

MIRRORED-Anims: Motion Inversion for *Rig-space Retargeting* to Obtain a *Reliable Enlarged Dataset* of Character Animations

Théo Cheynel

Kinetix, LIX, École Polytechnique, CNRS,
IP Paris
Paris, France

Thomas Rossi

Kinetix
Paris, France

Omar El Khalifi

Kinetix
Paris, France

Oscar Fossey

Kinetix
Paris, France

Damien Rohmer

LIX, École Polytechnique, CNRS, IP Paris
Palaiseau, France

Marie-Paule Cani

LIX, École Polytechnique, CNRS, IP Paris
Palaiseau, France



Figure 1: MIRRORED-Anims allows to easily build a large-scale dataset of paired motions for a variety of humanoid characters. Given two skinned characters, our solution relies on the automatic creation of high-level control rigs to define a rig-space retargeting process. As shown above, it transfers feet and hand contacts without any sliding or floating behavior.

Abstract

We propose MIRRORED-Anims, a novel retargeting procedure for transferring motion between skinned humanoid characters of different morphologies. It is designed so as to mimic the strengths of the closed-source Mixamo’s retargeting method, currently used as a standard to create motion databases and train all state-of-the-art learning-based retargeting methods, despite severe shortcomings (namely, a lack of character diversity and notable penetration artifacts). Taking inspiration from the toolsets of 3D animators, our retargeting algorithm relies on the control rigs used to manipulate skinned characters, by identifying and transferring controller values on predefined bone mechanisms. While producing motions which are closer to Mixamo’s ground truth than any state-of-the-art

learning-based technique, MIRRORED-Anims creates fewer penetration artifacts than observed in the Mixamo dataset, improving the perceived quality of the output. Moreover, motion can be retargeted in real-time to and from the SMPL body model, making it possible to leverage the large motion databases available in SMPL format for the retargeting task. Because it relies solely on transparent, explainable rig operations, MIRRORED-Anims can be used to generate ground-truth motions for any humanoid character, providing a reliable baseline for the future training of learning-based methods. Project page: <https://mirrored-anims.github.io/MIRRORED-Anims>

Keywords

Motion Retargeting, Motion Transfer, Character Animation, Control Rig, Collision Avoidance

1 Introduction

The task of transferring motion from one animated 3D character to another, called “retargeting”, is a long-standing problem in the field of computer animation. Indeed, solving this highly challenging task requires understanding the influence that different bone lengths, skeletal structures, and body shapes may have on motion, in order to find the best compromise and generate a new movement that most perceptually resembles the input.

In an attempt to capture the implicit interplay of these different factors, recent advances relied on deep neural networks to learn the distribution of motions for each character using unsupervised learning, which then allows transferring motion between characters. To this end, NKN [Villegas et al. 2018] introduced the “Mixamo” dataset, which consists in unpaired motions for a dozen different characters. The motions were collected using the Adobe Mixamo software [Adobe 2024], which unfortunately relies on a closed-source algorithm for motion transfer. A family of later works [Aberman et al. 2020; Hu et al. 2023; Lim et al. 2019; Villegas et al. 2021; Zhang et al. 2024a, 2023] rely solely on this dataset to train neural networks for the task of motion retargeting.

Unfortunately, NKN’s “Mixamo” dataset is severely limited, for the following reasons:

- It is limited to a very small number of characters (7 for training, 6 for validation), which all share the same skeleton (with varying bone lengths or missing bones only);
- It includes a relatively small number of motions, when compared to academic motion-capture datasets;
- The “ground-truth” motions in the dataset show frequent artifacts, most notably penetrations into the ground (or lack of floor contacts), loss of semantic-rich contacts, and self-penetrations.

Moreover, we noticed that state-of-the-art methods fail to generalize to unseen characters from the test-set, which have different limb proportions. We claim that this failure is a direct consequence of the aforementioned limitations of the Mixamo dataset. While the field of retargeting would greatly benefit from a larger dataset, Mixamo’s retargeting method is entirely closed-source, only available through Adobe’s web platform, and subject to a prohibitive license that does not allow its direct use for machine learning.

To solve these issues, we propose a new, open-source retargeting method, named MIRRORRED-Anims, aimed at providing a new baseline for motion retargeting tasks, which can be used to generate large quantities of data on virtually any character. We designed our method to:

- Accept any humanoid character as input, regardless of its morphology or the number of bones in the skeleton;
- Produce high-quality animations, both in terms of measurable metrics (foot sliding, motion smoothness, self-penetration) and in terms of perceived quality;
- Be explainable, so that it can rationally be trusted to generate new data for training purposes, with a minimal amount of data curation.

To meet these goals, we draw inspiration from the way that 3D animation experts manually edit motions. In particular, we borrow one of the most widespread tools from their production pipelines: “control rigs”, which are interfaces that allow them to manipulate characters using a set of high-level controllers, instead of editing the

pose of each bone separately. These control rigs are engineered by artists called “riggers”, to provide an appropriate set of controllers, based on the requirements of 3D animators. We asked a rigger to provide a control rig template adapted to our use case, and we propose a method to automatically build this control rig from any humanoid character. Moreover, we introduce a dedicated algorithm for “rig inversion”, allowing us to extract the control values from the input animation on the source character. We then propose a procedure for “skeleton-aware rig-space retargeting”, in which we compute a first version of the target character’s control values. Finally, we adapt the controller values by performing a morphology-based correction, to alleviate penetration artifacts. An overview of our method is shown in Figure 2.

To assess the quality of the MIRRORRED-Anims method, we provide a full analysis of quality of our results based on a series of metrics, a comparison with previous works, and generate a large dataset (shown in Figure 1) to train a learning-method, comparing it with the Mixamo dataset both quantitatively and qualitatively.

2 Previous Work

2.1 Optimization-based retargeting

Optimization-based motion retargeting was first introduced by Gleicher [1998], which frames the task as a space-time optimization problem, aiming to solve the target’s motion so that it respects some pre-identified constraints. Later, [Lee and Shin 1999] refines the approach by solving frame-wise inverse kinematics (IK), and Choi and Ko [2000] adds joint-level constraints. Dynamic constraints are incorporated by Tak and Ko [2005] to enhance physical plausibility. There have been several extensions to skinned characters, with Jin et al. [2017] introducing the Aura Mesh for two interacting characters, and Basset et al. [2020] optimizing energy functions for volume preservation and collision handling. Ho et al. [2010] further addresses collision prevention using hard constraints. Recently, Cheynel et al. [2025] simplifies the mesh to a set of key-vertices to perform real-time optimization in order to address semantic contact conservation.

2.2 Kinematic retargeting

In the domain of virtual reality, real-time interactions are crucial. This is why some solutions in this field rely on purely kinematic approaches, that do not require computationally expensive optimization steps. These methods often model the relationship between the joints of the characters: Molla et al. [2017] uses egocentric coordinates and inverse kinematics, and Delahaye et al. [2023] adds finger retargeting in a real-time scenario. Qualitative evaluations from this neighbouring field seem to highlight that kinematic approaches offer promising results for our motion retargeting objective.

2.3 Learning-based retargeting

In contrast, most recent methods use a data-driven approach, training deep learning models to generate solutions to the retargeting problem. The first, seminal work was NKN [Villegas et al. 2018], which initiates the use of the Mixamo dataset containing motions retargeted by Adobe Mixamo [Adobe 2024] on a dozen characters. They argue that supervised learning is not possible, as acquiring paired motion data for source and target characters is very complex;

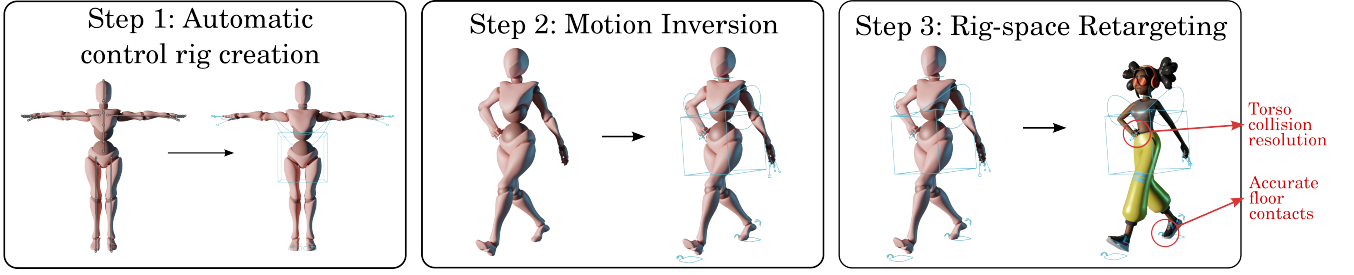


Figure 2: The three steps of our the MIRRORED-Anims retargeting method. First, we automatically create a control rig for the source and target characters, using a modular template that we adapt to their skeleton and morphology. Then, for a given input animation, we perform motion inversion: that is, we find the rig controls that give the same animation. This allows us to perform a rig-based retargeting, with a collision resolution method to keep fine contact details with the torso.

instead, they train a recurrent neural network using a cycle consistency loss and a discriminator. PMNet [Lim et al. 2019] later tries to decouple the pose and the overall movement (root trajectory). SAN [Aberman et al. 2020] introduces convolution operations on skeletons, to allow the network to take diffeomorphic skeletons as input. Training an encoder/decoder architecture for each skeleton, with a shared latent space, allows them to perform retargeting without the need for ground-truth pairs of motion on different characters. SAME [Lee et al. 2023] also trains an autoencoder to learn a skeleton-agnostic latent-space, performing retargeting amongst other tasks. PAN [Hu et al. 2023] adopts a similar approach, but for each limb instead of treating the whole body at once, and Yan et al. [2023] applies a similar principle for human-to-robot retargeting.

In addition to skeletal information, a few methods also take into consideration the mesh of the character, in order to avoid inter-penetrations during motion while preserving accurate contacts. Villegas et al. [2021] identifies limb contacts on the source character, and enforced preservation of contacts using a geometry-informed RNN. R²ET [Zhang et al. 2024b, 2023] attenuates both self-collisions and missing contacts by integrating geometry processing inside their training losses, while MeshRet [Ye et al. 2024] encode fine-grained mesh information for a more precise semantic correspondence of contacts. Even more recently, SMTNet [Zhang et al. 2024a] performs a differentiable rendering of the characters’ meshes, and uses a vision-language model to evaluate the conservation of motion semantics.

First, it is worth mentioning that those works perform evaluation on the mean-squared error (MSE) in 3D joint positions with respects to the Mixamo dataset. However, as pointed out by ReCon-ForM [Cheynel et al. 2025], there are several qualitative issues with the motions of the Mixamo dataset, caused by artifacts of Adobe Mixamo’s closed-source retargeting method.

Moreover, none of the aforementioned methods were able to use supervised training, due to the lack of ground truth dataset of sufficient size and on a large-enough variety of characters. Therefore, instead of relying on paired motions on source and targets character, the models are implicitly trying to learn the distribution of acceptable motions for each character, based on their limb proportions and morphology. This task is arguably more difficult than supervised learning (which the current Mixamo dataset does not allow), especially given its extremely limited character diversity

that makes it difficult for models to generalize to new characters with unseen limb proportions or body shape.

We argue that the field of retargeting is missing a way to obtain high-quality data on an almost continuous distribution of input skeletons and morphologies, and introduce our method as a reliable way to generate a larger training dataset. A high-quality reliable retargeting method could already be used by itself in a lot of contexts, but training learning-based methods is still relevant for later use when a differentiable retargeting is needed (for instance, when training other learning-based architectures), or to incorporate new losses that are hard to encode manually.

2.4 Control rig inversion

Despite its widespread use in industrial settings, the concept of control rigs remains largely unexplored in academic literature. When used, control rigs are most often seen as a black box, given as input from an animation expert. This “black box” interpretation of the rig assimilates it to a function $f : c \mapsto (M_n)$, which takes as input a set of control values c , and returns either the transformation matrices M of the N bones (which drive the mesh through skinning), or directly that of the mesh’s vertices.

Some previous works manipulate control rigs through the task of “rig inversion”, which consists in finding the inverse f^{-1} of that black-box function – essentially, finding the control values which match a given skeleton pose. For instance, Holden et al. [2015] learn an inverse rig mapping using gaussian process regression, along with the Jacobian of the control rig, which allows for an optimization procedure to reach more accurate control values. Follow-up work [Holden et al. 2017] shows the impact of deep learning instead of traditional machine learning techniques. Marquis Bolduc and Phan [2022] take a similar approach, by first teaching a multi-layer perceptron to approximate the rig function, before teaching a separate network to invert the first one. Gustafson et al. [2020] derive an analytical approximation of the rig, and invert the rig using a finite-difference method. Several other works specifically focus on facial rigs (which behave slightly differently from our use case), once again taking the “black box” approach [Racković et al. 2024; Racković et al. 2023].

It is worth noting that industrials have presented new animation systems that rely on some flavour of control rigs [Bereznyak 2016; Hecker et al. 2008]. Both of them allow for easy retargeting to characters of different morphologies; however, motions shown

are only locomotion, and the proposed systems are not released. Similarly, using “simplified” or “virtual” skeletons to manipulate poses for the retargeting task has also been proposed [Kulpa et al. 2005; Monzani et al. 2000], which can be understood as prototypes of control rigs.

To the best of our knowledge, no previous academic work explains in detail the inner mechanisms that drive the construction of control rigs. In contrast, we provide a detailed explanation of these mechanisms and leverage their strengths to propose a robust and efficient retargeting method.

3 Method Overview

We propose a pipeline to transfer motions from one animated character to another. Our method comprises three main stages, depicted in Figure 2:

- Given a source and target skinned characters (meshes + bones + skinning weights), we automatically build two control rigs adapted to their respective morphologies. We also pre-compute a proxy for the shape of the torso, using a set of cone primitives, which will serve during the collision refinement stage.
- Given the animation of the source character’s bones, we perform “rig inversion”, which is a procedure designed to find the position, rotation and scale of the rig’s controllers, in order to replicate the original motion.
- Finally, we perform retargeting in rig-space, where the high-level abstraction allowed by the rig makes it effortless to transfer the most important aspects of motion (for instance, foot unrolling while walking, or hand positions with respect to one another). We also use our mesh proxy to identify collisions (or lack thereof) on the source motion and transfer them to the target character.

4 Automatic Control Rig

Artists often use “control rigs” as a tool to manipulate 3D characters in a more intuitive way. Using the advice of an experienced rigging artist, we built a template control rig in the open-source Blender software [Community 2025]. Our rig implements several mechanisms to allow for easier manipulation of the character’s skeleton, especially thought for the task of motion retargeting. Moreover, we propose a method to automatically build such a rig for any input character, depending on its morphology, by following the steps described next. For more detail on the rig structure, we provide the full control rig on a sample character on the project webpage.

4.1 Heel bone placement

A preliminary requirement is to place a mechanical bone (i.e., a bone not directly deforming the mesh via skinning) for the heel of the character, which serves as a starting point for the foot roll mechanism. To do this, we select the vertices which are skinned to all foot bones. We keep only the vertices whose height is in the lowest 25% range, as we are interested only in the sole of the foot. We then project the 3D position of the vertices on the ground plane, and perform singular value decomposition (SVD) on the 2D positions, taking care to weigh each vertex by the average size of the faces it belongs to. The SVD yields two vectors $\vec{u}$ and $\vec{v}$, which correspond to the axis with the most variation (arguably, the foot’s

longitudinal axis) and the axis with the least variation (arguably, the foot’s side-to-side axis). We compute the oriented bounding box along those two axes; that is (p^i being the position of vertex i):

$$\begin{aligned} \bullet \quad u_{\min} &= \min_i p^i \cdot \vec{u}, \quad u_{\max} = \max_i p^i \cdot \vec{u} \\ \bullet \quad v_{\min} &= \min_i p^i \cdot \vec{v}, \quad \text{and} \quad v_{\max} = \max_i p^i \cdot \vec{v} \end{aligned}$$

This allows us to place the heel bone between the points $u_{\min} \vec{u} + v_{\min} \vec{v}$ and $u_{\min} \vec{u} + v_{\max} \vec{v}$ – with a height of 0 so it rests on the ground plane in 3D space. We provide additional figures illustrating this process in the Supp. Mat.

4.2 Foot mechanism

We then automatically build a standard foot roll mechanism, which has two rotation-only controllers. The core mechanism revolves around two dual-hinge systems, one for the pronation (ankle rolling inwards) and supination (ankle rolling outwards), and the other for the dorsiflexion and plantar flexion. Figure 3 visually explains the constraint setup which allows for the latter: it is built entirely with “Copy Rotation” and “Limit Rotation” constraints. The “Copy Rotation” transfers the world-space rotation of the controller to the bones (“MCH bone left” and “MCH bone right” in the figure), just before the “Limit Rotation” restricts those same rotations to a specific range, expressed in local space (allowed range illustrated in white in the figure). This effectively creates a dual hinge, where a positive rotation causes the whole system to rotate around the left bone, and a negative rotation causes the whole system to rotate around the right bone.

Figure 4 demonstrates the usefulness of this mechanism for a walk cycle: the first heel impact (3a) and last toe contact (3b) can be modeled with the rotation of a single controller, while the foot’s position controller stays stationary.

4.3 Hip control

To control the pelvis’ rotation and position in space, we create a control bone which is located directly under the pelvis, at the height of the thigh bones. This vertical offset prevents stretching of the legs when the controller rotates forward / backward.

4.4 Legs and arms inverse kinematics

For hands and feet, we automatically place a controller bone right outside the mesh (in the palm for hands, and under the soles for the feet), to which the hand and feet mechanisms are parented. We then use Blender’s “Inverse Kinematics” constraint, which uses the position of the hand/ankle bone along with the position of a newly created “pole target” controller to drive the bones of the chain.

4.5 Spine and neck rotations

Unlike the foot roll (which is a fairly standard mechanism), there are several valid ways to rig spines. In our implementation, we take a simple approach, which consists in splitting the rotation of the controller equally amongst all spine bones (see Figure 6), allowing for retargeting to a character with fewer or more spines.

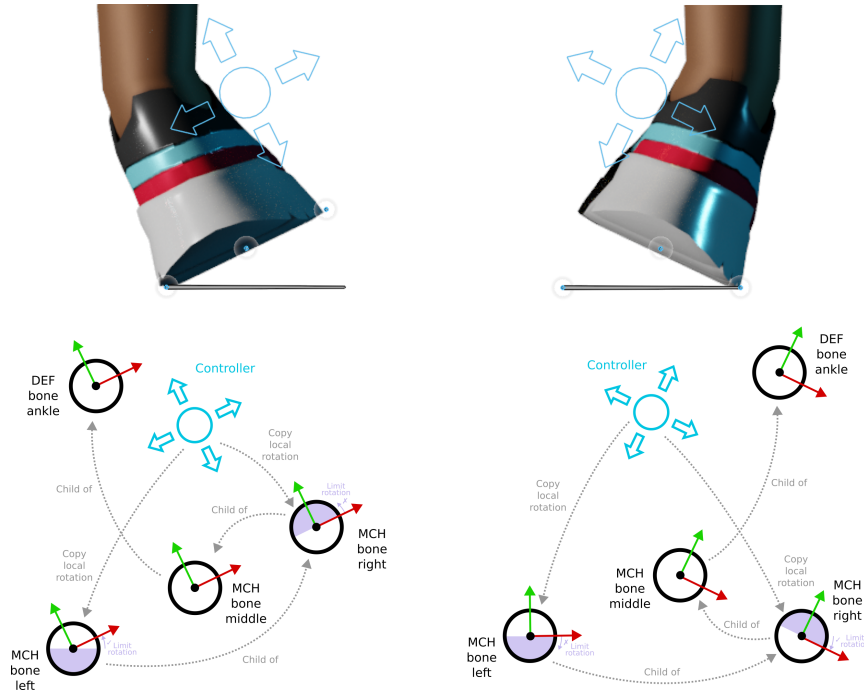


Figure 3: Schematics of the constraint system which allows for the dual-hinge mechanism (bottom), with the corresponding deformations of the mesh (top). DEF denotes the original “deform” bone, and MCH denotes the “mechanical” bones of the control rig.

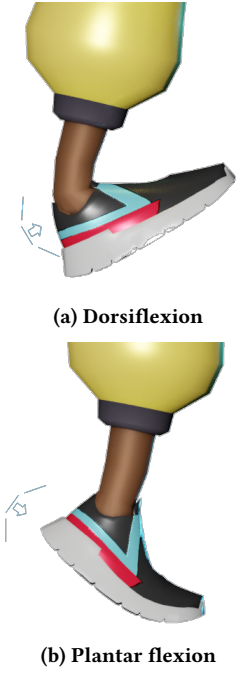


Figure 4: The heel controller allows for easy foot un-rolling, which is useful for animating locomotion.



Figure 6: The spine controller (blue) splits its rotation between the spine bones (red).

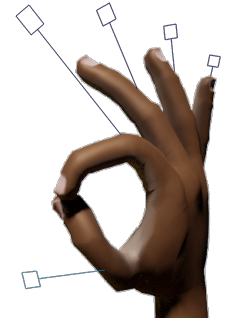


Figure 7: Finger controls.

4.6 Fingers

Finally, our method automatically rigs each finger using a single controller. Its rotation determines the rotation of the proximal phalanx, and its scale controls the rotation of other phalanges (how “curled” the finger is), as shown on Figure 7.

5 Control Rig Inversion

From an animation defined on a source character, rig inversion is the task that consists in finding the right values for the control rig’s parameters, to obtain visually similar poses. In general, control rig inversion is a complex task, which requires scripting a set of rules for each new rig. Moreover, the rig function is neither injective nor

surjective: some poses of the bones simply cannot be reached with the controllers, and several different controller values may yield the same pose. However, in our case, all characters now follow the same simple template. We are thus able to derive a procedure which inverts this control rig, regardless of limb proportions, number of bones, or skeleton topology. Our inverse rig algorithm is based on opinionated rules, which solve the aforementioned ambiguity in a convenient way for the retargeting task. Conceptually, we perform the following:

- We rotate the hip controller and place it so that the rig’s pelvis bone is at the same position as the original skeleton’s.

- We rotate the spine (and neck) controllers so that the rig’s last spine (and head) have the same rotation as the original skeleton’s.
- We determine the rotations of the two controllers of the feet (case disjunction depending on the heel and toes rotations).
- We determine the rotation and scales of the fingers controllers so that the rig’s first and last phalanges has the same rotation as the original skeleton’s.
- We rotate the hand and feet controllers and place them so that the wrist and ankle bones are at the same positions as the original skeleton’s.
- We determine the angle of the pole targets of arms and legs IK chains that match the position of the elbows / knees.

The bones of the original skeleton and the control rig having different matrices in T-pose, several changes of coordinate frames are required at each step (i.e., “same rotation” means “same world-space rotation between matrices in T-pose and in final pose”). We left them out in the conceptual explanation above; as such, for reproducibility purposes, we provide a complete pseudo-code description of this algorithm in the Supplementary Material.

6 Rig-space retargeting

At this point, we now have two rigged characters (the source and target), whose control rigs are automatically created through the process described in Section 4. We also have transferred the animation of the source character onto its control rig, through the algorithm described in Section 5. In this section, we explain how we transfer the motion from the source control rig onto the target character’s control rig, while preserving the motion semantics.

6.1 Driving controllers

First, we analyze the source motion to determine which of the rig’s controllers are “driving” the motion. In traditional character animation (i.e., when no control rig is used), the root bone of the skeleton is often placed near the center of mass of the character (the hips), which – most of the time – doesn’t accurately reflect the way that a human interacts with their environment, unless they are in freefall. As an example, when a human walks, the motion of their hips is due to friction between their feet and the floor, so we can say that each foot is successively “driving” the motion in that case. We propose a mathematical formulation to capture the intuitive concept described above, based on three assumptions:

- Our first assumption is that humans interact mostly with the ground floor, that is, the plane $z = 0$. As such, for each frame f , we assign a first weight to each controller c (hands, feet, hips) based on the inverse of their height h_c^f .
- Our second assumption is that an extremity interacting with the environment tends to stay stationary and in contact. To give more importance to stationary controllers, we define a weight based on the inverse of their velocity $v_c^f = \frac{1}{\Delta t} \|p_c^f - p_c^{f-1}\|$ (where p_c^f is the position of controller c at frame f).
- Our final assumption is that, when several controllers touch the floor, the ones that support the most weight are more likely to stay fixed and “drive” the rest of the motion. As such, we define a third weight for each controller, based on

the inverse of the distance between their position p_c^f and the projection of the hip controller $p_{\text{COM}}^f = p_{\text{hip}}^f - h_{\text{hip}}^f \vec{z}$ on the floor (which approximates the center of mass).

All in all, each controller c is assigned a weight for frame f , which is the product of the three weights defined previously:

$$w_c^f = \frac{1}{1 + \alpha h_c^f} \cdot \frac{1}{1 + \gamma v_c^f} \cdot \frac{1}{1 + \beta \|h_c^f - p_{\text{COM}}^f\|}$$

which we group into the following vector:

$$w^f = \{w_c^f \mid c \in \text{Controllers}\}$$

By applying the softmax, we can get a vector which sums to 1: $W^f = \text{softmax}(T \times w^f)$, where $T = 1$ is the temperature parameter.

The hyper-parameters α , γ and β are used to balance the contributions of the three weights. For our experiments, we arbitrarily estimate threshold values for the three components: that is, we might want a controller whose height $h_c^f = 0.01\text{m}$ to weigh twice as much as a controller whose height $h_c^f = 0.03\text{m}$, giving us $\alpha = 100$. Similarly, we might want a controller whose speed $v_c^f = 0.01\text{m.s}^{-1}$ to weigh twice as much as a controller whose speed $v_c^f = 0.03\text{m.s}^{-1}$, giving us $\beta = 100$. And finally, we might want a controller that is 0.1 m away from the projection of the center of mass to weigh twice as much as a controller that is 0.3 m away, giving us $\gamma = 10$.

6.2 Retargeting with respect to a controller

Say we have a single driving controller (the left foot) on the source character, and that we want to retarget the pose onto the target character’s control rig. We start by putting the target controller at the same position. Then, we can determine the target hip controller’s location, by computing the vector (source ankle to source hip) in the reference frame of the source ankle, scaling it by the ratio between the limb lengths on source and target, and transferring it to the reference frame of the target ankle. We can then perform the same operation to locate the right foot based on the hip position, then each hand based on the last spine position, and finally the pole targets (for the knees and elbows). Additional rules are used at each step to ensure, for example, accurate height with respects to the floor: see Supp. Mat. for full details.

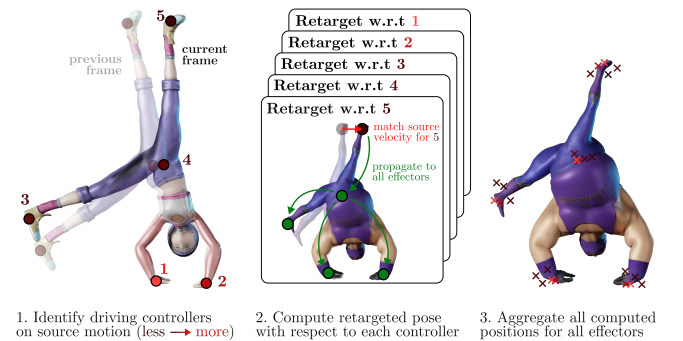


Figure 8: Visual explanation of the retargeting method



Figure 9: Cones fitted to the source character



Figure 10: Output of section 6, showing collisions



Figure 11: After the collision resolution algorithm

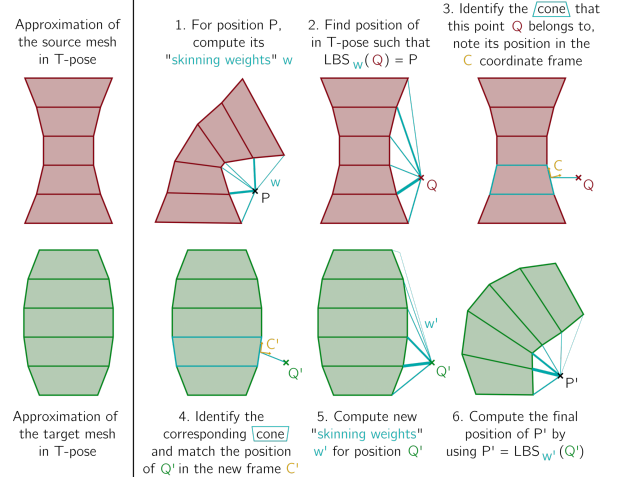


Figure 12: Schematized procedure applied when retargeting controllers close to the torso

6.3 Retargeting consecutive frames

To retarget the first frame of an animation, we first perform the process explained in Section 6.2 for the controller with the highest weight. For each subsequent frame, we then run as many processes as there are controllers. For each target controller, we use its position at the previous frame, along with the world-space velocity of the source controller, to determine its new position, before solving for the rest of the chain (with the same process as described in Section 6.2). This gives us 5 different positions for each controller, and the final result is obtained by a weighted average, using W^f . This hereditary process is depicted on Figure 8.

7 Restoring semantic contacts

Avoiding interpenetration between the character’s limbs is crucial when retargeting motion from a source character whose mesh does not self-collide. In order to achieve this, we refine the retargeted effector positions whenever the input animation drives them close to the surface of the mesh. To achieve this, we first approximate the surfaces of the source and target characters with a set of cones primitives fitted to the mesh vertices, and use the features of this easily manipulable shape representation to perform a mesh-aware refinement of the effector positions. We detail this process next.

7.1 Mesh approximation

Let us focus on approximating the character’s torso, which is the source of most unwanted inter-penetration (generally with the arms) when an animation is transferred to a character of different morphology. To do so, we select the vertices which are skinned to a certain set of k bones $(\mathcal{B}_i)_{1 \leq i \leq k}$ (the hip and spine bones). We consider the axis $\vec{d} = \mathcal{B}_k - \mathcal{B}_1$ defined by the first and last bones. We split the vertices into n uniformly sampled slices along this axis: that is, the i -th slice contains vertices $\{v_j \mid (v_j - \mathcal{B}_1) \cdot \frac{\vec{d}}{\|\vec{d}\|^2} \in [\frac{i-1}{k}, \frac{i}{k}]\}$.

For each slice, we fit a circle to the vertices using a least-squares method. The successive circles describe a set of truncated cones,

which individually provide a convex approximation of the mesh between two slices, as shown on Figure 9.

Moreover, using the average skinning weights of the vertices, we can effectively skin each circle to each of the bones $(\mathcal{B}_i)$, which allows us to keep the approximation of the mesh when it deforms.

7.2 Retargeting a controller close to the mesh

Having defined similar approximations for the source’s and target’s torsos, the goal is now to use the relative position of the effectors to the source torso in order to adjust the position of the retargeted effectors, whenever they come close to (or inside) the mesh. This process is illustrated in Figure 12.

First, given an effector position P , we want to identify which cone this point is “in front of”. As shown on Fig. 12.1, this is ambiguous to answer when the character is in a certain pose. To solve this ambiguity, we convert this position P into a corresponding position Q when the character is in T-pose. Therefore, we compute virtual “skinning weights” for the point P , so that we can invert the linear-blend skinning (LBS) to obtain Q . Those skinning weights w are defined using the inverse of the distances between P and the closest point on each circle: $w = \text{softmax}\left(\left\{\frac{1}{\|P - P_i\|}\right\}_{0 \leq i \leq k}\right)$, where P_i is the point closest to P on the circle of the i -th slice.

Then, we compute the position of Q such that, if it were skinned to the circles (which are themselves skinned to the bones) with the weights w , it would end up at the position P .

Now that we have a position for Q where the character’s spines are straightened, we can project Q on the axis $\vec{d}$ (between first and last bones), and identify which cone this projection lands in. Then, we build a **coordinate system** C whose position is the projection of Q on that **cone**, and whose orientation is defined by the normal and tangent vectors to the cone at that point. We store the position of Q in that coordinate system: $Q|_C$.

Finally, to obtain the corresponding point on the target, we identify the corresponding **coordinate system** C' on the corresponding cone. We compute the matching position Q' such that $Q'|_{C'} = Q|_C$.



Figure 14: Results of our collision refinement algorithm. Left to right: source pose, Mixamo's retargeting, Ours (rig-retargeting only), Ours (collision refinement)



Figure 15: Motion where rig-space retargeting helps preserve the semantic contact between the hands. Left to right: source, Mixamo, MIRRORED-Anims.

Once again, we compute virtual skinning weights w' using inverse distances: $w' = \text{softmax} \left(\left\{ \frac{1}{\|Q' - Q'_i\|} \right\}_{0 \leq i \leq k} \right)$, where Q'_i is the point closest to Q' on the circle of the i -th slice. Having solved for the T-pose Q' position, we use linear blend skinning to obtain the position of P' when the target mesh is in the retargeted pose.

All in all, we are able to obtain a matching world-space position P' for the original controller position P , while taking into account the source's and target's meshes. After our retargeting process, we interpolate between the current position of the target controller, and this new position P' , based on how close P is to the source mesh.

Metric	Aggregate	Value (m)
Average	mean	$6.06 \cdot 10^{-3}$
pos. err.	max	$1.32 \cdot 10^{-2}$
Worst	mean	$6.77 \cdot 10^{-2}$
pos. err.	max	$1.50 \cdot 10^{-1}$

Table 1: Results of the rig inversion algorithm (average and worst joint position error across all joints of the animation), aggregated over all animations in the dataset.

Network	Dataset	MSE (↓)	Pen.% (↓)
R ² ET	Mixamo	0.447	9.32
R ² ET	omaxiM	0.360	8.24

Table 2: Comparison between the standard Mixamo dataset, and our omaxiM dataset, both used to train the same network from R²ET [Zhang et al. 2023] (skeleton-aware part only).

8 Results

8.1 Rig inversion

We evaluated the accuracy of our rig inversion algorithm by measuring the average error in joint position across the 30 motions of SMTNet's test set [Zhang et al. 2024a], and report it in Table 1. Since the rig is over-constrained in some places (fingers, spine), it is not possible to reach a perfect accuracy; however, our rig inversion method still manages to reach a high level of fidelity with respect to the FK motion.

8.2 Improved Dataset for Learning-based Retargeting

We used MIRRORED-Anims to retarget the entire Mixamo database [Adobe 2024] (2446 motions) to 40 different characters, in order to create omaxiM: a more extensive, open-source motion retargeting dataset. Then, we trained the skeleton-aware network proposed by R²ET [Zhang et al. 2023], and compare it to a training performed only on the standard Mixamo dataset. We report, in Table 2, the performance of both trainings on unseen characters and motions from the NKN's test set [Villegas et al. 2018].

This experiment demonstrates that the generalization abilities of the network are closely tied to the diversity of characters in the training dataset, highlighting the impact that a larger academic dataset could have on the field of retargeting.

8.3 Quantitative comparisons

The main motivation for our method was to tackle the lack of high-quality data in learning-based methods, so a direct comparison between our retargeting procedure and the results of learning-based methods is not the primary focus of this work. Nonetheless, to provide quantitative context and demonstrate the quality of our generated data, we report evaluations against state-of-the-art

Method	MSE ($\downarrow$)	MSE ^{lc} ($\downarrow$)	Pen.% ($\downarrow$)
Source	–	–	4.43
GT	–	–	9.06
Copy Rotations	0.005	0.005	9.03
NKN [Villegas et al. 2018]	0.326	0.231	8.71
SAN [Aberman et al. 2020]	0.435	0.255	9.74
R ² ET [Zhang et al. 2023]	0.499	0.496	7.62
SMTNet [Zhang et al. 2024a]	0.284	0.229	3.50
Ours	0.008	0.002	3.04
Ours (no coll.)	0.008	0.002	3.06

Table 3: Quantitative comparison to several state-of-the-art method. Ours (no coll.) denotes our rig-space retargeting without the collision refinement.

learning-based methods [Aberman et al. 2020; Lim et al. 2019; Villegas et al. 2018; Zhang et al. 2023] in Table 3. For this, we used the test set introduced by SMTNet [Zhang et al. 2024a], which consists of 30 clean Mixamo motions performed by 3 characters, retargeted onto 3 different characters.

To evaluate closeness to the Mixamo “ground-truth”, we used the mean squared error of the joint positions in world-space (MSE), as well as the mean-squared error with respect to the root bone’s position (MSE^{lc}). We also evaluated penetration using the proportion of vertices that penetrate with another part of the mesh. Our method reaches results which are far closer to the outputs of Mixamo than learning-based methods.

8.4 Qualitative comparison

We offer several visual comparisons of our method’s results compared to Mixamo’s proprietary algorithm, which is currently used as “ground truth” in all the learning-based methods.

For one, our method allows for more precise hand contacts, as demonstrated by Figures 17 and 18: retargeting the cartwheel motion on a character with longer legs and shorter arms automatically causes a lower center of mass during the “upside-down” portion of the motion, with the hands touching the floor. We obtain better hand-floor contacts, while matching Mixamo’s level of quality on foot-floor contacts and foot sliding. Our method also reaches better results on hand-hand interactions, since the controller’s position is computed in the shared coordinate system of the spine, as demonstrated on Figures 15 and 19. The control rig proves useful for poses like that shown in Figure 20: having the heel of the foot precisely on the ground, or the palm of the hand on the floor, would have been unattainable with previous methods. Moreover, thanks to our approximation of the skin mesh, we can refine limbs that overlap with the torso. Figure 14 demonstrates the importance of this approximation when retargeting onto characters with a different body shape. Additional explanations as to why our rig-space retargeting performs better than Mixamo can be found in the Supp. Mat.

We performed a double-blind user study to estimate human preference on SMTNet’s test set [Zhang et al. 2024a], retargeted onto ten different characters. For each sample, we asked 30 people to rank the results of Mixamo, R²ET [Zhang et al. 2023], SAME [Lee et al. 2023], and MIRRORED-Anims, from 1 (preferred output) to 4 (worst output). We obtain 660 individual answers, and report the

Method	Mixamo	R ² ET [2023]	SAME [2023]	Ours
Rank ($\downarrow$)	2.04	2.41	3.54	2.00

Table 4: Average rank of each method in our user study.

mean rank of each method in Table 4. This study demonstrates that MIRRORED-Anims often reaches the same level of quality as Mixamo (48% of answers tied Mixamo and MIRRORED-Anims), and fixes a lot of Mixamo’s defects (27% of answers strictly favored MIRRORED-Anims to Mixamo).

Finally, an experiment demonstrates that manually cleaning up eventual retargeting errors is faster for MIRRORED-Anims than for Mixamo, and in both cases, even faster when using rig controllers rather than editing the “deform bones”. See Supp. Mat. for more details.

8.5 Retargeting SMPL datasets

Two limitations of the Mixamo dataset, mentioned in Section 1, are the small numbers of motions and characters it contains. Our method could help increase the diversity of motions and character proportions used to train retargeting tasks, by tapping into the available motion capture datasets for the SMPL character [Loper et al. 2015]. Our method allows to retarget any of these motions onto any other characters, even out-of-distribution ones (having extreme, even unrealistic body proportions). We can also retarget these motions onto the cartoon characters from Mixamo. To demonstrate this, we applied our method to a motion from the Moyo dataset [Tripathi et al. 2023]. The SMPL body model is famously difficult to retarget onto artist-made characters, due to the non-conventional placement of its bones. However, our universal control rig allows for a high-level abstraction which circumvents those issues, as shown on Figure 16.

9 Discussion, Limitations and Future work

We proposed a motion retargeting method that relies on the adaptation of a standardized control rig on both the source and target characters. Once expressed as an animation of the rig parameter values, the input animation is easily transferred to the target rig. The latter is finally corrected to avoid any unwanted interpenetration due to morphological change, and to restore meaningful contacts. However, we noted a few limitations:

First, the method may fail on some motions where the driving controllers cannot be properly identified by our method, that is, when driving controllers do not follow our assumptions described in Section 6.1 (for instance, a character doing pull-ups). A hybrid method could use a learning-based approach to identify the driving controllers.

Second, even though the results are aesthetically pleasing and offer a high degree of fidelity for foot contacts and self-collisions, it is worth noting that the method does not strictly ensure physical realism. To this end, a reinforcement-learning method [Reda et al. 2023] could be used with the results of MIRRORED-Anims, which would improve physical accuracy.

Moreover, our retargeting method is not always able to preserve all semantically meaningful contacts in a pose, which may be a



Figure 16: Retargeting from SMPL (left) to a Mixamo character (right).

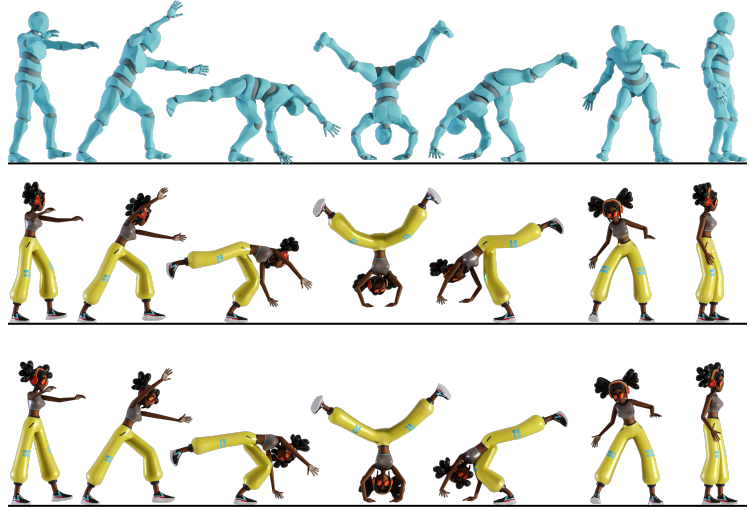


Figure 17: Comparison of our method to Mixamo. Top to bottom: source motion, Mixamo, MIRRORED-Anims. Note: on the source motion, the hands do not fully touch the floor, which explains the small offset on our output.



Figure 18: Our method is better than Mixamo at capturing hand-floor interactions. Left to right: source pose, Mixamo's retargeting, MIRRORED-Anims.



Figure 19: Our method also performs better than Mixamo for hand-hand interactions like clapping. Left to right: source pose, Mixamo's retargeting, MIRRORED-Anims.

problem when the animation includes multiple contacts, e.g. between the hands and the feet. One could also adapt the retargeting algorithm so that controllers that are close together are Another approach could be to use optimization-based retargeting, which would manage to give satisfying results [Cheynel et al. 2025]; however, these optimization-based methods work on the skeleton level and cannot optimize through the control rig layer.

Furthermore, the collision refinement method could easily be improved with ellipses instead of circles. While our current implementation only prevents penetrations of the end-effectors with the torso, a similar procedure could easily be applied to prevent inter-penetrations between limbs. Finally, the MIRRORED-Anims method could also be refined with a more complex rig setup (for instance, by using Spline Inverse Kinematics) to allow for finer

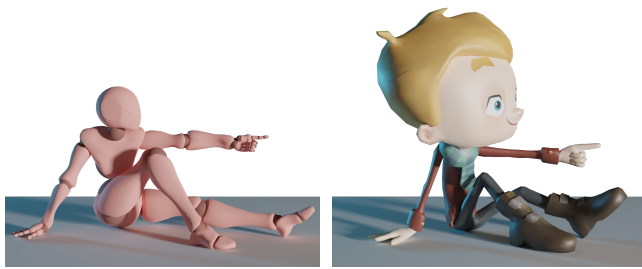


Figure 20: Example of our retargeting on a complex pose, which demonstrates the importance of the foot and finger rigs. Left: source character, right: target character.

control of some parts of the character, or by integrating ready-made mechanisms found in industry-standard rigging tools like Rigify [Community 2025].

10 Conclusion

MIRRORED-Anims is, to the best of our knowledge, the first method to build on industry-grade rigging techniques to perform motion retargeting in rig-space. Retargeting is often a quality bottleneck in motion-related tasks, from human pose estimation to robotics, and learning-based methods seem to be the most promising way to capture a fine understanding of motion semantics. This can only happen through large, high-quality motion datasets. MIRRORED-Anims is a step towards building these large datasets, and we see it as a stepping stone to steer future academic retargeting works towards improved sources of data. We hope to see future work apply our retargeting method to motion capture datasets with a wide diversity of tasks and styles.

References

- Kfir Aberman, Peizhuo Li, Dani Lischinski, Olga Sorkine-Hornung, Daniel Cohen-Or, and Baoquan Chen. 2020. Skeleton-Aware Networks for Deep Motion Retargeting. *ACM Transactions on Graphics (TOG), Proc. SIGGRAPH* 39, 4 (2020), 62.
- Adobe. 2024. Mixamo. <https://www.mixamo.com/>. Accessed: 2025-03-03.
- Jean Basset, Stefanie Wuhler, Edmond Boyer, and Franck Multon. 2020. Contact preserving shape transfer: Retargeting motion from one shape to another. *Computers & Graphics* 89 (2020), 11–23. doi:10.1016/j.cag.2020.04.002
- Alexander Berezhnyak. 2016. IK Rig: Moving Forward. (2016). <https://gdcvault.com/play/1023279/GDC>.
- Théo Cheynet, Thomas Rossi, Baptiste Bellot-Gurlet, Damien Rohmer, and Marie-Paule Cani. 2025. ReConForM: Real-time Contact-aware Motion Retargeting for more Diverse Character Morphologies. In *Computer Graphics Forum*. Wiley Online Library, e70028.
- Kwang-Jin Choi and Hyeon-Seok Ko. 2000. Online motion retargeting. *The Journal of Visualization and Computer Animation* 11, 5 (2000), 223–235.
- Blender Online Community. 2025. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam. <http://www.blender.org>
- Mathias Delahaye, Bruno Herbelin, and Ronan Boulic. 2023. Real-time self-contact retargeting of avatars down to finger level. (2023).
- Michael Gleicher. 1998. Retargeting motion to new characters. In *Proc. ACM SIGGRAPH*. 33–42.
- Stephen Gustafson, Aaron Lo, and Paul Kanyuk. 2020. Analytically Learning an Inverse Rig Mapping. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Talks*. 1–2.
- Chris Hecker, Bernd Raabe, Ryan W. Enslow, John DeWeese, Jordan Maynard, and Kees van Prooijen. 2008. Real-time motion retargeting to highly varied user-created morphologies. In *ACM SIGGRAPH 2008 Papers* (Los Angeles, California) (SIGGRAPH '08). Association for Computing Machinery, New York, NY, USA, Article 27, 11 pages. doi:10.1145/1399504.1360626
- Edmond Ho, Taku Komura, and Chiew-Lan Tai. 2010. Spatial Relationship Preserving Character Motion Adaptation. *ACM Trans. Graph.* 29 (07 2010). doi:10.1145/1833351.1778770
- Daniel Holden, Jun Saito, and Taku Komura. 2015. Learning an inverse rig mapping for character animation. In *Proceedings of the 14th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 165–173.
- Daniel Holden, Jun Saito, and Taku Komura. 2017. Learning Inverse Rig Mappings by Nonlinear Regression. *IEEE Transactions on Visualization and Computer Graphics* 23, 3 (2017), 1167–1178. doi:10.1109/TVCG.2016.2628036
- Lei Hu, Zihao Zhang, Chongyang Zhong, Boyuan Jiang, and Shihong Xia. 2023. Pose-Aware Attention Network for Flexible Motion Retargeting by Body Part. *IEEE Transactions on Visualization and Computer Graphics* (2023), 1–17. doi:10.1109/tvcg.2023.3277918
- Taeil Jin, Meekyung Kim, and Sung-Hee Lee. 2017. Motion Retargeting to Preserve Spatial Relationship between Skinned Characters. In *Symposium on Computer Animation (SCA)*. Article 25, 2 pages. doi:10.1145/3099564.3106647
- Richard Kulpa, Franck Multon, and Bruno Arnaldi. 2005. Morphology-independent representation of motions for interactive human-like animation. In *Eurographics*.
- Jehee Lee and Sung Yong Shin. 1999. A hierarchical approach to interactive motion editing for human-like figures. In *Proc. ACM SIGGRAPH*. 39–48.
- Summin Lee, Taeho Kang, Jungnam Park, Jehee Lee, and Jungdam Won. 2023. SAME: Skeleton-Agnostic Motion Embedding for Character Animation. (2023). doi:10.1145/3610548.3618206
- Jongin Lim, Hyung Jin Chang, and Jin Young Choi. 2019. PMnet: Learning of Disentangled Pose and Movement for Unsupervised Motion Retargeting. In *British Machine Vision Conference (BMVC)*.
- Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. 2015. SMPL: A Skinned Multi-Person Linear Model. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)* 34, 6 (2015), 248:1–248:16.
- Mathieu Marquis Bolduc and Hau Nghiep Phan. 2022. Rig Inversion by Training a Differentiable Rig Function. In *SIGGRAPH Asia 2022 Technical Communications* (Daegu, Republic of Korea) (SA '22). Association for Computing Machinery, New York, NY, USA, Article 15, 4 pages. doi:10.1145/3550340.3564218
- Eray Molla, Henrique Galvan Debarba, and Ronan Boulic. 2017. Egocentric mapping of body surface constraints. *IEEE transactions on visualization and computer graphics* 24, 7 (2017), 2089–2102.
- Jean-Sébastien Monzani, Paolo Baerlocher, Ronan Boulic, and Daniel Thalmann. 2000. Using an intermediate skeleton and inverse kinematics for motion retargeting. In *Computer Graphics Forum*, Vol. 19. Wiley Online Library, 11–19.
- Stevó Racković, Cláudia Soares, Dušan Jakovetić, and Zoranka Desnica. 2024. A majorization–minimization-based method for nonconvex inverse rig problems in facial animation: algorithm derivation. *Optimization Letters* 18, 2 (2024), 545–559.
- Stevó Racković, Cláudia Soares, and Dušan Jakovetić. 2023. Distributed Solution of the Inverse Rig Problem in Blendshape Facial Animation. *arXiv:2303.06370 [cs.CV]* <https://arxiv.org/abs/2303.06370>
- Daniele Reda, Jungdam Won, Yuting Ye, Michiel Panne, and Alexander Winkler. 2023. Physics-based Motion Retargeting from Sparse Inputs. *SCA, Proceedings of the ACM on Computer Graphics and Interactive Techniques* 6 (08 2023), 1–19. doi:10.1145/3606928
- Seyoon Tak and Hyeon-Seok Ko. 2005. A physically-based motion retargeting filter. *ACM Transactions on Graphics* 24, 1 (2005), 98–117.
- Shashank Tripathi, Lea Müller, Chun-Hao P. Huang, Taheri Omid, Michael J. Black, and Dimitrios Tzionas. 2023. 3D Human Pose Estimation via Intuitive Physics. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 4713–4725. <https://ipman.is.tue.mpg.de>
- Ruben Villegas, Duygu Ceylan, Aaron Hertzmann, Jimei Yang, and Jun Saito. 2021. Contact-aware retargeting of skinned motion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 9720–9729.
- Ruben Villegas, Jimei Yang, Duygu Ceylan, and Honglak Lee. 2018. Neural Kinematic Networks for Unsupervised Motion Retargeting. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Yashuai Yan, Esteve Valls Mascaro, and Donghui Lee. 2023. ImitationNet: Unsupervised Human-to-Robot Motion Retargeting via Shared Latent Space. In *2023 IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids)*. 1–8. doi:10.1109/Humanoids57100.2023.10375150
- Zijie Ye, Jia-Wei Liu, Jia Jia, Shikun Sun, and Mike Zheng Shou. 2024. Skinned motion retargeting with dense geometric interaction perception. *Advances in Neural Information Processing Systems* 37 (2024), 125907–125934.
- Haodong Zhang, Zhike Chen, Haocheng Xu, Lei Hao, Xiaofei Wu, Songcen Xu, Zhen-song Zhang, Yue Wang, and Rong Xiong. 2024a. Semantics-aware motion retargeting with vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2155–2164.
- Jiaxu Zhang, Zhigang Tu, Junwu Weng, Junsong Yuan, and Bo Du. 2024b. A Modular Neural Motion Retargeting System Decoupling Skeleton and Shape Perception. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 46, 10 (2024), 6889–6904. doi:10.1109/TPAMI.2024.3386777
- Jiaxu Zhang, Junwu Weng, Di Kang, Fang Zhao, Shaoli Huang, Xuefei Zhe, Linchao Bao, Ying Shan, Jue Wang, and Zhigang Tu. 2023. Skinned Motion Retargeting With Residual Perception of Motion Semantics & Geometry. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 13864–13872.