EUROGRAPHICS 2025 / A. Bousseau and A. Dai
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 44 (2025), Number 2

Supplementary Material – ReConForM : Real-time Contact-aware Motion Retargeting for more Diverse Character Morphologies

T. Cheynel[1,2] and T. Rossi[1] and B. Bellot-Gurlet[1] and D. Rohmer[2] and M.P. Cani[2]

[1]Kinetix
[2]LIX, École Polytechnique, CNRS, IP Paris

1. Related work

We propose in Table 1 a comparative summary of the features provided by the previous works regarding the constraints (avoiding mesh penetration, preserving foot contacts, handling multiple characters, adaptation to diverse and unseen morphologies), and their computational efficiency.

2. Validation set

Table 4 itemizes the contents of our validation set, with all animations taken from Mixamo [Ado24]. The source character is either "XBot" (the red character with a feminine morphology) or "YBot" (the blue character with a masculine morphology), as those seem to be the original characters on which the motion was designed, and thus are the best available quality for those motions. The column "Difficulty" corresponds to the difficulty of the motion considering mesh contacts, as explained in the main article (Sec. 5.2.1).

3. Key-vertices transfer

We propose a quantitative evaluation of the accuracy of our key-vertices transfer algorithm. For each of our 10 Mixamo characters, we asked a 3D character expert to manually transfer the key-vertices, and we measured the average distance between those key-vertices and the ones automatically produced by our method. On average, we report an error of 6.46 cm, while the average edge length of these meshes is 3.72 cm.

As mentioned in the main article, we performed an ablation study with regards to the number of key-vertices. For this, we used $N = 96$ vertices, in order to represent the mesh of the characters in finer detail. The location of these vertices, as well as the results of the transfer algorithm on several characters, are illustrated in Figure 1.

4. Hyperparameters

We report hyperparameters used in the fine-tuned version of our method. We use a learning rate of 0.01 for the optimization process, and set the number N of keypoints to 41 on the template, source and target meshes. Table 2 reports the weights of main loss terms w_{reg}, w_{smooth} and w_{sem}, along with additional ponderations for each term within the semantic loss (resp. distance, direction, penetration, height, and sliding) $w_{dist}, w_{dir}, w_{pen}, w_{height}$, and $w_{sliding}$. Finally, we provide values for characteristic distances used in the computation of the weighting matrices (see section 4.1): $d_{min}, d_{max}, h_{min}, h_{max}$. We empirically found that this set of hyperparameters works well in most of the cases, although they can easily be refined thanks to their reduced amount and explicit purpose in the method.

5. Metrics

Figure 9 shows the distribution of the values of each metric across all animations of our validation dataset. This allows us to obtain finer information than the values reported in the table of the main article.

6. Robustness

We provide several additional results that demonstrate the robustness of our method.

6.1. Initialization

As explained in the main article, we initialize the target character with the "copy rotations" method, before optimizing the poses. To evaluate the robustness of our method with respect to the initialization, we perform the following experiment: we add uniform random noise to the poses obtained from the "copy rotations" method, and measure the average difference in joint positions. Table 3 shows the results of our experiment (performed on the motions of character "Sophie"). While this experiment shows some degree of robustness against initialization, it is important to mention that a proper initialization makes the results less prone to the loss of semantic information during the optimization process. It also helps the optimization process converge faster.

6.2. Individual impact of the losses

We demonstrate the usefulness of each component of the semantic loss $\mathcal{L}_{\text{sem}}$ by performing an ablation study. For each component, we

Figure 1: *Results of the key vertices transfer algorithm ($N = 96$). Top-left character shows the vertices locations on the template mesh [LMR*15], top row shows characters from Mixamo [Ado24], while bottom row shows additional, production-ready characters from Blender Studio [Stu24]. Key vertices are shown with corresponding colored spheres to show the automatic transfer to the different meshes.*

show its impact on a selected case that highlights specific aspects of a pose or motion.

6.2.1. Distance and direction losses

Figure 5 shows the impact of $\mathcal{L}_{\text{dist}}$ and $\mathcal{L}_{\text{dir}}$, and explains the need for both in our optimization:

- In Figure 2b, neither $\mathcal{L}_{\text{dist}}$ nor $\mathcal{L}_{\text{dir}}$ are active. Because the target character has longer forearms, the hands do not meet in the middle, leading to a considerable loss of motion semantics.
- In Figure 2c, only $\mathcal{L}_{\text{dist}}$ is active. The distance between the two palms is respected (a few centimeters), but the hands are not in a similar pose, because of their respective position at the initialization of our optimization.
- In Figure 2d, both $\mathcal{L}_{\text{dist}}$ and $\mathcal{L}_{\text{dir}}$ are active. This time, the hands of the target character are placed in order to respect both the distance (a few centimeters) and direction (right hand forward) present on the source pose. The arms are open wider, to compensate for the target character having longer forearms.

6.2.2. Penetration loss

Figure 6 shows the impact of the penetration loss. With all other losses active (Figure 3b), the left elbow is colliding with the body. Adding the penetration loss (Figure 3c) greatly reduces collisions.

6.2.3. Height and foot sliding losses

Figure 7 depicts our experiment to evaluate the impact of $\mathcal{L}_{\text{height}}$ and $\mathcal{L}_{\text{sliding}}$ on the foot contacts:

- Without those losses (Figure 4a) we observe an offset to the ground, of an amount that varies during the contact. It is due to the difference of bone lengths along the kinematic chain.
- In response to this, the height loss (Figure 4b) forces the foot to touch the ground during the entire duration of the contact. However, this causes horizontal sliding during the contact period.
- The foot sliding loss (Figure 4a) mitigates this issue by finding a compromise between the two constraints, producing cleaner contacts with very little foot skating.

Method	Approach	Focus on mesh penetration	Focus on foot contacts	Temporal continuity	Multi char.	Can adapt to diverse characters	Generalizes to unseen morphologies	Speed
NKN [VYCL18]	Self-supervised	✗	✗	✓	✗	✗ (fixed number of joints)	✓	unreported
PMNet [LCC19]	Self-supervised	✗	✗	✓	✗	✗ (fixed number of joints)	✓	unreported
SAN [ALL*20]	Self-supervised	✗	✓	✓	✗	✓ (common primal skeleton)	✓	unreported
R^{2}ET [ZWK*23]	Self-supervised	✓	✗	✓	✗	✗ (fixed number of joints)	✓	∼120 fps
Villegas et al. [VCH*21]	Supervised + Optimization	✓	✓	✓	✗	✓	✗ (needs retraining for each new char.)	unreported
Gleicher [Gle98]	Optimization	✗	✓	✓	✓	✗ (only tested on one character)	✓	∼5 fps (need to manually identify constraints)
AuraMesh [JKL17]	Optimization	✓	✓	✗	✓	✗ (only for SMPL [LMR*15])	✓	∼1 fps
Basset et al. [BWBM20]	Optimization	✓	✓	✗	✗	✗ (only for SMPL [LMR*15])	✓	∼0.003 fps
Ho et al. [HKT10]	Optimization	✗ (only to avoid collisions)	✓	✓	✓	✗ (same skeleton structure)	✓	∼1.7 fps
PAN [HZZ*23]	Unsupervised Learning	✗	✓	✓	✓	✗ (common primal skeleton)	✗ (needs retraining for each new char.)	unreported
Zhang et al. [ZGY*23]	Reinforcement Learning	✓	✓	✓	✓	✗ (only one type of character)	✗ (needs retraining for each new char.)	(needs retraining for each new motion)
Reda et al. [RWY*23]	Reinforcement Learning	✓	✓	✓	✓	✗ (only one type of character)	✗ (needs retraining for each new char.)	∼36 fps
Ours	Optimization	✓	✓	✓	✓	✓	✓	∼67 fps

Table 1: *Comparison of existing retargeting methods*

(a) *Source pose (XBot - Plotting)* **(b)** *No distance loss, no direction loss* **(c)** *Distance loss, no direction loss* **(d)** *Distance loss, direction loss*

Figure 2: *Visualizations of the impact of $\mathcal{L}_{dist}$ and $\mathcal{L}_{dir}$.*

(a) *Source pose (YBot - Arm Stretching)* **(b)** *All losses active, except* $\mathcal{L}_{pen}$ **(c)** *All losses active, including* $\mathcal{L}_{pen}$

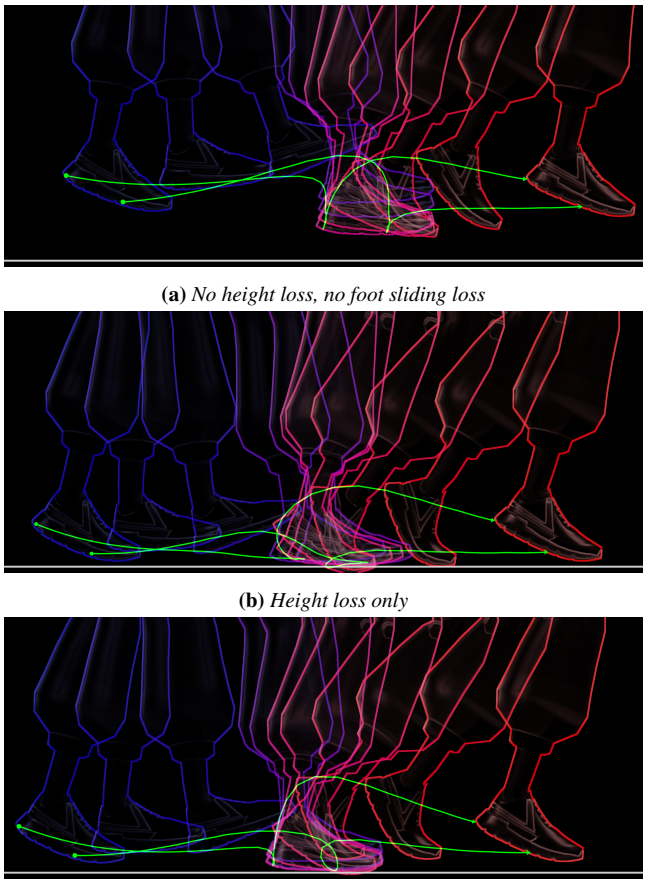
Figure 3: *Visualizations of the impact of* $\mathcal{L}_{pen}$

Table 2: *Hyperparameters used in our method*

Hyperparameter	**Value**
d_{min}	5% of character height
d_{max}	15% of character height
h_{min}	5% of character height
h_{max}	15% of character height
w_{reg}	$1 \cdot 10^{-2}$
w_{smooth}	$1 \cdot 10^{-4}$
w_{sem}	1
w_{dist}	1
w_{dir}	0.5
w_{pen}	10.0
w_{height}	1
$w_{sliding}$	0.5
lr	$1 \cdot 10^{-2}$
N	41 (or 96 when explicitly mentioned)

Range of uniform noise	Average joint position difference
$\pm$ 5 degrees	$1.2 \cdot 10^{-2}$ m
$\pm$ 20 degrees	$3.2 \cdot 10^{-2}$ m

Table 3: *Impact of random noise after initialization*



(a) *No height loss, no foot sliding loss*

(b) *Height loss only*

(c) *Height loss and foot sliding loss*

Figure 4: *Visualizations of the impact of the losses on foot contact. Overlay of the position of the foot at every third frame (blue: past, red: future). The trajectory of two vertices from the sole and heel plotted in green. Ground floor is represented by the white horizontal line.*

7. Failure cases

In the following paragraphs, we discuss several failure cases regarding the key-vertices transfer and regarding the retargeting process, and identify possible solutions to fix them.

7.1. Key-vertices transfer

7.1.1. Lack of geometry

The transfer of key vertices can yield suboptimal results if the target mesh contains too few vertices, such as the blocky characters from Minecraft [Moj24] depicted in Figure 5a. In that case, there are not enough vertices to accurately model the overall shape of the body parts, which prevents the method from placing key vertices in relevant places. A simple solution consists in subdividing the mesh to increase the number of vertices. Additionally, this issue could be automatically detected based on simple rules like face area and number of vertices.

We notice that the key vertices on the higher-resolution mesh are better located, as seen on Figure 5b, but there are still some errors : for instance, the chin vertex and eye vertex are too high compared to the location of the chin and eyes in the texture. This cannot be taken into account based on geometry alone.

7.1.2. Asymmetry

Figure 5c shows the impact of an asymmetric mesh. Due to the presence of a satchel on the side of the character, the key-vertices are pulled in to the right, which could deteriorate the pertinence of the semantic loss, and lead to contacts in the wrong places. In particular, the key-vertex "hips-front" (green) ends up closer to the right thigh. This can be corrected by manually removing the satchel that makes the character asymmetric, as shown in Figure 5d.

7.1.3. Accessories/Props

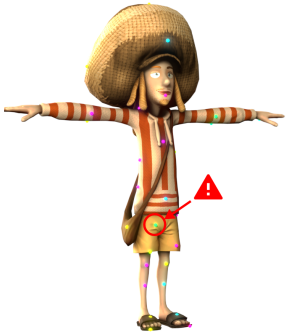
The key-vertices transfer algorithm also tends to be affected by the presence of additional props in the target character, even when they are symmetrical. For instance, some cartoon characters come with a large hat/ haircut, or accessories, like those seen in Figures 5d, 5e and 5f. In that case, the key-vertices of the head are likely to be placed on the most extreme points. As an example, on Figure 5e, the "ear" vertices are misplaced, on Figure 5d, the "eye" vertex is misplaced, and on Figure 5f, the "chin" vertex is misplaced. This may have a positive impact on some motions, as the optimization will prevent collisions with the hair or cigar. Nevertheless, it can also lead to a loss of semantics because contact might occur at the wrong position on the mesh in some motions.

7.1.4. Overlapping meshes

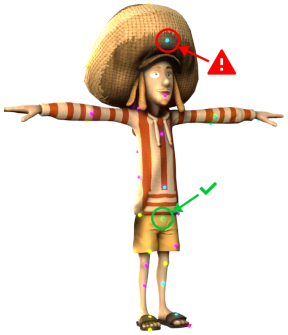
Some characters have overlapping meshes (for instance, the top of their pants being hidden behind their top). In that case, the transfer algorithm might place the vertices on "hidden" vertices, causing small amounts of self-penetration as the outer mesh is neglected. To alleviate this issue, a simple filtering of those vertices from the mathematical formulation would be sufficient.

(a) *A low-res character from Minecraft [Moj24] with only 84 vertices.*

(b) *Higher-resolution version of the same character, with 1644 vertices.*



(c) *Asymmetric character*

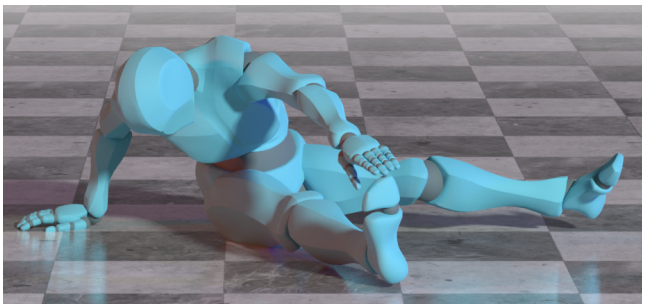


(d) *Nearly-symmetric character*

(e) *Large haircuts*

(f) *Accessories*

Figure 5: *Examples of failure cases of the key-vertices transfer algorithm*



(a) *Source pose*

(b) *"Copy Rotations" (for reference)*

(c) *Output of our method*

Figure 6: *Failure case on a complex pose with several contacts.*

7.2. Retargeting

7.2.1. Complex poses

In very complex poses, with several points of contact, it becomes quite difficult for the optimization to find a compromise between the various semantic aspects of the motion. Figure 6 shows an examples of a complex failure case : the floor contacts (left hand, both feet), that were lost by "copy rotations" (Figure 6b) are then restored by our method (Figure 6c), but the foot-hand contact is lost.

7.2.2. Unrealistic motion trajectories

Our optimization considers all frames almost independently, apart from $\mathcal{L}_{\text{smooth}}$. When the retargeting imposes a considerable change of position on a portion of the animation, there are no guarantee that it will respect the laws of physics. This is shown on Figure 8 : to prevent the head from colliding with the floor during a flip, the character's position is shifted vertically, making the trajectory look less like a parabola. In cases like this, the floor penetration loss leads to a decrease in motion plausibility. An additional term taking into account the global trajectory of the center of mass could be used to alleviate this issue.

7.2.3. Mesh over-simplification

In cases where the key-vertices are not sufficient to capture important parts of the mesh, collisions might still occur. This can be seen on Figure 7, where the upper part of the ear penetrates the ground freely as there are no key-vertices encoding this part of the head. Increasing the number of key-vertices is likely to solves this kind of issues, as this allows for a finer encoding of the mesh.

7.3. Additional characters

We demonstrate that, unlike previous methods, ReConForM can adapt to a variety of characters, regardless of how different their skeleton structure or morphology might be. For this, we show results on five characters used in animated short films, graciously provided by Blender Studio [Stu24] under CC-BY license. The characters are: Rain, Rex, Gabby, Sprite and Phil. The result of key-vertices transfer can be seen in the main paper (for $N = 41$) and in Figure 1 (for $N = 96$). Moreover, some results of several motions retargeted on these characters are provided in the main paper.

8. User study

8.1. Details

We selected 8 characters from Mixamo for the user study, and we retargeted each of the 45 animations composing our evaluation set on a randomly chosen character, as described in Table 4. We rendered 45 videos containing the source animation on the left, and ours / R^{2}ET's [ZWK*23] animations on the right (in a random order).

We used Prolific [Pro23] to collect answers from 116 English-speaking users (users were compensated on average £10.39 per hour), and we also asked 17 animation students and professionals to participate in our survey. Each participant was asked to fill

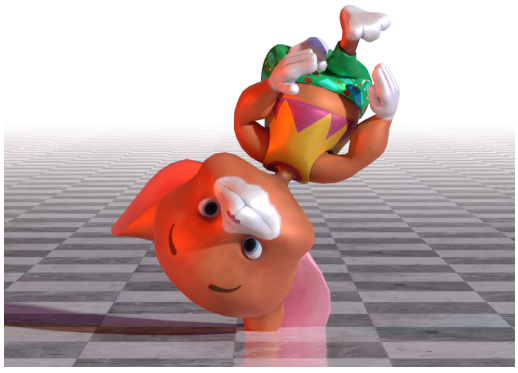
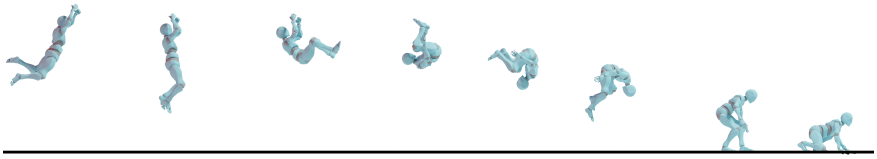
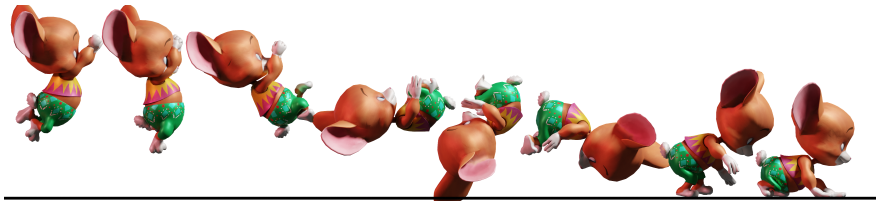


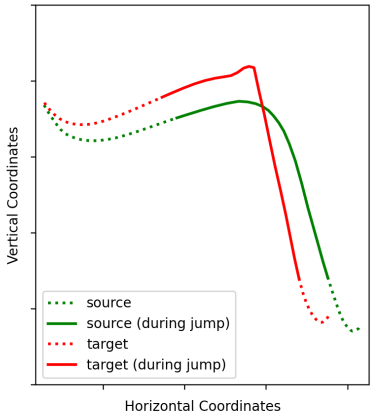
Figure 7: *Collisions can appear on parts of the mesh where no key-vertices are placed (motion : Backflip, character : Mousey)*



(a) *Chronophotography of the source motion*



(b) *Chronophotography of the retargeted motion*



(c) *Trajectory of the pelvis position for both characters*

Figure 8: *Failure case on an unrealistic trajectory (motion : Swing To Land, character : Mousey)*

a form containing 12 questions, with an additional question to test their attention and filter out inattentive users. In total, we collected a total of 1596 individual answers. Amongst all respondents:

- 1 was under 18
- 88 were aged 18-30
- 38 were aged 30-50
- 6 were over 50

We also asked them to report their familiarity with the task at hand (multiple answers allowed):

- 85 of them declared that they were “familiar with video games or interactive experiences including 3D characters”
- 17 of them declared “having used 3D animation software before”
- 7 of them declared being “a professional / student in the field of character animation”
- 36 declared having “no experience in particular” in the field of 3D animation

For each of the 12 videos, the users were tasked to answer two questions:

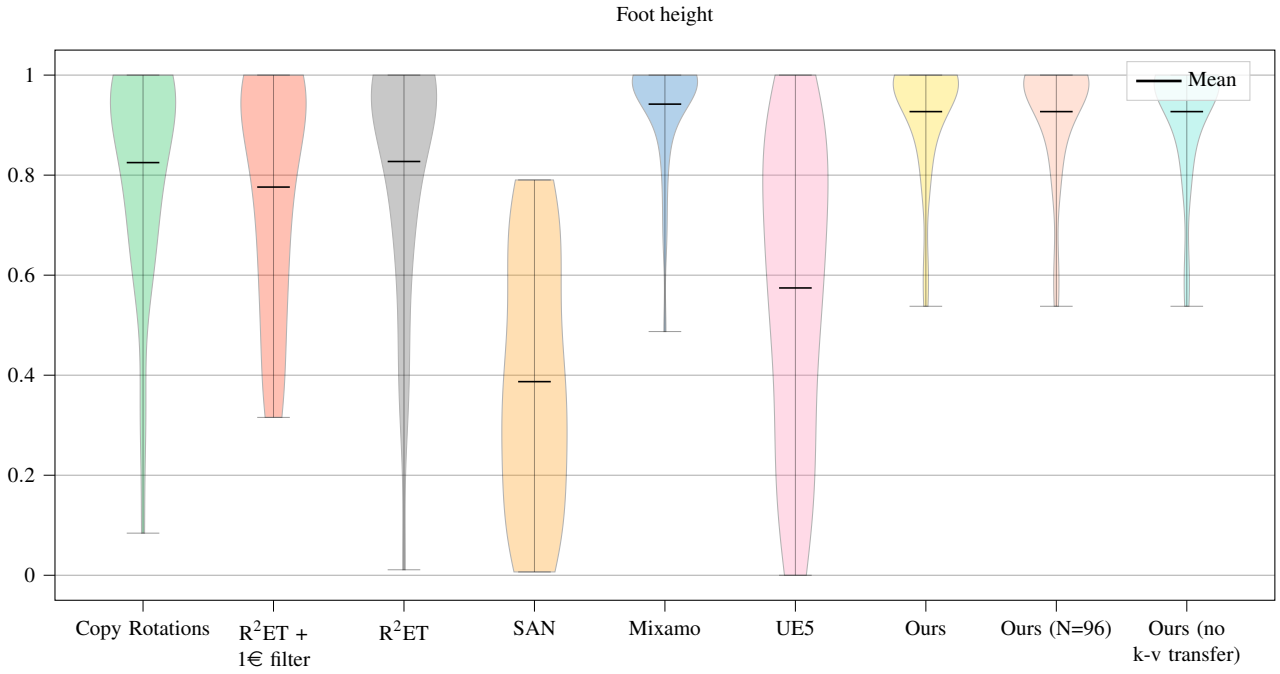
- “In the example above, with regards to fidelity to the source animation:”
- “In the example above, which output is the most visually pleasing (regardless of the source):”

The possible answers were:

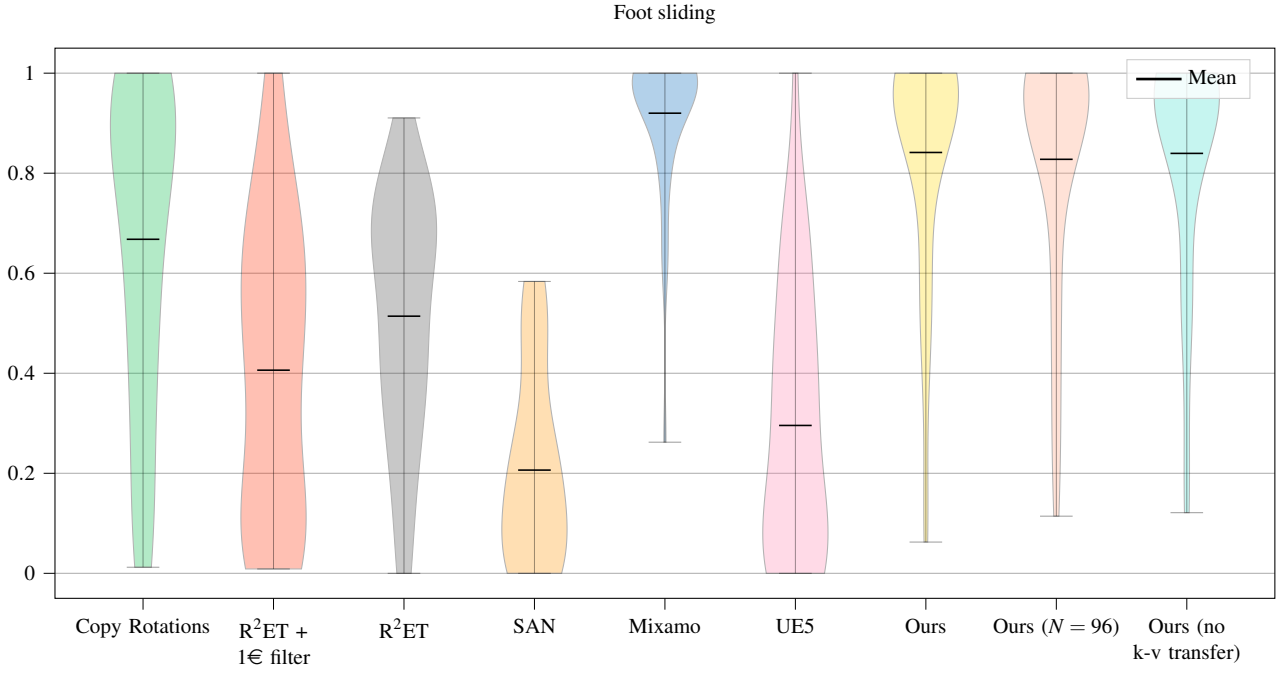
- A is better
- B is better
- A and B are tied (both good)
- A and B are tied (could both be improved)
- Not sure

8.2. Filtering

Upon reviewing the results obtained with R^2ET’s pre-trained weights, we observed a lot of jitter and flicker. After exchanging with the authors about this issue, we decided to use a OneEuro filter [CRV12] with a cutoff frequency of $f_c = 0.02$ Hz and $\beta = 0.5$, which resulted from a manual tuning in order to obtain the best results. We used this filter on R^2ET’s results for our user study, and (when explicitly mentioned) for our quantitative comparison.

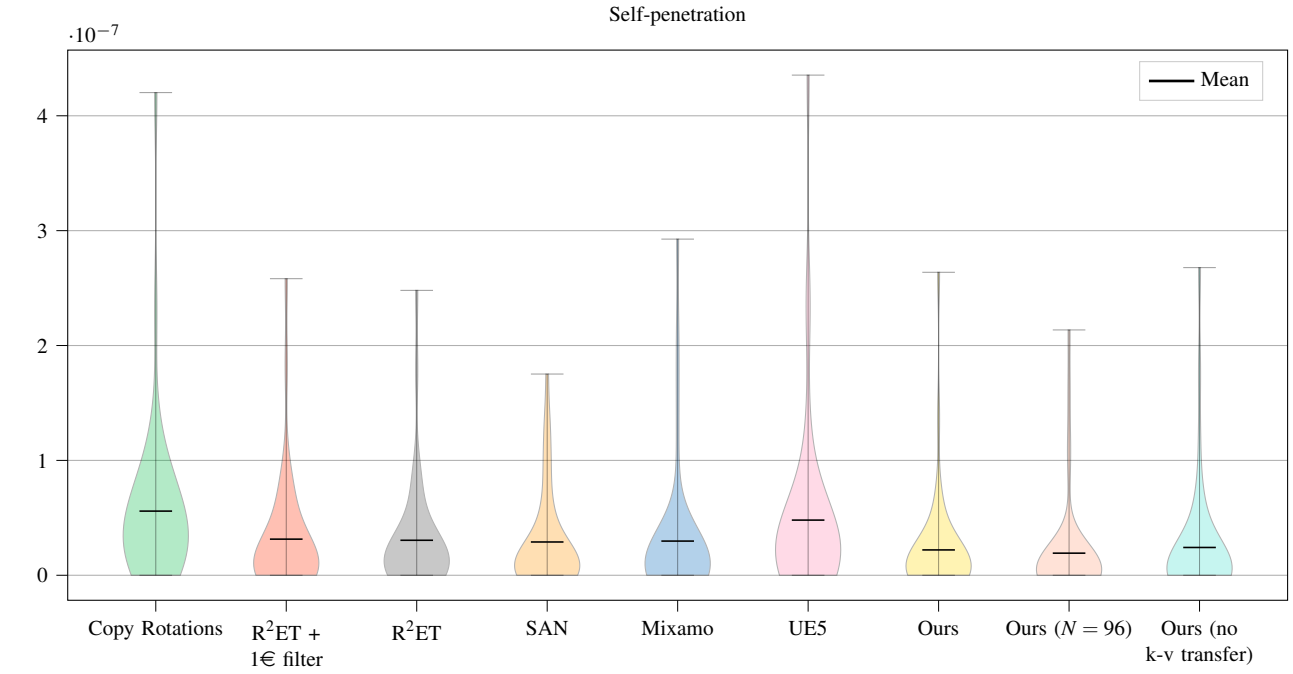


(a) *Comparison of the foot-ground contacts F-1 score (source taken as ground truth) (↑)*

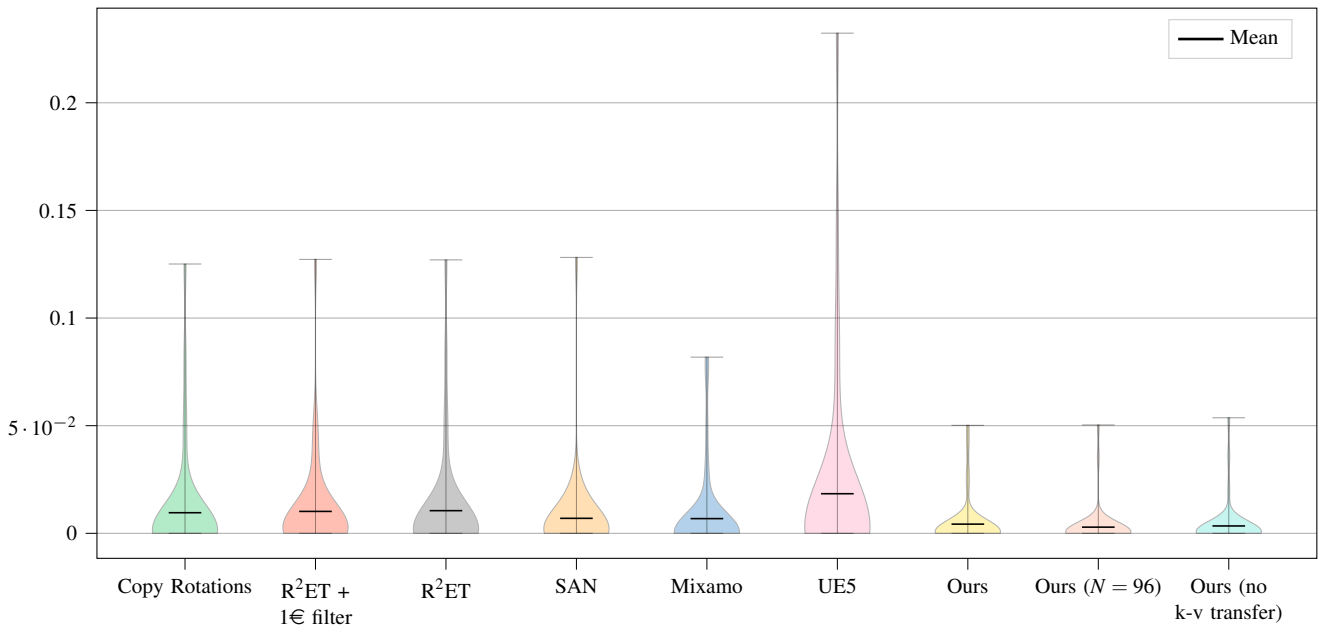


(b) *Comparison of the foot sliding F-1 score (source taken as ground truth) (↑)*

Figure 9: *Distribution of the metrics across all 45 animations of the validation dataset. For each method, the minimum, maximum and mean values across all animations are indicated by horizontal lines.*

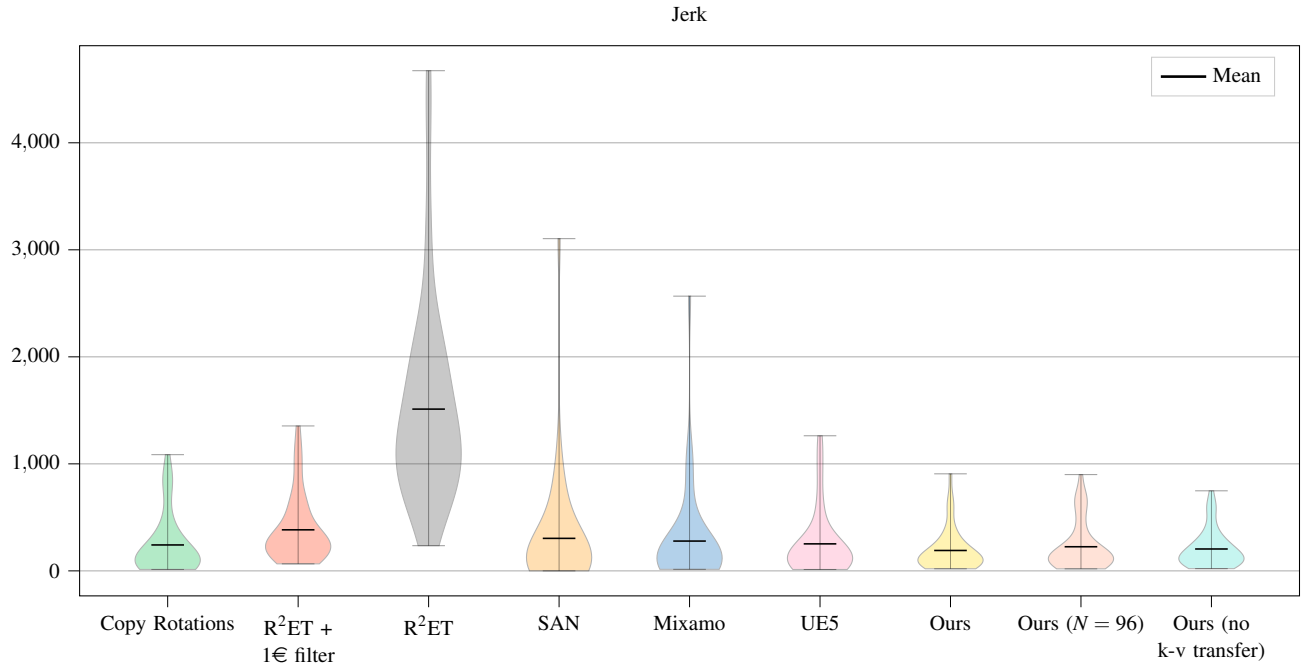


(c) *Comparison of the limb penetration volume (↓)*



(d) *Comparison of the floor penetration volume (↓)*

Figure 9: *Distribution of the metrics across all 45 animations of the validation dataset. For each method, the minimum, maximum and mean values across all animations are indicated by horizontal lines. When relevant, the metric is computed for the source animation as a reference.*



(e) *Comparison of the jerk values (↓)*

Figure 9: *Distribution of the metrics across all 45 animations of the validation dataset. For each method, the minimum, maximum and mean values across all animations are indicated by horizontal lines. When relevant, the metric is computed for the source animation as a reference.*

Animation name	Source	Difficulty	Target character in our user study
Angry	XBot	high	Michelle
Arm Stretching	YBot	high	TheBoss
Backflip	XBot	low	Mousey
Breakdance Foot to Idle	XBot	high	Ortiz
Brooklyn Uprock	XBot	high	BigVegas
Butterfly Twirl	XBot	low	SportyGranny
Chapeau de Couro	YBot	high	Michelle
Charge	YBot	low	Mousey
Combo Punch	YBot	low	TheBoss
Crazy Gesture	XBot	low	Ortiz
Cross Jumps	XBot	low	SportyGranny
Cross Jumps Rotation	YBot	low	Mousey
Crying	XBot	high	Ortiz
Dive roll	YBot	high	Michelle
Ducking	YBot	high	TheBoss
Evading A Threat	YBot	low	BigVegas
Football Stance	YBot	low	Ortiz
Forward Jump	YBot	low	Sophie
Gangnam Style	YBot	low	BigVegas
Golf Pre-Putt	XBot	high	Mousey
Golf Putt	YBot	low	Kaya
Golf Tee Up	YBot	low	SportyGranny
Hit On Side Of The Head	YBot	low	Sophie
Insult	XBot	low	SportyGranny
Kneeling	YBot	high	TheBoss
Knocked Out	XBot	high	Sophie
Looking	YBot	high	Mousey
Low Crawl	XBot	high	Michelle
PikeWalk	YBot	high	Ortiz
Plotting	XBot	low	Kaya
Push Up	XBot	high	Sophie
Shooting	YBot	low	Michelle
Shoulder Rubbing	XBot	high	Mousey
Silly Dancing	YBot	high	TheBoss
Situps	XBot	high	Michelle
Standing Arguing	XBot	low	Sophie
Standing Up	YBot	high	Kaya
Stumble Backwards	YBot	high	BigVegas
Stunned	YBot	high	SportyGranny
Swagger Walk	XBot	low	Ortiz
Talking on a Cell Phone	YBot	high	Kaya
Thankful	XBot	low	SportyGranny
Threatening	XBot	high	TheBoss
Tripping	YBot	high	Michelle
Waving	XBot	low	Kaya

Table 4: *Composition of our dataset and user study*

9. "Copy Rotations" Algorithm

We noticed that the "copy rotations" baseline, despite being commonly used in several works, does not appear to have a formal definition. Intuitively, it can be understood as a way to copy the local rotation of the bones form the source to the target. However, in practice, the formulation is more complex, as it needs to take into account the bones' initial rotation in a common pose (in our case, a T-pose).

As such, for reproducibility and completeness purposes, we propose the following formalization of the "copy rotations" algorithm, and we explain how it is implemented in ReConForM.

Definition: A **skeleton** is a tuple $\mathcal{S} = (\mathcal{M}, \mathcal{H})$, where:

- $\mathcal{M} \in \mathbb{R}^{N\times 4\times 4}$ is the world-space TRS (translation, rotation, scale) matrix of each of the N bones, in a special pose (in our case, a T-pose). In our case, the scales are unitary, as we are only interested in rotations and positions.
  Thus, the rotation matrices have the following structure:
  $$\mathcal{M}^i = \left(\begin{array}{ccc|c} & \mathcal{R}^i & & \mathcal{P}^i \\ \hline 0 & 0 & 0 & 1 \end{array}\right)$$
- $\mathcal{H} : [\![1, N-1]\!] \to [\![0, N-1]\!]$ is the "hierarchy" function, taking as input an index of a bone, and returning the index of its parent bone. The first bone is said to be the "root" bone, and has no parent ; it is the only one whose position is animated.

Definition: A **pose** is a tuple $(p, \mathbf{q})$, where:

- $p \in \mathbb{R}^3$ is the world-space position of the root bone.
- $\mathbf{q} \in \mathbb{R}^{N\times 3}$ is the rotation of each bone, in local space (i.e., relative to its parent's coordinate frame)

Definition: A **mapping** from a source skeleton $\mathcal{S}_S$ to a target skeleton $\mathcal{S}_T$, denoted as $\mathcal{M}_{S\to T}$, is a set of several pairs (b_S, b_T), where:

- $b_S \in [\![0, N_S]\!]$ is the index of a bone in the source skeleton, which has N_S bones
- $b_T \in [\![0, N_T]\!]$ is the index of the corresponding bone in the target skeleton, which has N_T bones

9.1. Simple formulation

Given a source skeleton $\mathcal{S}_S := (\mathcal{M}_S, \mathcal{H}_S)$ and a target skeleton $\mathcal{S}_T := (\mathcal{M}_T, \mathcal{H}_T)$, our goal is to transfer a pose $(p_S, \mathbf{q_S})$ from the source to the target.

9.1.1. Pelvis position

The height (in T-pose) of the root bone of the source is noted $\mathcal{M}^0_S[3,4] := (0\ 0\ 0\ 1) \cdot \mathcal{M}^0_S \cdot (0\ 0\ 1\ 0)^T$, as our coordinate system has the Z coordinate pointing upwards.

As such, we use the ratio between the source and target's heights to define $p_T = \dfrac{\mathcal{M}^0_T[3,4]}{\mathcal{M}^0_S[3,4]} p_S$

9.1.2. Bone rotations

To transfer the bone rotations, it is necessary to proceed in a specific older, so that every bone is processed after its parents, and before any of its children. This is because rotating a bone will update the world matrices of all of its successors. In order to do this, a topological sort of the target skeleton is necessary.

During the execution of our retargeting process, the rotations of the bones are updated, starting from the root bone and moving in topological order. Let us denote by $\mathcal{M}^{\text{curr}}_T \in \mathbb{R}^{N_T\times 4\times 4}$ the current world matrices of the target skeleton, during the execution. Also, let us denote by $\mathcal{M}^{\text{curr}}_S \in \mathbb{R}^{N_S\times 4\times 4}$ the world matrices of the source skeleton when the local pose $(p_S, \mathbf{q}_S)$ is applied.

When considering the next pair of bones to retarget, $(b_S, b_T) \in \mathcal{M}$, let us define $b'_S = \mathcal{H}_S(b_S)$ and $b'_T := \mathcal{H}_T(b_T)$, their respective parents. We have the following:

- $q_S^{b_S}$ is the rotation we want to transfer, expressed in local space (i.e., with respect to the parent of b_S).
- $\left(\mathcal{R}_S^{b'_S}\right)^{-1} \mathcal{R}_S^{b_S}$ is the rotation of b_S in T-pose, expressed in local space (with respect to its parent).

Knowing that, the aim is to transfer the relative rotation with respect to the T-pose, which is:

$$L_S^{b_S} := \left(\left(\mathcal{R}_S^{b'_S}\right)^{-1} \mathcal{R}_S^{b_S}\right)^{-1} q_S^{b_S}$$

or, more simply, $L_S^{b_S} := \left(\mathcal{R}_S^{b_S}\right)^{-1} \mathcal{R}_S^{b'_S} q_S^{b_S}$

To transfer this rotation to the target, we must first express it in world space:

$$W^{b_S} := \mathcal{R}_S^{\text{curr},b'_S} L_S^{b_S} \left(\mathcal{R}_S^{\text{curr},b'_S}\right)^{-1}$$

This gives us the local rotation to apply to the target bone:

$$L_T^{b_T} := \left(\mathcal{R}_T^{\text{curr},b'_T}\right)^{-1} W^{b_S} \mathcal{R}_T^{\text{curr},b'_T}.$$

"Applying" this rotation to the bone in its current state means that, *in fine*, $\mathbf{q}_T^{b_T} = \mathcal{R}_T^{b_T} L_T^{b_T}$.

9.2. Complex cases

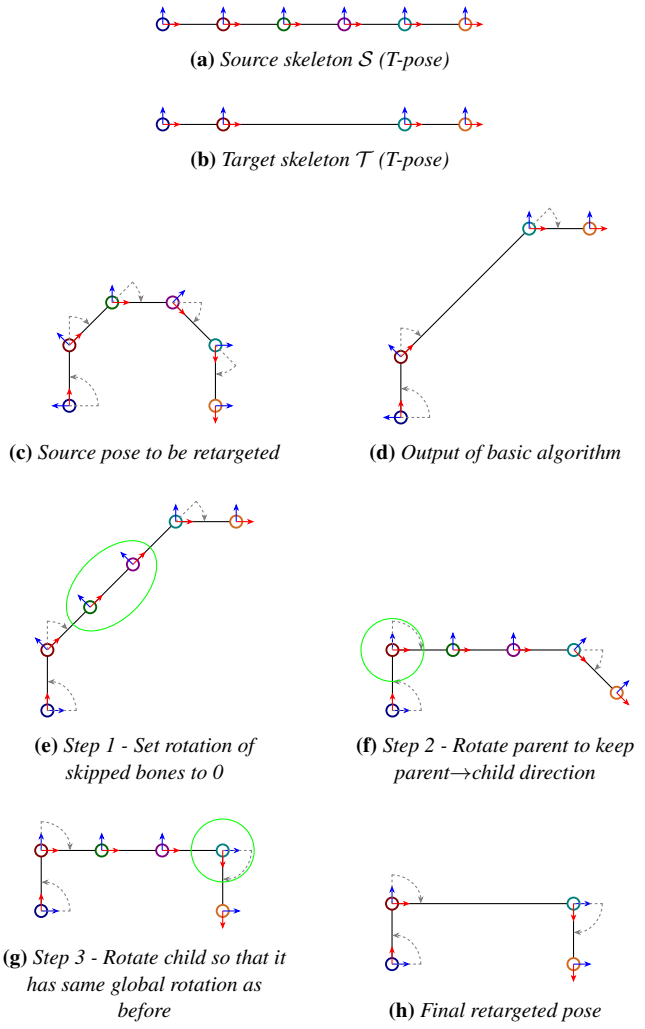
The previous formulation needs to be altered when both b_S and b_T have a single child (respectively $\overline{b_S}$ and $\overline{b_T}$ and that $(\overline{b_S}, \overline{b_T}) \notin \mathcal{M}$. We can distinguish two specific cases:

- The case where several source bones correspond to a single target bones, as illustrated in Figure 10
- The case where a single source bone corresponds to multiple target bones, as illustrated in Figure 11

9.2.1. Missing bones on the target

In the first case, transferring only the rotation of bones in the mapping means that some rotations are lost, which has a negative impact on the target pose, especially if the amount of rotation is important.

This is shown in Figure 10. Let's suppose that the source skeleton corresponds to the one on Figure 10a, shown in its T-pose, and that the target skeleton corresponds to the one on Figure 10b, with



(a) *Source skeleton $\mathcal{S}$ (T-pose)*

(b) *Target skeleton $\mathcal{T}$ (T-pose)*

(c) *Source pose to be retargeted*

(d) *Output of basic algorithm*

(e) *Step 1 - Set rotation of skipped bones to 0*

(f) *Step 2 - Rotate parent to keep parent→child direction*

(g) *Step 3 - Rotate child so that it has same global rotation as before*

(h) *Final retargeted pose*

Figure 10: *Explanation of "Copy Rotations" algorithm when the target has fewer bones than the source*

identical joint color defining the mapping between the source and target. Applying the algorithm mentioned in Section 9.1 to the pose shown on Figure 10c yields the result on Figure 10d: we can see that the pose does not look similar at all.

To fix this problem, we modify the source pose so that all non-mapped bones keep their T-pose rotation (Figure 10e). Then, we rotate the previous bone B_1 (which appears in the mapping) to align the child of the last non-mapped bone B_2 (which appears in the mapping), so that the vector $\overrightarrow{B_1B_2}$ keeps the same direction as before (albeit with a different norm), as shown on Figure 10f. Finally, we rotate B_2 so that it has the same global orientation as it had at first, in order to keep the rest of the kinematic chain unchanged, as shown on Figure 10g.

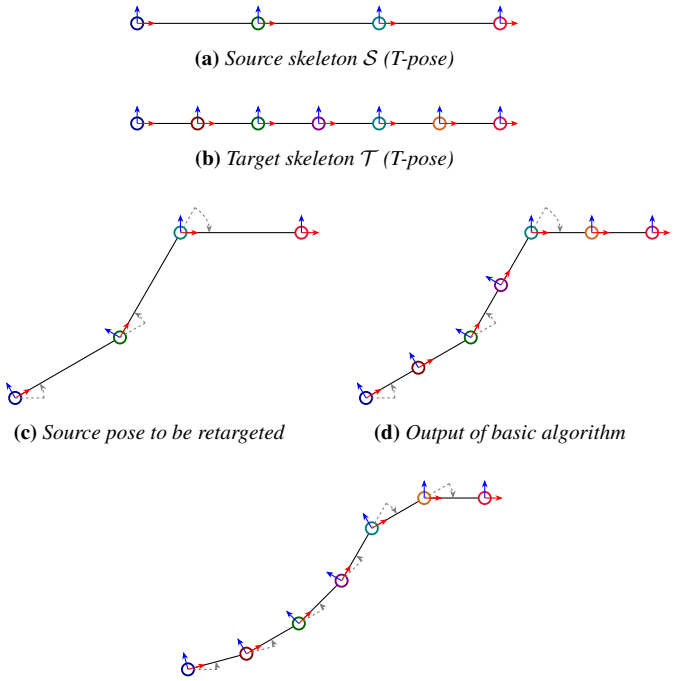
After modifying the source pose this way, the retargeting algorithm described in Section 9.1 yields the output shown in Figure 10h, which is a much better output than that of Figure 10d.

9.2.2. Additional bones on the target

In the second case, if we only transfer the rotation of the bones which appear in $\mathcal{M}$, we lose the benefit of the additional bones present on the target.

This is visible in Figure 11. Once again, let's suppose that the source skeleton (in T-pose) corresponds to the one on Figure 11a, and that the target skeleton corresponds to the one on Figure 11b. If we simply transfer the rotations of the mapped bones, for a given pose on Figure 11c, we obtain the result show on Figure 11d.

To mitigate this, we propose to "split" the rotation of the source bone, and to spread it across all of the additional bones. This gives the result shown on Figure 11e, which makes use of all bones equally and gives a more "flexible" appearance to the character.



(a) *Source skeleton $\mathcal{S}$ (T-pose)*

(b) *Target skeleton $\mathcal{T}$ (T-pose)*

(c) *Source pose to be retargeted*

(d) *Output of basic algorithm*

(e) *Final retargeted pose*

Figure 11: *Explanation of "Copy Rotations" algorithm when the target has more bones than the source*

References

[Ado24] ADOBE. *Mixamo*. https://www.mixamo.com/. Accessed: 2023-09-12. 2024 1, 2.

[ALL*20] ABERMAN, KFIR, LI, PEIZHUO, LISCHINSKI, DANI, et al. "Skeleton-Aware Networks for Deep Motion Retargeting". *ACM Transactions on Graphics (TOG), Proc. SIGGRAPH* 39.4 (2020), 62 3.

[BWBM20] BASSET, JEAN, WUHRER, STEFANIE, BOYER, EDMOND, and MULTON, FRANCK. "Contact preserving shape transfer: Retargeting motion from one shape to another". *Computers & Graphics* 89 (2020), 11–23. ISSN: 0097-8493. DOI: https://doi.org/10.1016/j.cag.2020.04.002. URL: https://www.sciencedirect.com/science/article/pii/S0097849320300406 3.

[CRV12] CASIEZ, GÉRY, ROUSSEL, NICOLAS, and VOGEL, DANIEL. "1€ filter: a simple speed-based low-pass filter for noisy input in interactive systems". *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2012, 2527–2530 7.

[Gle98] GLEICHER, MICHAEL. "Retargetting motion to new characters". *Proc. ACM SIGGRAPH*. 1998, 33–42 3.

[HKT10] HO, EDMOND, KOMURA, TAKU, and TAI, CHIEW-LAN. "Spatial Relationship Preserving Character Motion Adaptation". *ACM Trans. Graph.* 29 (July 2010). DOI: 10.1145/1833351.1778770 3.

[HZZ*23] HU, LEI, ZHANG, ZIHAO, ZHONG, CHONGYANG, et al. "Pose-Aware Attention Network for Flexible Motion Retargeting by Body Part". *IEEE Transactions on Visualization and Computer Graphics* (2023), 1–17. ISSN: 2160-9306. DOI: 10.1109/tvcg.2023.3277918. URL: http://dx.doi.org/10.1109/TVCG.2023.3277918 3.

[JKL17] JIN, TAEIL, KIM, MEEKYUNG, and LEE, SUNG-HEE. "Motion Retargeting to Preserve Spatial Relationship between Skinned Characters". *Symposium on Computer Animation (SCA)*. 2017. ISBN: 9781450350914. DOI: 10.1145/3099564.3106647. URL: https://doi.org/10.1145/3099564.3106647 3.

[LCC19] LIM, JONGIN, CHANG, HYUNG JIN, and CHOI, JIN YOUNG. "PMnet: Learning of Disentangled Pose and Movement for Unsupervised Motion Retargeting". *British Machine Vision Conference (BMVC)*. 2019 3.

[LMR*15] LOPER, MATTHEW, MAHMOOD, NAUREEN, ROMERO, JAVIER, et al. "SMPL: A Skinned Multi-Person Linear Model". *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)* 34.6 (2015), 248:1–248:16 2, 3.

[Moj24] MOJANG. *Minecraft*. 2024. URL: http://minecraft.net 5.

[Pro23] PROLIFIC. https://www.prolific.com. Accessed: 2023-12. 2023 6.

[RWY*23] REDA, DANIELE, WON, JUNGDAM, YE, YUTING, et al. "Physics-based Motion Retargeting from Sparse Inputs". *SCA, Proceedings of the ACM on Computer Graphics and Interactive Techniques* 6 (Aug. 2023), 1–19. DOI: 10.1145/3606928 3.

[Stu24] STUDIO, BLENDER. *Characters*. Apr. 30, 2024. URL: https://studio.blender.org/characters/ 2, 6.

[VCH*21] VILLEGAS, RUBEN, CEYLAN, DUYGU, HERTZMANN, AARON, et al. "Contact-aware retargeting of skinned motion". *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, 9720–9729 3.

[VYCL18] VILLEGAS, RUBEN, YANG, JIMEI, CEYLAN, DUYGU, and LEE, HONGLAK. "Neural Kinematic Networks for Unsupervised Motion Retargetting". *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018 3.

[ZGY*23] ZHANG, YUNBO, GOPINATH, DEEPAK, YE, YUTING, et al. "Simulation and Retargeting of Complex Multi-Character Interactions". *ACM SIGGRAPH 2023 Conference Proceedings*. 2023. ISBN: 9798400701597. DOI: 10.1145/3588432.3591491. URL: https://doi.org/10.1145/3588432.3591491 3.

[ZWK*23] ZHANG, JIAXU, WENG, JUNWU, KANG, DI, et al. "Skinned Motion Retargeting With Residual Perception of Motion Semantics & Geometry". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2023, 13864–13872 3, 6.