▷ Run customizable joint Indexing Function to retrieve matched source joint idx_{js}
    $idx_{js} := jointIndexingFunc(idx_{jt}, CS, CT)$
    if idx_{js} is a valid joint **then**
      Construct source joint local transform matrix $\overset{s}{M}_{local}$
      Construct source parent joint transform $\overset{s}{M}_{parent}$
      ▷ use angle imitation function to transfer motion.
      $\overset{t}{M}_{newLocal} := \overset{t}{M}_{parent}^{-1} \cdot \overset{s}{SM} \cdot \overset{t}{OM}^{-1}$
      $M_{parent} := M_{parent} \cdot \overset{t}{M}_{newLocal}$ ▷ update M_{parent}
      Deconstruct $M_{newLocal}$ and set local joint transform
    end if
  end if
  if no joint was matched **then**
    ▷ keep current posture
    Construct target joint local transform matrix $\overset{t}{M}_{local}$
    $M_{parent} := M_{parent} \cdot \overset{t}{M}_{local}$ ▷ update M_{parent}
  end if
  for $child$ **in** J.children **do**
    imitate($child$,M_{parent})
  end for
end procedure

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.

If there are more target than source joints in the chain (injective mapping), certain target joints are skipped, keeping restpose transformation of the parent. On the other hand if there are more source than target joints (surjective mapping) rotation of multiple source joints

have to be combined to avoid omitting motion information. For surjective mapping the retargeted local transformation can be extended to:

$$\overset{t}{M}_{newLocal} = \overset{t}{M}_{parent}^{-1} \cdot \overset{s}{M}_{global} \cdot (\overset{s_2}{M}_{local} \cdot ... \cdot \overset{s_N}{M}_{local}) \cdot \overset{s_N}{OM} \cdot \overset{t}{OM}^{-1} \tag{5}$$

Where $\overset{s}{M}_{global}$ is the global transformation of the source joint and s_i for $i > 1$ are subsequent child joints of s in up to s_N in the source Chain.

As of now, it remains unclear which joints from the source chain should be matched for motion transfer. To address this, a joint indexing function will automatically determine which joints angular information should be taken from. There are numerous ways in which joint indexing could be realized. For instance, joints could be chosen based on distance, index in the chain, or overall distribution within the chain.

In the ideal case of a bijective mapping, disregarding the lengths of the joints while mapping them will result in greater differences in end-effector positions the greater the disparity in joint lengths. However, inverse kinematics should adequately address this scenario. If the mapping is not bijective, an appropriate indexing function must be identified to imitate the motion style and follow the source end-effectors as closely as possible. In the case of an injective mapping, redundant target joints align with another target joints in the chain. These joints can then be used during IK if required. Conversely, a surjective mapping combines multiple source joints into a single target joint by combining motion data using equation 5.

For non-bijective mappings, our approach leverages a mapping based on normalized cumulative lengths, as outlined in algorithm 3. It establishes a normalized, linear representation of both the source and target kinematic chains by assigning corresponding nodes the distance of each joint to the root joint of the chain. These lengths are then normalized to the range of zero to one, thus disregarding the chain's overall geometric length.

After normalization, a greedy one-to-one mapping is performed. To determine the optimal pairing between the source and target nodes, the absolute difference between their normalized cumulative lengths is calculated. These pairings are then sorted by distance in ascending order. By iterating through this sorted list, the method creates an initial nonoverlapping assignment, taking the best available match for each target node and ensuring that each source node is used for at most one target node.

Next, any source nodes not assigned in the initial one-to-one mapping are assigned to the closest mapped target node. Due to the reduced number of joints in compared chains, it is feasible to compute the mapping for every joint in real time, caching the mapping result is also an option.

The root bone is a unique element that necessitates specialized management due to its pivotal role in determining the character's overall posture. Its distinct position within the hierarchy of bones ensures that it does not fall within the category of any specific limb.

Algorithm 3 Joint Indexing Joint based on normalized cumulative lengths

```
1:  procedure MapNodes(src, tar)
2:      srcNorm, tarNorm ← array of normalized cumulative joint lengths
3:      sortedMappings ← empty list
4:      for i from 0 to numTarNodes - 1 do
5:          for j from 0 to numSrcNodes - 1 do
6:              distance = |tarNorm[i] − srcNorm[j]|
7:              Add {distance, {i, j}} to sortedMappings
8:          end for
9:      end for
10:     Sort sortedMappings
11:     nodeMapping ← empty map, usedSrcNodes ← empty set
                                                    ▷ Greedy one-to-one mapping
12:     for each {distance, {tarIdx, srcIdx}} in sortedMappings do
13:         if tarIdx not mapped AND srcIdx not used then
14:             nodeMapping[tarIdx] ← srcIdx
15:             usedSrcNodes.add(srcIdx)
16:         end if
17:     end for
                                    ▷ Map unassigned source nodes to nearest mapped target
18:     finalMap ← ConvertToFinalMap(nodeMapping)
19:     for each unused srcIdx do
20:         bestTarIdx ← FindClosestMappedTarget(srcIdx, nodeMapping)
21:         finalMap[bestTarIdx].add(srcIdx)
22:     end for
        return finalMap
23: end procedure
```

Our proposed method is characterized by its decoupling of angle imitation from skeletal topology, a feat achieved by leveraging matched chains for the determination of on-the-fly joint-to-joint correspondence. The proposed algorithm for designated modes Identity and Restpose relative is challenging to adapt. In the future, proper rest relative mode has to be integrated by incorporating rest pose differences at the root of the chain.

3.2.2 Kinematic Chain Scaling

Subsequent to the transfer of joint angles, the resulting posture will display artifacts, as previously discussed. To rectify these artifacts, a desired IK solver is employed, with the targets representing the source characters' end effectors.

The key motivations behind motion retargeting are not only facilitating the transfer of animation across different skeletal structures, but also across a range of sizes and heights. Nevertheless, establishing a definition of a good quality motion retargeting remains challenging, as its objective quality is dependent upon the specific task at hand.

To elucidate, one may consider an animation of a virtual character holding a box, with different arm lengths. During scaling, it is not clear if the box should maintain its position in object space or be held according to the targets natural holding pose. The same analogy can be applied to the study of gait motion in relation to transferred walk speed and root height.

Methodologies for adjusting the target position according to variations in armature between the source and target are outlined in the relevant literature. Within our framework, the simplest approach at hand is to compare the lengths of the matched limbs and determine the target's position relative to the limb scale.

Let T denote the target position and R the root position of the inspected chain C:

$$T_{new} = \frac{(T - R) \cdot ||C_{tar}||}{||C_{src}||} + R \tag{6}$$

Depending on the task specification, this term can be extended using linear interpolation term d:

$$T_{new} = \frac{(T - R) \cdot (d(||C_{tar}|| - ||C_{src}||) + ||C_{src}||)}{||C_{src}||} + R \tag{7}$$

In the case of $d = 0$, the source relative position of the target is employed. Conversely, when $d = 1$, the rescaled target position is utilized.

Algorithm 4 presents the process employed following the imitation of joint angles. Target and root positions for each chain match are rescaled in the Motion Retargeting routine update function incorporating linear interpolation options exposed in the user interface. Additionally targets are placed relative to the root position of each chain.

Algorithm 4 Target rescaling process. Computing target limb lengths based on source limb lengths and interpolating positions using specified scaling values for limbs and roots ensures accurate transformation between corresponding chains.

procedure TARGETRESCALE
  Input: SLL ▷ array of source limb lengths
  Input: TLL ▷ array of target limb lengths
  Input: $corr$ ▷ chain correspondences between CS and CT
  Input: $limbScale$ ▷ array of interpolation values for limbs
  Input: $rootScale$ ▷ array of interpolation values for roots
  for CS and CT **in** $corr$ **do**
    $scale := lerp(1., TLL[CT]/SLL[CS], limbScale[CT])$
    $pos_{root} := lerp(CS.pos_{root}, CT.pos_{root}, rootScale[CT])$
    $dir := CS.pos_{target} - CS.pos_{root}$ ▷ direction vector from of source limb
    $CT.pos_{target} := pos + dir \cdot scale$
  end for
end procedure

3.3 Retargeting Routine

The motion retargeting routine is comprised of three distinct phases: initialization, establishment of the motion retargeting state, and the retargeting loop itself.

The initialization process entails the storage of temporary references to the source and target characters. Concurrently, the source and target limb lengths are initiated, and temporary targets are automatically generated. These targets serve as references for the target characters' chains to track as an IK constraint throughout the procedure. This process persists until motion retargeting is disabled.

As targets are defined in local space, world space transformation is not taken into consideration. However, in order to view both models clearly, world space transformation can be applied beforehand to separate the source and target characters spatially, while keeping them logically at the same place.

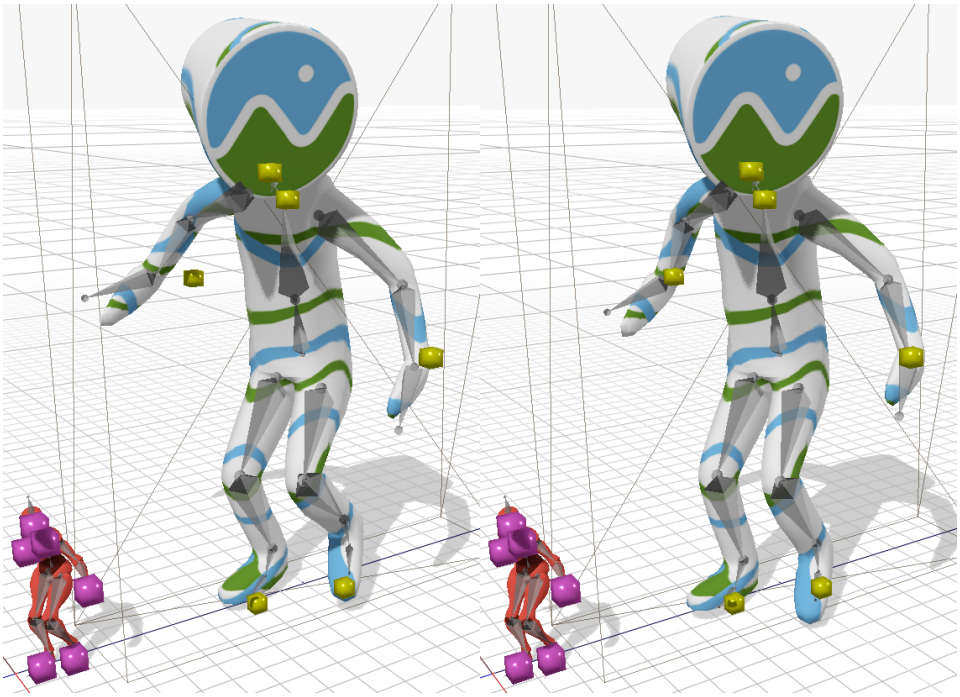

Fig. 4: Retargeted motion on the left only applies joint imitation, copying the root joint angle and position. Due to differences in the rest pose, Cesium Man does not reach its targets. On the right, inverse kinematics is used on the defined limbs of the armature to reach the designated targets of the source animation. The temporary targets of the retargeting routine are visualized in yellow. Notice the size difference between the models handled by Kinematic Chain Scaling.

Subsequent to initialization, retargeting occurs implicitly during the animation playback of the source character. At this phase, temporary target points, which the target character tracks, are updated by the source character targets.

Thus the Retargeting routine looks as follows:

1. imitate joint angles of the source character
2. rescale temporary target positions for each matched chain
3. solve the target armature using ik to clean up potential artifacts

This routine is executed in each frame, thereby reinitializing the cleanup required pose by imitating joints. Although IK yields plausible results for an individual pose of a keyframe, its temporal and spatial coherency across frames is not utilized yet.

Figure 4 shows the proposed motion retargeting technique in action. By rescaling target positions relative to the matched limbs, motion can be retargeted between characters of different sizes without issue.

To use the retargeted motion outside of our editor, it can be exported by examining the state of the target animation controller at designated intervals during the retargeting process.

4 Discussion

We implemented our motion retargeting approach in CrossForge [Uh20], an open-source, cross-platform 3D visualization framework developed by Tom Uhlmann at Chemnitz University of Technology. The core of our modular framework is an interactive editor that additionally facilitates the importing and exporting of rigged characters in common formats and IK interaction with the armature for testing purposes. We have utilized and tested various inverse kinematics solutions, including Cyclic Coordinate Descent (CCD), FABRIK [AL11], and Jacobian IK.

Characteristic differences between these IK methods, as described in the related literature, have been observed. Although Aristidou et al. [AL11] state that FABRIK produces more natural results than CCD or Jacobian inverse kinematics, this is not necessarily the case for all types of motion.

For example, it may be true that FABRIK produces more plausible results for grasping or reaching motions. High energy motions, which are described as requiring more physical force, result in characteristic energy minimization as a measure of reducing energy expenditure. This can be observed in running or punching motions where all limb muscles and levers are used. Jacobian Inverse has the ability to represent these more naturally due to its way of distributing change and thus work over all bones instead of those closest to the end effector.

Figure 5 signifies this difference between FABRIK and Jacobian IK.

Monzani et al. [Mo00] describe a similar observation of Jacobian methods. But also note that they generally give better results for motion cleanup due to their characteristic distribution of energy over the entire chain, preserving the motion style better compared to CCD or FABRIK, where the rate of change is more concentrated at the end or root of the chain, depending on the type used.

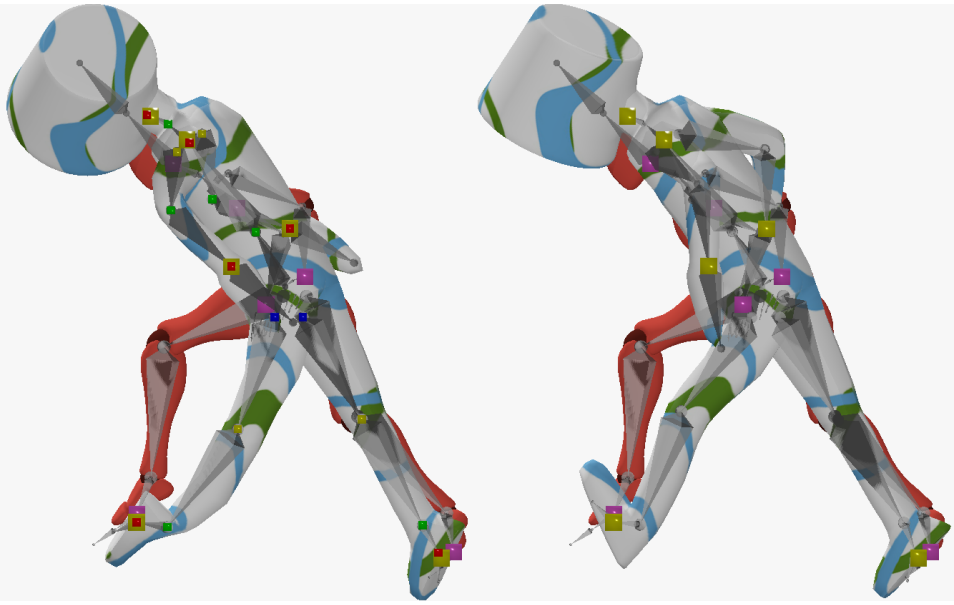

Fig. 5: Motion retargeted from the Mixamo X bot to Cesium Man without imitating joint angles and only utilizing Inverse Kinematics. On the left side, FABRIK is used, resulting in the knee not bending forward. On the right side, Jacobian IK is used, showing the knee bending more in its supposed direction.

Although Jacobian methods require a substantial computational investment and yield unrealistic motion when the effector is far from the target, they remain a relevant consideration when transferring motion. The incorporation of Jacobian in only the necessary chains emerges as a viable option in our proposed system, thereby reducing the runtime by leveraging CCD or FABRIK when the conditions are sufficient.

The lack of established datasets for motion retargeting approaches makes comparison and quality evaluation difficult. To promote the reproducibility of results, the free Mixamo dataset [Ad25], which provides both Characters and Animations, was used. As demonstrated in Figure 6, a diverse array of poses from animations is employed to illustrate retargeting quality on various characters exhibiting a wide range of proportions. All characters were in the T rest pose, and the animation was sourced by Mixamo's Y bot character, which has similar proportions to the X bot in previous figures.

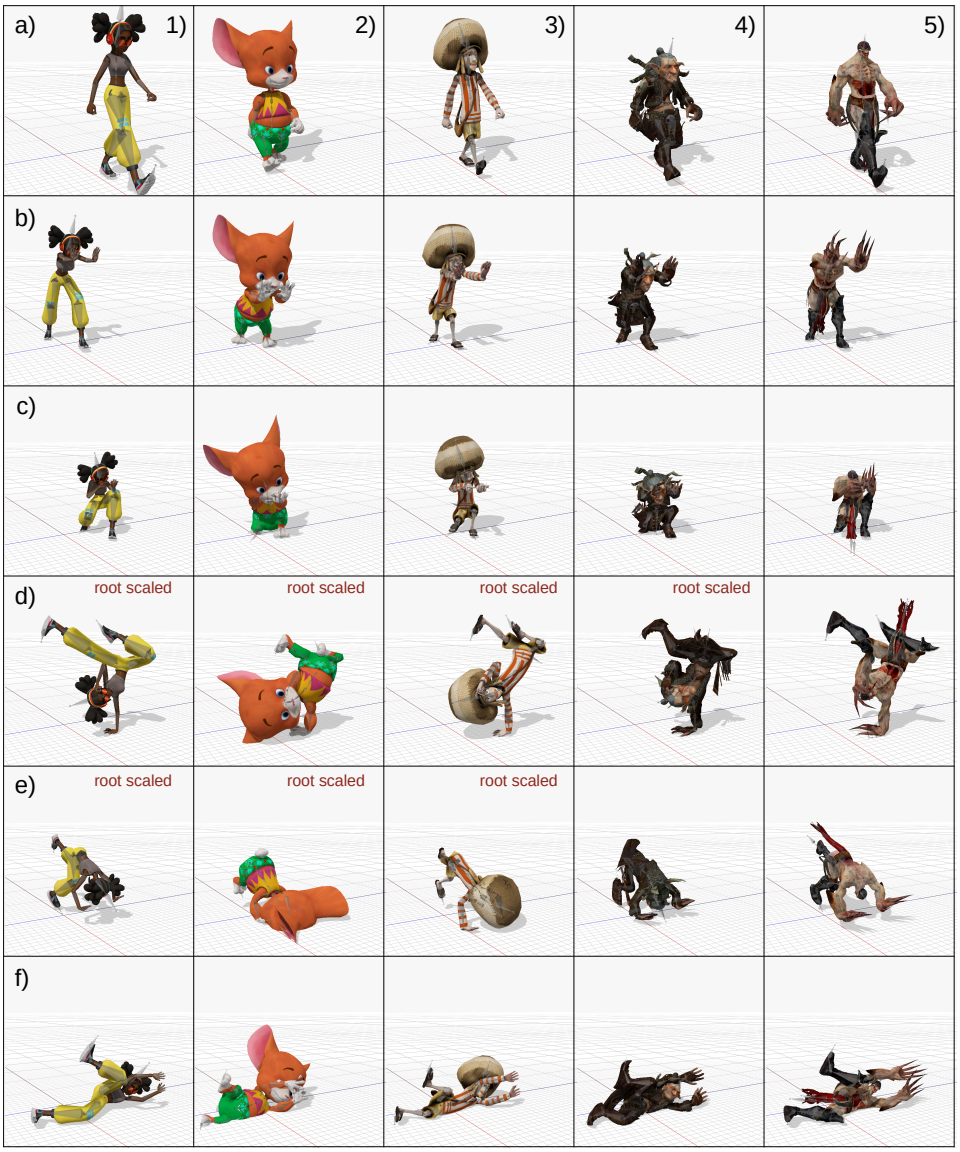

Fig. 6: Various retargeted poses from the Mixamo dataset [Ad25] are presented. Animations of source Y bot (not displayed) include the following: a) Walking, b) Male Action Pose, c) Strafing, d) Breakdance Freeze Var 2, e) Inverted Double Kick To Kip, and f) Goalkeeper Body Block. These animations are applied to target characters: 1) Michelle, 2) Mousey, 3) Kaya, 4) Goblin, and 5) Skeletonzombie. Poses labeled with "root scaled"required root joint height re-scaling to mitigate ground penetration.

In order to ground poses where both legs are in the air, such as the handstand in d), it was necessary to temporarily rescale the root position using the presented limb scaling method to avoid ground penetration or floating. However, when both feet were grounded in the same animation, penetration or floating would then occur there instead. Moving forward, limb scaling parameters could be easily bound to the sequencer. This would allow for smooth transitions using easing functions (f-curves) at arbitrary keyframes.

Currently, IK chains are evaluated in sequence, which can potentially overwrite subsequent chains. For example, if the right arm is evaluated before the spine, the right arm will be moved again by the end effector of the spine chain. While this phenomenon may not be noticeable during real-time interactions due to temporal coherence, it can pose a challenge when manipulating joint angles during motion retargeting. This can result in unintended changes to the chain structure. Selecting chains to prioritize reach is also desirable. For example, if both targets for the left and right arms are out of reach but would be within reach if the spine flexed accordingly.

While our framework offers several advantages, there are several promising avenues for future research. One area for further exploration is enhancing the robustness of heuristic skeleton matching for highly dissimilar characters. Integrating physics-based simulations that can automatically handle complex interactions and environmental constraints, such as collisions and ground contact, could be beneficial.

5 Conclusion

In this paper, we presented a comprehensive framework and dynamic editor that are designed to streamline the complex process of real-time, kinematic-chain-based motion retargeting for virtual characters. Our work addresses critical challenges in motion retargeting and character setup by introducing a novel, kinematic-chain-based approach that prioritizes adaptability, real-time performance, and extensibility.

The significance of this work extends beyond mere technical advancements. By open-sourcing our framework, we aim to foster community collaboration and establish a foundation for future evaluation and development of motion retargeting methods. The extensive set of simple yet powerful operations for rigged characters demonstrated in our editor is applicable to less-abstracted environments and offers a versatile toolkit for character animation. Because the editor was developed with real-time performance in mind, motion retargeting and related methods can be tested and prototyped quickly during development. The real-time capabilities of our system enable rapid iteration and immediate feedback, allowing the advantages and drawbacks of the prototyped methods to be addressed quickly. This will be invaluable for creating interactive content, including applications in virtual reality, video games, and real-time simulations in the future.

Our source code is available in the following GitHub repository: https://github.com/nSkade/VRAR25-MotionRetarget

References

[Ab20] Aberman, K. et al.: Skeleton-aware networks for deep motion retargeting. ACM Transactions on Graphics (TOG) 39 (4), pp. 62–1, 2020.

[Ad25] Adobe Inc.: Mixamo: Online 3D Character Animation Service, Accessed: 2025-08-30, Adobe Systems Incorporated, 2025, https://www.mixamo.com/%5C#/.

[AL11] Aristidou, A.; Lasenby, J.: FABRIK: A Fast, Iterative Solver for the Inverse Kinematics Problem. Graphical Models 73 (5), pp. 243–260, 2011, https://linkinghub.elsevier.com/retrieve/pii/S1524070311000178, accessed: 07/12/2024.

[Al18] Al Borno, M. et al.: Robust Physics-based Motion Retargeting with Realistic Body Shapes. Computer Graphics Forum 37 (8), pp. 81–92, 2018.

[Au25] Autodesk Inc.: Autodesk MotionBuilder: 3D Character Animation Software, Accessed: 2025-08-30, Autodesk, Inc., 2025, https://www.autodesk.com/products/motionbuilder/overview.

[AYB17] Abdul-Massih, M.; Yoo, I.; Benes, B.: Motion Style Retargeting to Characters With Different Morphologies. Computer Graphics Forum 36 (6), pp. 86–99, 2017, https://onlinelibrary.wiley.com/doi/10.1111/cgf.12860, accessed: 09/30/2024.

[Bi21] Biswas, S. et al.: Hierarchical Neural Implicit Pose Network for Animation and Motion Retargeting, 2021, arXiv: 2112.00958 [cs.CV].

[CB04] Callennec, B. L.; Boulic, R.: Interactive motion deformation with prioritized constraints. In: Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation. SCA '04, Eurographics Association, Grenoble, France, pp. 163–171, 2004.

[CK00] Choi, K.-J.; Ko, H.-S.: Online Motion Retargetting. The Journal of Visualization and Computer Animation 11 (5), pp. 223–235, 2000.

[DCR15] Du Sel, Y. P.; Chaverou, N.; Rouillé, M.: Motion Retargeting for Crowd Simulation. In: Proceedings of the 2015 Symposium on Digital Production. DigiPro '15: The Digital Production Symposium. ACM, Los Angeles California, pp. 9–14, 2015, https://dl.acm.org/doi/10.1145/2791261.2791264, accessed: 07/12/2024.

[Fe12] Feng, A. et al.: Automating the Transfer of a Generic Set of Behaviors onto a Virtual Character. In (Kallmann, M.; Bekris, K., eds.): Motion in Games. Vol. 7660, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 134–145, 2012, http://link.springer.com/10.1007/978-3-642-34710-8_13, accessed: 09/30/2024.

[Gl98] Gleicher, M.: Retargetting Motion to New Characters. In: Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques - SIGGRAPH '98. The 25th Annual Conference. ACM Press, Not Known, pp. 33–42, 1998, http://portal.acm.org/citation.cfm?doid=280814.280820, accessed: 07/12/2024.

[Ha16] Harish, P. et al.: Parallel Inverse Kinematics for Multithreaded Architectures. 35 (2), 2016.

[He08] Hecker, C. et al.: Real-Time Motion Retargeting to Highly Varied User-Created Morphologies. ACM Transactions on Graphics (TOG) 27 (3), pp. 1–11, 2008.

[KMA05] Kulpa, R.; Multon, F.; Arnaldi, B.: Morphology-independent representation of motions for interactive human-like animation. Computer Graphics Forum, 2005.

[LS99] Lee, J.; Shin, S. Y.: A hierarchical approach to interactive motion editing for human-like figures. In: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques. SIGGRAPH '99, ACM Press/Addison-Wesley Publishing Co., USA, pp. 39–48, 1999, https://doi.org/10.1145/311535.311539.

[MBM05] Ming-Kai Hsieh; Bing-Yu Chen; Ming Ouhyoung: Motion Retargetting and Transition in Different Articulated Figures. In: Ninth International Conference on Computer Aided Design and Computer Graphics (CAD-CG'05). Ninth International Conference on Computer Aided Design and Computer Graphics (CAD-CG'05). IEEE, Hong Kong, China, pp. 457–462, 2005, http://ieeexplore.ieee.org/document/1604675/, accessed: 07/14/2024.

[Mo00] Monzani, J.-S. et al.: Using an Intermediate Skeleton and Inverse Kinematics for Motion Retargeting. Computer Graphics Forum 19 (3), pp. 11–19, 2000, https://onlinelibrary.wiley.com/doi/10.1111/1467-8659.00393, accessed: 07/12/2024.

[PW99] Popović, Z.; Witkin, A.: Physically based motion transformation. In: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques. SIGGRAPH '99, ACM Press/Addison-Wesley Publishing Co., USA, pp. 11–20, 1999, https://doi.org/10.1145/311535.311536.

[Re25] Reallusion Inc.: AccuRIG: Free Auto Rig for Any 3D Character, Accessed: 2025-08-30, Reallusion Inc., 2025, https://actorcore.reallusion.com/auto-rig/accurig.

[Ro25] Rokoko: Rokoko Studio Live Blender Plugin, Accessed: 2025-08-30, Rokoko Electronics ApS, 2025, https://github.com/Rokoko/rokoko-studio-live-blender.

[Sh01] Shin, H. J. et al.: Computer puppetry: An importance-based approach. ACM Trans. Graph. 20 (2), pp. 67–94, 2001.

[Ta12] Tang, C. et al.: Motion Retargeting for Characters with Heterogeneous Topologies. In: 2012 5th International Congress on Image and Signal Processing. 2012 5th International Congress on Image and Signal Processing (CISP). IEEE, Chongqing, Sichuan, China, pp. 756–760, 2012, http://ieeexplore.ieee.org/document/6469919/, accessed: 11/22/2024.

[Te25] Team, S.: SmartBody: Character Animation Platform, Accessed: 2025-08-29, USC Institute for Creative Technologies, 2025, https://smartbody.ict.usc.edu/about.

[TK05] Tak, S.; Ko, H.-S.: A physically-based motion retargeting filter. ACM Trans. Graph. 24 (1), pp. 98–117, 2005.

[Uh20] Uhlmann, T.: CrossForge: A Cross-Platform 3D Visualization and Animation Framework for Research and Education in Computer Graphics, 2020, https://github.com/Tachikoma87/CrossForge.

[Un08] Unzueta, L. et al.: Full-body performance animation with Sequential Inverse Kinematics. Graphical Models 70 (5), pp. 87–104, 2008.

[Vi18] Villegas, R. et al.: Neural kinematic networks for unsupervised motion retargetting. In: Proceedings of the IEEE conference on computer vision and pattern recognition. Pp. 8639–8648, 2018.

[Vi21] Villegas, R. et al.: Contact-Aware Retargeting of Skinned Motion. In: 2021 IEEE/CVF International Conference on Computer Vision (ICCV). 2021 IEEE/CVF International Conference on Computer Vision (ICCV). IEEE, Montreal, QC, Canada, pp. 9700–9709, 2021, https://ieeexplore.ieee.org/document/9710501/, accessed: 07/12/2024.

[Wa23] Wang, H. et al.: HMC: Hierarchical Mesh Coarsening for Skeleton-Free Motion Retargeting. CoRR abs/2303.10941, 2023, https://doi.org/10.48550/arXiv.2303.10941.

[Ye24] Ye, Z. et al.: Skinned Motion Retargeting with Dense Geometric Interaction Perception. In: The Thirty-Eighth Annual Conference on Neural Information Processing Systems. 2024.