International Conference on Artificial Reality and Telexistence
Eurographics Symposium on Virtual Environments (2023)
J.-M. Normand, M. Sugimoto, V. Sundstedt, J. Dingliana, and M. Manzke (Editors)

Real-time self-contact retargeting of avatars down to finger level Appendix

Mathias DELAHAYE[1], Bruno HERBELIN[1] and Ronan BOULIC[1]

[1]École Polytechnique Fédérale de Lausanne (EPFL), Switzerland

Appendix A: Tracking technology

In this paper, we used Vive trackers (Figure 1) to track the body as this solution is consumer grade and, therefore, widely available.

Figure 1: *Vive Tracker 3.0 is a consumer-grade device whose 3D localization can be retrieved through SteamVR. Those trackers are well suited for the gaming experience, for tracking limbs (e.g., feet, knees, etc.) or objects to interact with (e.g., tracking a dummy gun). However, their dimensions prevent them from being placed on each finger.*
Picture sourced and edited from https://www.vive.com/fr/accessory/tracker3/

The finger tracking was performed using gloves on which PhaseSpace LEDs [Pha19] were added (Figure 2) to have a fine level of motion tracking required for the animation of fingers.

Both systems were calibrated to use a common frame of reference through an abstraction layer (as illustrated in Figure 3), allowing another tracking system to be used for our animation pipeline.

Appendix B: Users' Body Calibration

The user calibration process starts with the calibration of the user's extremities: the hands (with fingers section B) and feet (section B), and the head (section B). Then, our pipeline computes the CoR (Center of Rotation) starting from the effectors and tracks back towards the trunk to calibrate the arms and legs. A second pass involving self-touches is used to increase the accuracy of joint positions and to measure limb radius. Finally, the trunk (section B) and its crude mesh envelope are calibrated.

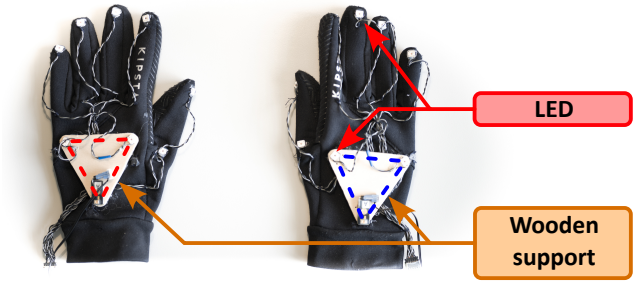


Figure 2: *The gloves' black texture helps to reduce light reflections from the tracking LED to enhance tracking. Each fingertip has an LED to track its position in the 3D space. The wooden support provides a rigid body reference to reduce the LED lateral motion due to the gloves' flexibility.*

Hands and Fingers calibration

Calibrating the user's hand is a task that involves measuring many parameters, and rather than repeating similar poses several times, we chose to measure multiple parameters simultaneously. The first pose consists in placing both hands' palms in contact with each other (Figure 4) to calibrate:

- the hands' reference frame as a regular rigid body tracked using three LEDs to determine its position and rotation in space
- the hands' surface plans: Each position from each LED of one hand is averaged with its opposite position from the other hand, and those averaged points are used to fit a plane that defines each hand's palm surface measured in each hand's frame.
- fingers radius: Knowing the surface plane in each hand's frame and the position of the LED on top of the finger, the finger radius

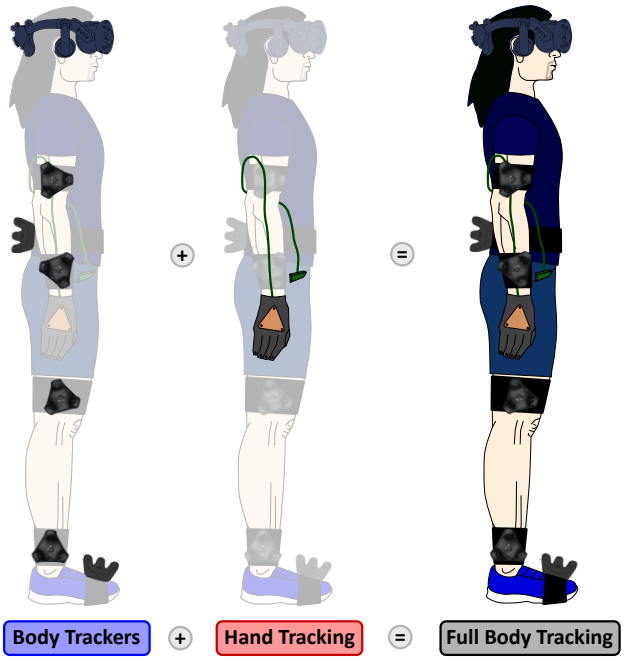


Figure 3: *The setup involves a mix of Vive Trackers 3.0 with home-made tracking gloves, therefore mixing tracking solutions.*

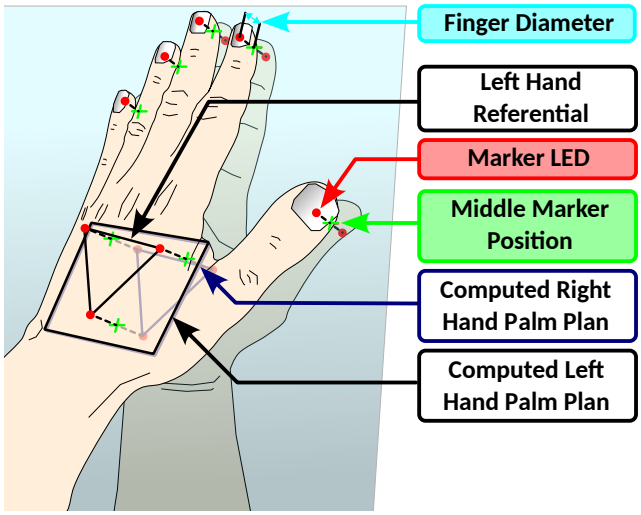


Figure 4: *The middle plan that separates both hands is illustrated in cyan. Its location is computed locally to both hands ´ reference frames (i.e., computed left-hand palm plan and computed right-hand palm plan) so that each hand model knows where is its contact surface. This pose also determines the radius of the fingertips and the local positions of extended fingers used in the animation stage to animate fingers' kinematic chains.*

is computed as half the length between the LED position and its projection on the hand's palm surface plane.
- fingers extended position: Local extended fingertip positions are stored in the hands' reference frame to be used later as a reference to compute the angular rotation to apply on each finger.

The critical information required in the hand structure for its animation is the location of its joints (i.e., the wrist and the fingers' proximal root joints). The wrist position is measured by successively placing the index fingertip from the opposite hand on top and below the wrist (Figure 5a) to calibrate its position as the mean of the two measured positions in the hand's reference frame.

Unlike the method proposed by [Ari18], our approach does not require precise placement of tracking LEDs on the pinky and index finger base joints. However, the counterpart is that those finger base joints must be calibrated.

Our initial tests showed that recording extended finger motion to calibrate the finger base joint yielded unrealistic data. Therefore, a manual calibration method was designed to calibrate the finger base's joints precisely.

Once the wrist is calibrated, each finger's base location is measured by placing the opposite index on each finger's base joint. The joint base location is then computed as the measured LED position projected on the hand's palm surface, on which the radius of the finger is added toward the top of the hand. Digits are then initialized as capsules with a radius corresponding to the measured finger radius and placed in the alignment between the joint base position and the extended fingertip position (green bones in Figure 5a). As bones' motion within the hand is relatively small, a simplified rigid structure is used to attach the fingers' base joint (i.e., proximal root) to the wrist.

The last information to identify about the hand is the crude approximation of its palm surface. This information is measured as the projection of the other hand's fingertip on the palm surface (Figure 5b).

Feet calibration

Similarly to the hands, feet embed a reference frame: the tracker attached to each foot, a crude mesh representing the contact surface under the sole of the foot, and an anchor joint.

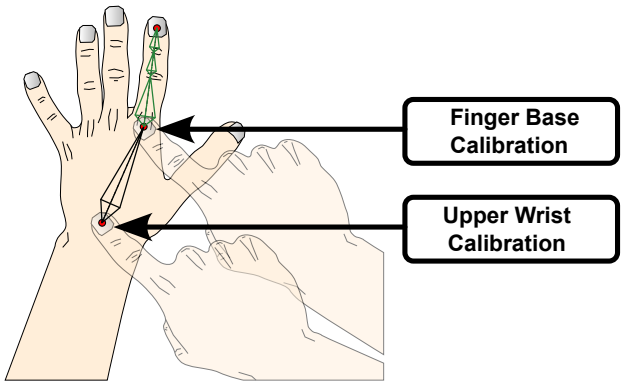
The local positions of the contact surface are measured as the fingertip's projection to the floor and then stored in the foot's reference frame (c.f., Figure 7). The ankle's local position is measured by placing the fingertip on both sides of the ankle and averaging the two positions.

Limb calibration

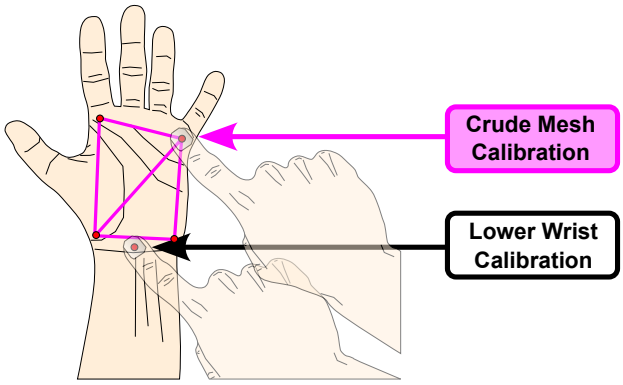
Once the effectors are calibrated, the next step is to retrieve the user's skeleton structure by progressing proximally toward the trunk.

Therefore, the next step is to calibrate the four limbs linking the effectors to the trunk: the arms and the legs.

Limbs are kinematic chains composed of two bones: one close to the trunk, which we call the anchored bone, and the other one chained to it and attached to the effector joint, called here, the intermediate bone.

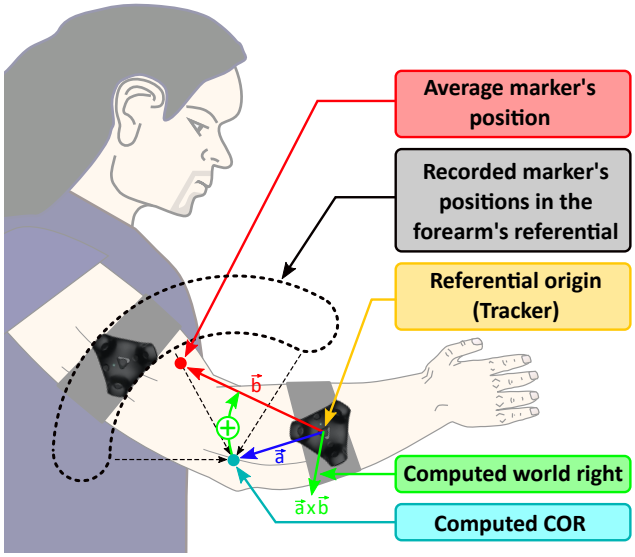


(a) *Finger base calibration*

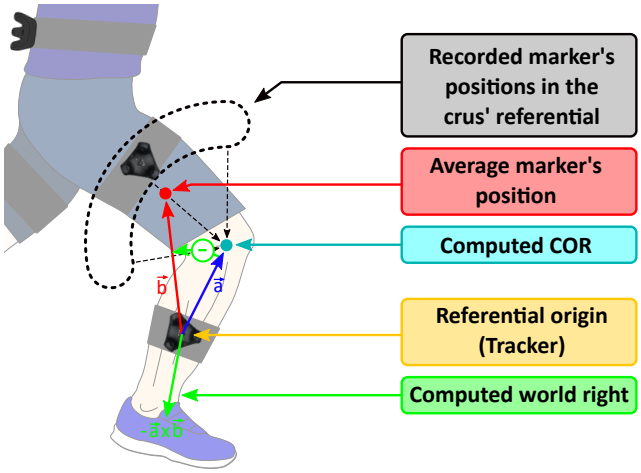


(b) *Hands' palm's crude mesh calibration*

Figure 5: *To calibrate a point, one must place the other hand's fingertips (the index by default) on top of the point of interest (e.g., joint) The order in which the user calibrates points is irrelevant, as all calibration points are first stored. The actual computation can be triggered later, hence avoiding flipping the hand several times to calibrate the wrist, surfaces, and finger base joint.*



(a) *The tracker's average position is above the half-plan passing by the tracker and the computed COR.*



(b) *The tracker's average position is below the half-plan passing by the tracker and the computed COR.*

Figure 6: *The position of the parent bone's tracker is recorded on a time window of 150 frames (this allows a sufficient average for measurements of the CoR when the user moves the limb). It generates a cloud of points illustrated in black, covering the history of the tracker positions, which is used to retrieve the unique normal corresponding to the joint direction's local right. N.B. Unity uses a left-handed coordinate system. Thus, the output of the cross-product is the opposite of what is expected with a right-handed coordinate system.*

Linear bones are fully constrained once the length, axis direction, and local right directions are determined (two orthogonal vectors are enough to fully constrain the three degrees of rotations from the bone's orientation, and the root point fixes the three remaining degrees of freedom for the bone's placement) as illustrated in Figure 8.

Therefore, the intermediate bone is calibrated by determining the intermediate joint position (elbow/knee) with its local right.

Based on the methodology from [MGB17], the user performs gym motion by flexing arms/legs. At the same time, the relative displacement of the anchored bone tracker is recorded in the intermediate bone's reference frame (Figure 6).

Intermediate Joint and Bone Calibration Knowing the topology of the intermediate joint, the expected shape of the recorded set of point shapes is a circle in a plan. However, knowing the plan is insufficient to determine which normal side should be used as the local right of the limb's kinematic chain.

Therefore, we rely on the knee and the elbow's articular limit, which prevents the joint's angle from exceeding 180 degrees. This means that the average position of the recorded set is necessarily on one side of the half-plan passing by the tracker position and the joint location as illustrated in Figure 6.

A first approximation of the joint location used in this computa-

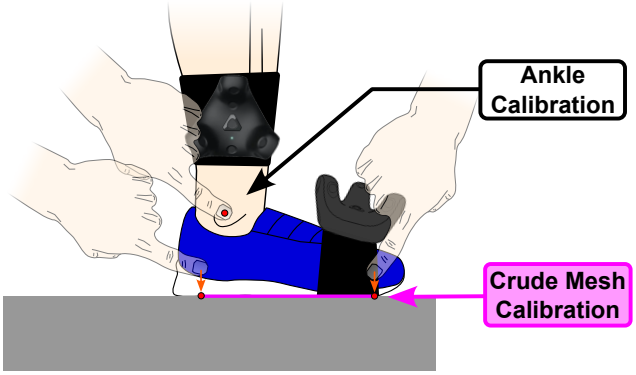


Figure 7: *The foot calibration process expects the user to place their fingertips on the edges of his foot to calibrate the foot's planar surface in contact with the floor, with a projection applied (small orange arrows) to ensure the measured position is on the floor. The user also places the fingertip on both sides of the malleolus to calibrate the ankle's joint position.*

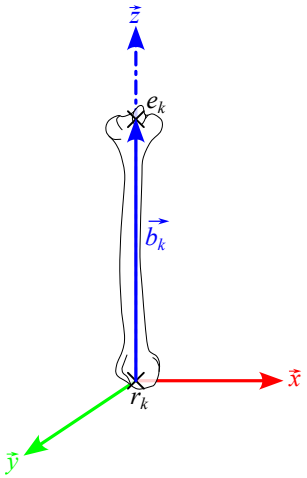


Figure 8: *The linear bone b_k has a structure containing an origin (r_k) and a bone vector $\vec{b_k}$ that links the proximal joint to the distal joint e_k. Each bone has its reference frame in which the bone axis is along the local forward direction $\vec{z}$. The bone consequently also has a local right ($\vec{x}$) and a local up ($\vec{y}$). The local right is set as the joint flexion axis (right-sided) for limbs. Constraining the axis direction and its local right (or up) is enough to constrain its world orientation fully. Setting the origin's position fixes the remaining degrees of freedom that fully constrain the bone's placement.*

tion is performed by fitting a plan from the recorded set of points and then projecting the points on this plan to fit a circle and compute its center.

The plan's fitting is performed by extracting the average positions from the dataset constituting the plan's origin and removing this computed origin from each dataset point. The two main directions of the plan are extracted from the two eigenvectors with the largest eigenvalues of the product of the transposed matrix of the dataset with itself.

Once the plan is determined, a circular regression retrieves the circle center and radius. A second pass is applied to remove the contribution from points whose distance is greater than two times the standard deviation in terms of distance toward the center.

However, this method can only retrieve the axis on which the joint is located but not the joint location itself, as this relies on the radius of the limbs that cannot be inferred from this motion as only one tracker is placed on the user's bone, unlike the approach from [MGB17] where multiple LEDs are placed around the arm and helps to retrieve the location of the joint.

Therefore, to calibrate more precisely the intermediate joint location and the limb radius simultaneously, the user places the fingertip from the other hand on each side of the elbow to calibrate its position as the average position between both points. The intermediate limb radius is computed as the average between the measured radius at the effector's joint (c.f., section B, section B) and the current measure of the limb's radius at the joint.

Anchor Joint and Bone Calibration The shoulders and hips are joints that provide more degrees of freedom than the elbows and knees. Thus, rather than having a tracker distribution be a circle, the distribution can now be extended to a sphere that fully constrains the location of the CoR.

Users, therefore, move their arms, paying attention not to lift the arm above the horizontal line and not to mobilize the clavicle. The algorithm records the root's tracker position in the brachium/thigh trackers' frame to locate the joint position.

The CoR is computed as the sphere's center of the recorded dataset using spherical regression. The radius is the average distance between each recorded point and the computed center. The second pass excludes points whose distance to the center is larger than two times the standard deviation, and the same process is reapplied with the filtered input.

This is followed by measuring two points diametrically opposed at the anchor joint to average the anchored limb's radius.

Head Calibration

The jaw's surface is calibrated with six calibration points located at the left and right ear, on the upper lip, at the chin, and on the middle of the left and right side of the jaw, as illustrated in magenta. The calibration is performed by placing fingertips on the illustrated locations. This crude mesh is rigidly attached to the HMD's tracker reference frame; therefore, it does not consider when users open their mouths.

The crude mesh topology differs slightly from the one from [MGB17] as the user does not wear an HMD in their approach; hence, their crude mesh can also cover the rest of the face, which is impossible here due to the presence of the HMD.

Also, here, we approximated the back of the head as a sphere rather than using the crude mesh; the sphere was calibrated by

placing the fingertips on the skull's top, right, left, and back (blue dots on Figure 9). Those calibration points are stored locally in the head's frame: the HMD. The same fitting procedure is used to fit a sphere passing through those calibration points to approximate the back of the head surface. Two additional measurements, illustrated in green, are performed behind the jaw to measure the skull base where the spine is attached to the head.

(a) *Front view* **(b)** *Right view*

Figure 9: *Except for the spine, which might contain a different number of joints compared to the user's skeleton model, the calibrated avatar contains the same structure as the user's skeleton. This means that each surface element has an equivalent in the source user model.*

Trunk Calibration

With all limb anchors and the head calibrated, we can perform the trunk calibration. When standing straight up, the sacrum bone width is measured as the distance between both hip joints. According to [LHS*20], the sacrum height is, on average, 11.4 cm for men (standard deviation of 1.1 cm) and 10.9 cm, with a standard deviation of 1.0 cm for women. Therefore, given the relatively small range of scale of this bone compared to the user's morphology, the root of the spine is statically set to be 10cm above the defined origin of the sacrum (the sacrum bone's model is constructed in a way that both hips are symmetrically placed from the origin) and 5cm backward, and its initial rotation along the hips axis is set so that the spine is vertically aligned as the user stands straight. Then, its position is stored in the back's tracker reference frame. Additionally, the height of the user's sacrum when standing up is also stored.

Knowing the location of the spine root on the sacrum bone and the skull's base joint, we compute the extended spine distance (used later to compute its flexion) and each vertebra length. Our model uses a spine composed of 12 vertebras, and the clavicular bones are calibrated to link the 5th vertebra to each shoulder.

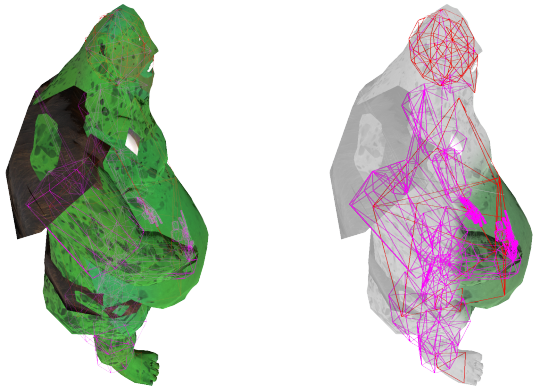
Finally, the user calibrates the torso shape by placing his hand palm on different key points of the torso marked as yellow spheres on Figure 11a to measure its body shape. Those crude mesh calibration points are stored in the closest vertebra reference frame so that when the user moves, the crude mesh can be deformed accordingly.

Appendix C: Egocentric coordinates

The retargeting pipeline relies on measuring distances toward each surface element and re-applying those scaled distances onto the targetted avatar. Therefore, the avatar structure must comply with the one from the user's skeleton model. Here, the calibration of skeleton bones is direct through the skeleton's rig of the avatar; only the local right directions for knees, elbows, and fingers must be specified to know along which axis joints flex.

The evaluation of the body shape uses the same principle as the user's body surface calibration, except those surface measurements are performed using ray cast hit points on the collider mesh of the avatar. Crude mesh calibration points are stored as the position on the avatar's mesh, and the number of vertebrae can differ from the user's skeleton model.

It was observed that the simple mesh representation of a character's belly in [MGB17], which consists of only seven points on the front and three on the back, is not suitable for accurately representing rounded surfaces, such as an ogre's large belly. This is because there may be interpenetration caused by the gap between the spherical surface and the crude mesh surface that is its chord, as illustrated in Figure 10.



(a) *Illustration of the pose with interpenetrations*

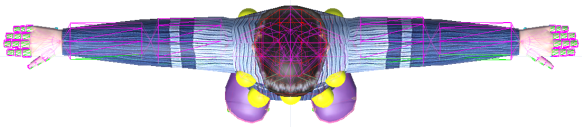
(b) *View of the inner structure used to animate the avatar; the right hand is correctly placed on the surface of the crude mesh, but the latter being within the belly, the hand is also within the belly*

Figure 10: *Example demonstrating the problem that can arise when the polygon count in the crude mesh is too low. In this case, the right hand is positioned correctly on the surface of the crude mesh, but since the crude mesh represents a chord of the belly's rounded shape, the hand interpenetrates with the belly.*

To mitigate this issue, we have included four additional points in the center of the crude mesh's belly to reduce the distance between the chord and the surface itself (Figure 11a). By default, these points are interpolated from the four corners that are used to define the user's belly unless the user has a large belly that necessitates more refined calibration. These points remain calibrated manually once for the targeted avatars. Figure 11 illustrates a cal-

ibrated avatar with the new topology of the crude mesh and the whole set of surface elements.

(a) *Front view with the new crude mesh topology*



(b) *Top view*

Figure 11: *Capsule bones are calibrated to represent the user's limb shape and finger digits bones, while the crude mesh represents the shape of the torso, hand and feet, palm surfaces, jaw, and a sphere approximates the back of the skull's surface.*

This process generates a configuration file that can be stored for each avatar; hence, this process needs to be applied only once per avatar.

Vector contributions

Raw contributions weights computation

The raw $\Lambda_{i,j}$ weights are computed as:

- $\frac{1}{\|\overrightarrow{v_{i,j}}\|^2}$ for spherical coordinates
- $\frac{1}{\|\overrightarrow{v_{i,j}}\|^2} \cdot \left|\sin\left(\angle(\overrightarrow{v_{i,j}}, \overrightarrow{b_i})\right)\right|$ for cylindrical coordinates with $\overrightarrow{b_i}$ the axis of s_i cylindrical coordinate system.
- $\frac{1}{\|\overrightarrow{v_{i,j}}\|^2} \cdot \cos\left(\angle(\overrightarrow{v_{i,j}}, \overrightarrow{n_i})\right)$ for crude mesh elements with $\overrightarrow{n_i}$ the normal of s_i's triangle
- $\frac{1}{h_j^2}$ for the floor's height contribution.

Weight contribution normalization

The floor contribution presents a singularity: It only contributes to the vertical height of p'_j but not the lateral location on the ground

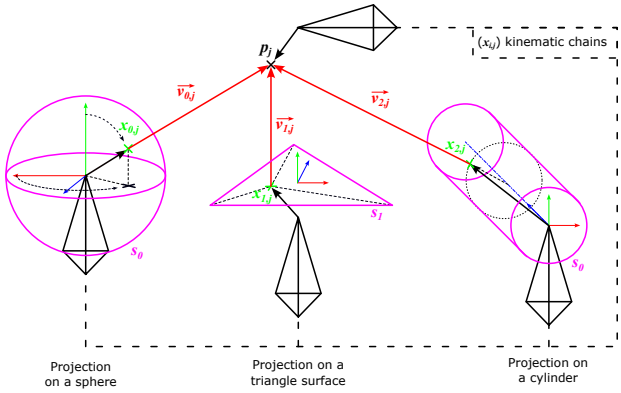


Figure 12: *In the egocentric coordinate system, each point position p_j is decomposed into a sum of contributions from each element surface s_i. Those surface elements can either be a sphere (on the left), a mesh triangle (middle), or a cylinder (on the right). The contribution for each surface element to the jth target is denoted $\overrightarrow{v_{i,j}}$ (in red) and represents the vector between p_j and the closest projection point of p_j on s_i that is noted $x_{i,j}$ (in green). Finally, a normalization factor $\tau_{i,j}$ is computed as the sum of the dot product of each bone's length and $\overrightarrow{v_{i,j}}$.*

plan. Hence, when a target is close to the ground, the weight contribution of the floor $\lambda_{\text{floor\_id},j}$ would tend to 1, erasing all the other contributions, which are the only ones contributing to the planar lateral position. Ultimately, it would result in a retro-projected point to the origin of the space rather than just on the floor, not to mention the precision issues when dealing with vectors with tiny amplitude to determine a direction.

Consequently, two sets of weights are actually computed (for both executions): $\left(\lambda_{i,j}\right)_{i\in\mathbb{S}}$ and $\left(\lambda_{g_{i,j}}\right)_{i\in\mathbb{S}}$. The last one ($\left(\lambda_{g_{i,j}}\right)_{i\in\mathbb{S}}$) is computed by normalizing the set of all raw contributions (i.e., including the one from the ground) while the former ($\left(\lambda_{i,j}\right)_{i\in\mathbb{S}}$) skips the contribution of the ground in its normalization process.

The retro-projection formula of p'_j is therefore adapted into Equation 1 to address this singularity (with $\overrightarrow{e_1}$, $\overrightarrow{e_2}$ and $\overrightarrow{e_3}$ the x (right), y (up), and z (front) world axis respectively).

$$\begin{cases} p'_j \cdot \overrightarrow{e_1} = \left(\sum_{i\in\mathbb{S}} \left(x'_{i,j} + \overrightarrow{v_{i,j}} \cdot \frac{\tau'_{i,j}}{\tau_{i,j}}\right) \cdot \lambda_{i,j}\right) \cdot \overrightarrow{e_1} \\ p'_j \cdot \overrightarrow{e_2} = \left(\sum_{i\in\mathbb{S}\setminus\{\text{floor\_id}\}} \left(x'_{i,j} + \overrightarrow{v_{i,j}} \cdot \frac{\tau'_{i,j}}{\tau_{i,j}}\right) \cdot \lambda_{g_{i,j}} + h_j \cdot \lambda_{g_{\text{floor\_id},j}}\right) \cdot \overrightarrow{e_1} \\ p'_j \cdot \overrightarrow{e_3} = \left(\sum_{i\in\mathbb{S}} \left(x'_{i,j} + \overrightarrow{v_{i,j}} \cdot \frac{\tau'_{i,j}}{\tau_{i,j}}\right) \cdot \lambda_{i,j}\right) \cdot \overrightarrow{e_3} \end{cases} \quad (1)$$

Kinematic Normalization

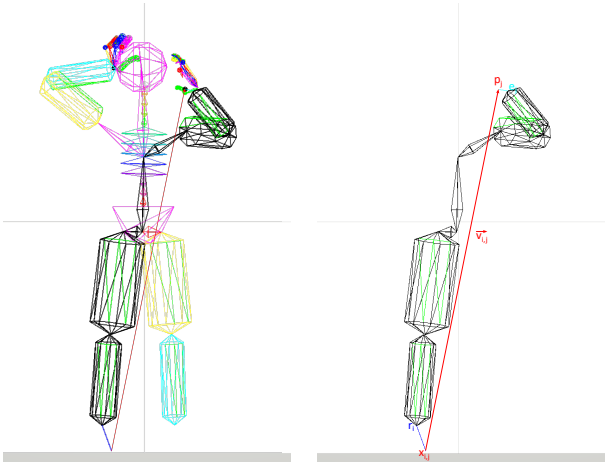
The scaling factor for the source kinematic chain linking the target p_j to its projection $x_{i,j}$ on the surface element s_i is noted $\tau_{i,j}$. Those

chains are pre-computed and stored so that the tree search is not performed in every frame.

Let $\left(\overrightarrow{b_k}\right)_{k\in[\![0,n]\!]}$ be the list of n bones axis vectors that comprise this kinematic chain, with r_i the root of the surface element s_i and e_j the root of the bone to which the target is attached. We have Equation 2

$$\tau_{i,j} = \sum_{k=0}^{n} \overrightarrow{\hat{v}_{i,j}} \cdot \overrightarrow{b_k} + \overrightarrow{\hat{v}_{i,j}} \cdot \overrightarrow{(x_{i,j} - r_i)} + \overrightarrow{\hat{v}_{i,j}} \cdot \overrightarrow{(p_j - e_j)} \tag{2}$$

This is illustrated in Figure 13 where the contribution vector $\overrightarrow{v_{i,j}}$ is displayed in red, the kinematic chain in black, and the extremity segments in blue and cyan. The kinematic path computation uses the trunk's simplified kinematic chain to accelerate the process.



(a) *Kinematic chain with the whole skeleton structure (without the crude mesh)*

(b) *Kinematic chain only with the surface reference points and extremity segments detailed*

Figure 13: *Illustration of the kinematic chain used to compute the normalization factor τ. The spine model was simplified to reduce the computational cost.*

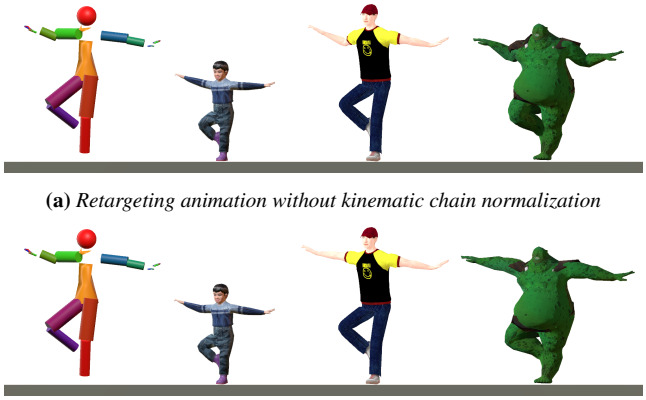
When the associated lambda for a kinematic chain is null, the computation of the kinematic chain is skipped, as the associated vector will not contribute to the reprojection stages. An example of a normalization process enhancing the final result is illustrated in Figure 14.

Appendix D: Body animation

Finger motion capture

The inputs from the transformed mocap data combined with the occlusion recovery pipeline from [PDP*19] are used to provide a set of ordered points for the animation of the hand's model structure. This is used in both the skeleton reconstruction (to acquire the reference finger poses of the user) and the reconstruction stage of the avatar's hands in the retargeting stage.

With the information retrieved from the hand's calibration



(a) *Retargeting animation without kinematic chain normalization*

(b) *Retargeting animation with kinematic chain normalization*

Figure 14: *Illustration of the same pose on different avatars with and without the normalization enabled. The structure on the left is the modelized user's skeleton. On top, the non-normalized animation produces flexions in the characters with long arms (the two avatars on the right), while for the child, the arm is completely extended, although this was not the case on the source skeleton. On the bottom, the normalization of effector positions straightens the arms of the two characters on the right and reduces the extension of the child's arms.*

stage, and assuming that the flexion angle is the same between the intermediate-distal joint and the proximal-intermediate joint [Ari18], we can compute the flexion angle of each finger based on the distance between the fingertip and the finger proximal's joint location that can later be applied on the finger's kinematic chain as illustrated in Figure 15.

The computation firstly computes and caches the coefficients from Equation 3 and then calculates the flexion angle following the steps from Equation 3.

$$a = 4 \cdot l_1 \cdot l_3, \qquad b = -2 \cdot (l_1 + l_3), \cdot l_2 \qquad c_1 = l_1^2 + l_2^2 + l_3^2 - 2 \cdot l_1 \cdot l_3 \tag{3}$$

$$c = c_1 - d^2, \qquad \Delta = b^2 - 4 \cdot a \cdot c, \qquad x_1 = \frac{-b - \sqrt{\Delta}}{2 \cdot a}$$
$$\hat{x}_1 = \begin{cases} -0.9999 & \text{if} \quad x_1 \leq -0.9999 \\ x_1 & \text{if} \quad -0.9999 < x_1 < 0.9999 \\ 0.9999 & \text{if} \quad 0.9999 \leq x_1 \end{cases} \tag{4}$$
$$\alpha = \pi - \arccos(\hat{x}_1)$$

Before its application, the finger is realigned with the artificial bone linking the wrist to the base joint, the angle α is constrained not to exceed $90°$, and then finally applied to the finger. For the thumb, the bone linking the wrist to the base joint is rotated along its axis by $45°$.

A second pass is then applied to enforce the alignment of fingertips with the expected effectors' positions. The realignment is

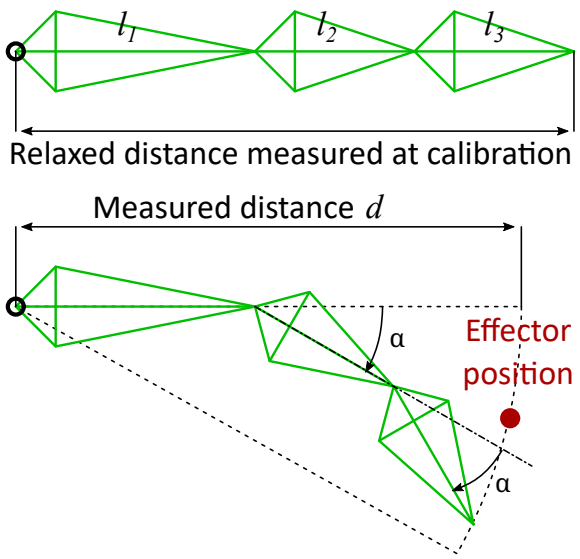


Figure 15: *The flexion of a finger is based on the effector-finger base joint distance. Flexing the finger is insufficient to enforce the effector's position to match the finger's kinematic chain extremity.*

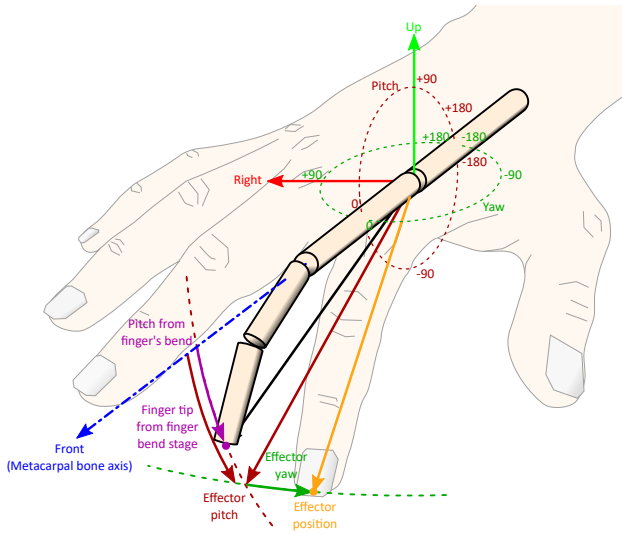


Figure 16: *In the realignment process, the algorithm measures the difference in yaw and pitch between the position of the user's fingertip and the only flexed finger model. Then, the rotation required to align the reference axis with the user's finger is computed as the rotation that rotates the reference axis by the differences in yaw and pitch. This rotation is then applied to all the joints of the finger, and the position of the distal end of each digit segment is computed to update the origin of the next proximal digit side.*

performed by measuring the pitch and yaw from the expected effector position in the metacarpal bone's reference frame centered on the proximal's root (Figure 16) As flexion induces pure pitch in the finger's tip location in the bone attached to the wrist's reference frame (e.g., metacarpal for the index), we already know that the current yaw of the flexed finger is zero; therefore, only the pitch of the animated finger is computed before computing the realignment rotation.

The differences in yaw and pitch are then applied to the proximal root joint and forwarded to the rest of the chain (Figure 16). To prevent impossible positions, the measured targeted yaw and pitch from the direction of the expected effector's position is capped using values from [Ari18] for the base joints: The yaw is constrained within $[-15°;15°]$ and the pitch within $[-85°;+10°]$ for the index, middle, ring, and pinky fingers, whereas the yaw is constrained within $[-30°;40°]$ and the pitch within $[-15°;+15°]$ for the thumb.

Limb animation

During the calibration process, joint locations were recorded in each tracker's frame; therefore, the computation of joint positions is done by expressing calibrated joint positions in world coordinates. Local directions of the bones' axis and local right are also stored during the calibration process, allowing for direct placement in the space of each individual limb bone.

However, trackers' locations are not perfectly rigidly attached to the user's bone, and some offsets may occur, leading to structural gaps. Therefore, anchored and intermediate limb bones (brachium/thigh and forearm/crus) are scaled to ensure a junction of the kinematic chains.

This process is performed by computing the intermediate joint location in both the intermediate and anchored bones' trackers and to average the computed position of the joint. As we gave priority to the effector over the intermediate joint location, the effector position is solely determined using the effector's reference frame and is not averaged with the intermediate's bone extremity.

The anchored bone is then scaled and oriented to align it with its previously computed anchor position and the newly average intermediate joint position. The intermediate bone is scaled and aligned to make the junction between the intermediate joint position and the effector's joint position.

The realignment is performed while maintaining the local right direction to prevent the twist of bones. This process is illustrated in Figure 17.

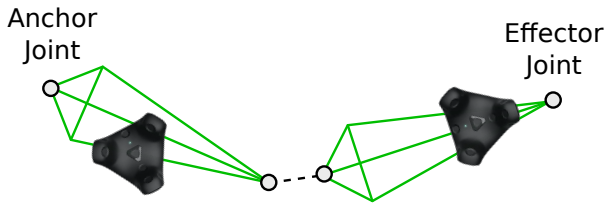
Trunk animation

The last component to animate before animating the trunk is the head. As a simple rigid body attached to the head tracker (i.e., the HMD), the head animation is a simple placement of a rigid body in space. This placement determines the skull base corresponding to the spine's targetted effector position IK.

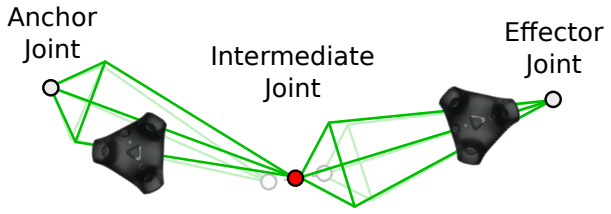
At this point, all four limb anchors' positions are determined, and the root trackers' positions and orientation are known as the targeted position of the skull base and its local right.

The animation of the trunk is performed in two passes:

- The first places the sacrum as a rigid body attached to the root's



(a) *Initial limb's structure*



(b) *Adapted limb's structure with continuity enforcement*

Figure 17: *Illustration of the process of linking bones on a limb*

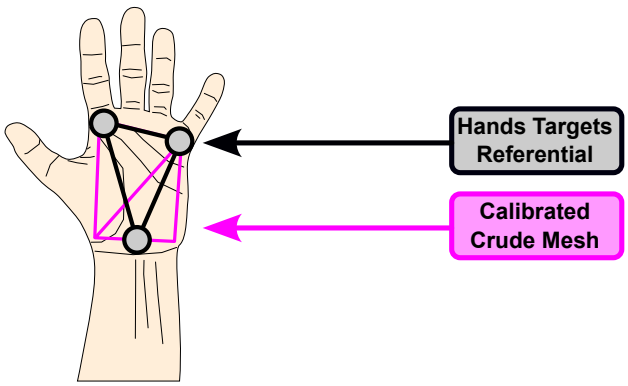


Figure 18: *Three targets are assigned to each effector used as a rigid body frame that allows the expression of the wrist location in this coordinate system and to measure a rotation.*

tracker. To accommodate the back tracker's potential lateral displacement compared to the sacrum bone, the sacrum's bone lateral rotation is averaged with the lateral direction computed from the hips using the thigh trackers. The same is also applied to the root position of the sacrum. Once the sacrum is placed, the approach from [UPBAS08] is used to animate the spine flexion as illustrated in Figure 19. The spine is then realigned with the targeted effector position, inducing an unrealistically large joint rotation between the sacrum and the first vertebra, according to biomechanics [UPBAS08].

- Therefore, a second pass is applied by rotating the sacrum along its hip flexion axis to align its upward direction with the first vertebra's direction. This changes the root position of the spine, hence the distance between the anchor (sacrum) and the effector (skull base); therefore, a second pass is applied to compute the new flexion and the new realignment of the spine producing the final position from Figure 19.

Finally, vertebrae are uniformly twisted along their axis to account for the hips-shoulders and shoulders-head twists.

Once the spine state is determined, the clavicle bones are computed to link the 5th vertebra (for our spine model) to the evaluated shoulder anchor positions.

For the later stage of kinematic path normalization, shortcut bones are added, in addition to the bones used to animate the user's skeleton (Figure 20), to link: The sacrum root to the left and right hips links the sacrum to the clavicle root and links the clavicle root to the head, similarly to what was proposed by the normalized skeleton representation from [KMA05]. Those bones have no constraints on the twist, as their contribution is only used to compute kinematic path normalization, which only considers the bone's axis vector. Those bones can be easily spotted in black down to Figure 13.

Appendix E: Inverse Kinematics

The IK works as follow: Knowing each bone's length (l_1 and l_2), this information is combined with the measurement of the distance effector base joint (d) to retrieve the intermediate flexion angle α (Figure 21a) using the formulas Equation 5 and Equation 6.

$$\cos\alpha = l_1^2 + l_2^2 - \frac{d^2}{2 \cdot l_1 \cdot l_2} \tag{5}$$

$$\alpha = \begin{cases} \pi & \text{if} \quad \cos\alpha \geq 1 \\ 0 & \text{if} \quad \cos\alpha \leq -1 \\ \pi - \arccos(\cos\alpha) & \text{otherwise} \end{cases} \tag{6}$$

To finish the limb's animation, an analytical IK is used with the newly computed location of the retargeted effector and intermediate joint position. This is applied to produce the animation of the four avatar's limbs.

Once the flexion angle is computed, it is applied to the intermediate joint as a rotation along its right (conversely left for the legs) axis to produce the flexion. The second stage then aligns the produced effector position with the expected one.

At that stage, the swivel angle along the root-effector axis remains to be determined (Figure 21b); logically, one expects to infer it from the location of the reprojected intermediate joint target point.

However, this approach becomes unstable the closer the retroprojected intermediate joint is to the root-effector axis, as three aligned points cannot constrain a plan.

Therefore, the swivel angle is adjusted using an interpolation between the source skeleton model limb swivel angle (used for unstable cases) and the angle to align the limb intermediate joint in the half-plane determined by the reprojected intermediate joint position (used when its distance to the swivel axis is sufficient to avoid instabilities). The alignment swivel angle δ is computed using Equation 7 and Equation 8 with r the radial distance of the reprojected

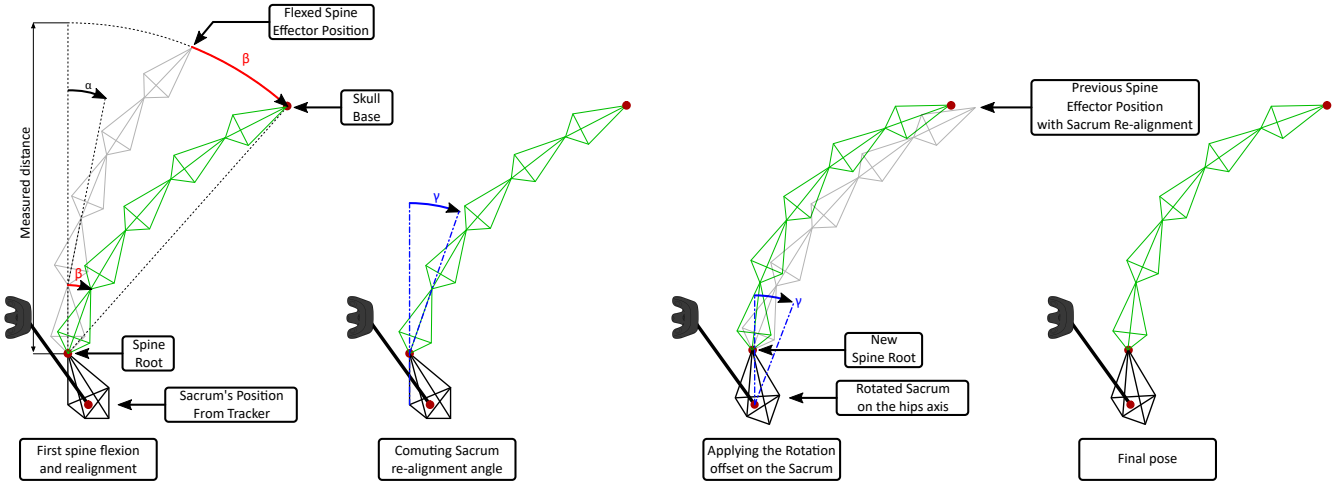


Figure 19: *The spine animation process comprises a first resolution of the spine using the IK method from [UPBAS08], followed by a realignment and a second pass.*

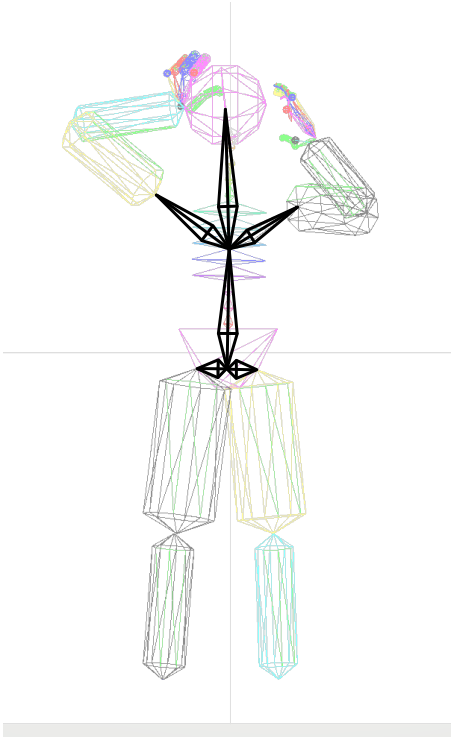


Figure 20: *Illustration, in black, of the shortcut bones used to skip intermediate bones for the normalization computation process.*

intermediate joint target position onto the root-effector axis (Figure 21b).

$$t = \begin{cases} 0 & \text{if} \quad r \leq 0.05 \\ (r-0.05)\cdot 10 & \text{if} \quad 0.05 < r < 0.15 \\ 1 & \text{if} \quad 0.15 \leq r \end{cases} \tag{7}$$

$$\delta = t \cdot \beta + (1-t) \cdot \gamma \tag{8}$$

Finally, the angle δ is applied through a rotation along the root-effector axis to end the process.

Appendix F: Subjective evaluation

Experimental view

Poses used in the evaluation

References

[Ari18] ARISTIDOU A.: Hand tracking with physiological constraints. *The Visual Computer 34*, 2 (Feb 2018), 213–228. doi:10.1007/s00371-016-1327-8. 2, 7, 8

[KMA05] KULPA R., MULTON F., ARNALDI B.: Morphology-independent representation of motions for interactive human-like animation. In *Eurographics* (2005). 9

[LHS*20] LANGNER I., HENKER C., STEINHAGEN K., BÜLOW R., LANGNER S., SCHMIDT C.-O.: Can sacrum height predict body height, age, and sex? a large population-based mri study. *Forensic Imaging 21* (2020), 200379. doi:https://doi.org/10.1016/j.fri.2020.200379. 5

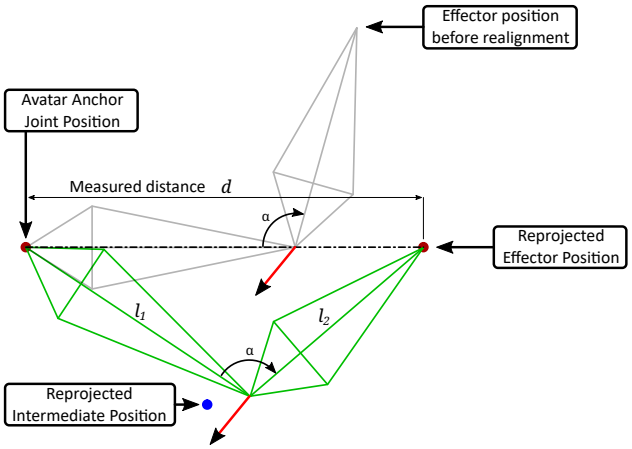
[MGB17] MOLLA E., GALVAN DEBARBA H., BOULIC R.: Egocentric mapping of body surface constraints. *IEEE Transactions on Visualization and Computer Graphics* (2017), 1–1. doi:10.1109/TVCG.2017.2708083. 3, 4, 5

Table 1: *Illustration of the targeted poses used for the evaluation*

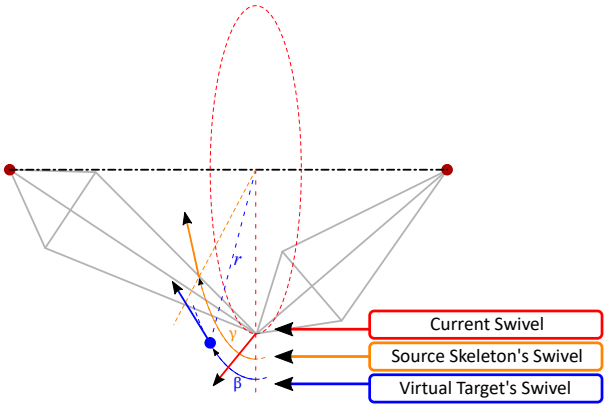
[PDP*19] PAVLLO D., DELAHAYE M., PORSSUT T., HERBELIN B., BOULIC R.: Real-time neural network prediction for handling two-hands mutual occlusions. *Computers & Graphics: X 2* (2019), 100011. doi:https://doi.org/10.1016/j.cagx.2019.100011. 7

[Pha19] PHASESPACE: Phasespace impulse x2. https://phasespace.com/x2e-motion-capture/, 2019. 1

[UPBAS08] UNZUETA L., PEINADO M., BOULIC R., ÁNGEL SUESCUN: Full-body performance animation with sequential inverse kinematics. *Graphical Models 70*, 5 (2008), 87–104. doi:https://doi.org/10.1016/j.gmod.2008.03.002. 9, 10



(a) *Limb flexion and realignment computation*



(b) *Swivel computation*

Figure 21: *Swivel alignment: The first step of the IK flexes the limb and aligns it towards the effector, leaving the swivel unconstrained. Then, based on the miss-alignment (r), the current swivel is re-aligned as a merged between the user's skeleton swivel and the one defined by the half-plane comprising the retro-projected target point and the root-effector axis.*

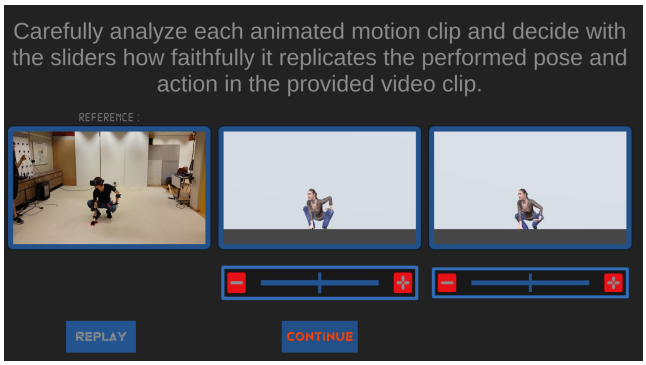


Figure 22: *Screenshot of the displayed interface to the user on the full right: the original action performed by a person, on the two middle and right: the animated avatar using either the approach with the retargeting pipeline enabled or with only raw angles provided by the user's skeleton input. The order between the two animation methods is randomized; hence, it is unknown to the participant. Two continuous sliders are displayed below the videos to allow participants to conduct individual video evaluations. Finally, the participant can replay the videos as much as they want and validate their choice using buttons from the interface.*