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

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

NePHIM: A Neural Physics-Based Head-Hand Interaction Model

Nicolas Wagner[1,2], Ulrich Schwanecke[2], and Mario Botsch[1,3]

[1]TU Dortmund University, Germany
[2]RheinMain University of Applied Sciences, Germany
[3]Lamarr Institute for Machine Learning and Artificial Intelligence, Germany

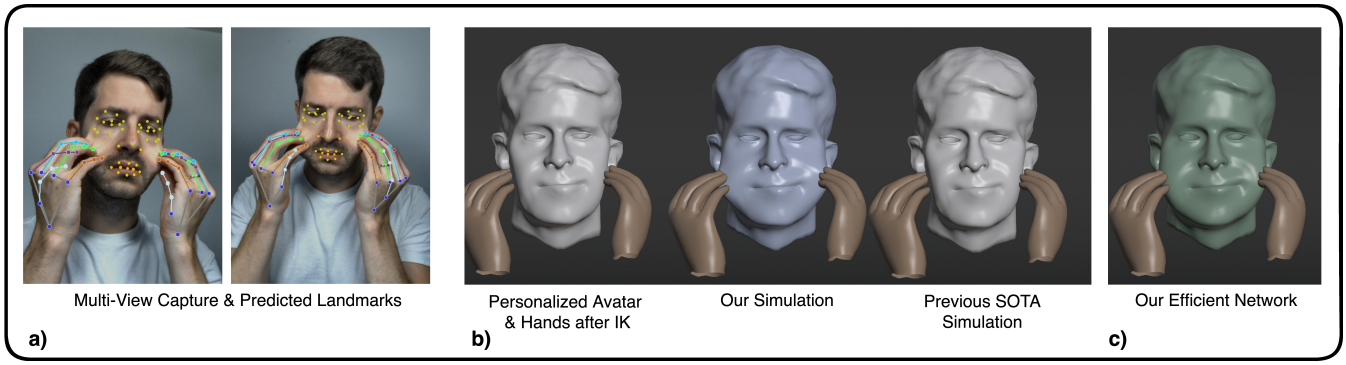


Figure 1: *The different steps of NePHIM by means of a single frame: a) Two of the 16 views of our multi-camera rig used to capture head-hand interactions and corresponding landmarks. b) Our proposed simulation for head-hand interactions in comparison to the tracked input (after fitting template surfaces to the landmarks (IK)) and the simulation used in the previous state-of-the-art [SGPT23]. c) Prediction of the efficient neural network trained to approximate our simulation.*

Abstract

Due to the increasing use of virtual avatars, the animation of head-hand interactions has recently gained attention. To this end, we present a novel volumetric and physics-based interaction simulation. In contrast to previous work, our simulation incorporates temporal effects such as collision paths, respects anatomical constraints, and can detect and simulate skin pulling. As a result, we can achieve more natural-looking interaction animations and take a step towards greater realism. However, like most complex and computationally expensive simulations, ours is not real-time capable even on high-end machines. Therefore, we train small and efficient neural networks as accurate approximations that achieve about 200 FPS on consumer GPUs, about 50 FPS on CPUs, and are learned in less than four hours for one person. In general, our focus is not to generalize the approximation networks to low-resolution head models but to adapt them to more detailed personalized avatars. Nevertheless, we show that these networks can learn to approximate our head-hand interaction model for multiple identities while maintaining computational efficiency.

Since the quality of the simulations can only be judged subjectively, we conducted a comprehensive user study which confirms the improved realism of our approach. In addition, we provide extensive visual results and inspect the neural approximations quantitatively. All data used in this work has been recorded with a multi–view camera rig. Code and data are available at https://gitlab.cs.hs-rm.de/cvmr_releases/HeadHand.

1 Introduction

How many times per hour do you think you touch your face? Probably more often than you are aware of. Although the answer to this question varies in scientific studies, it can be said that, on average, people touch their heads several dozen times an hour [KGM15, RMF20, MMG19]. There are many ways to interact, such as touching, stroking, scratching, rubbing, pulling, tugging, squeezing, and caressing, to name but a few. Behavioral sciences do not conclusively answer why people touch their faces,

© 2025 The Author(s). Computer Graphics Forum published by Eurographics - The European Association for Computer Graphics and John Wiley & Sons Ltd.
This is an open access article under the terms of the Creative Commons Attribution-NonCommercial License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited and is not used for commercial purposes.

yet the implications even extend to computer graphics. Due to the frequency and expressiveness of head-hand interactions, simulating them in facial animations would considerably improve user perception. Especially with the focus on photo-realistic avatars these days [QKS*24, MWSZ24, ZBT23, AXS*22, GKE*22], the relevance of authentic facial animations is further accentuated.

Only recently, attempts to incorporate head-hand interactions into facial animations have been proposed [SGPT23, WDX*24]. In particular, these approaches address two main challenges. Naturally, the simulation of interactions is the main emphasis, but three-dimensional tracking of the head and hands is also a prerequisite for realistic animations. Shimada et al. [SGPT23] and Wu et al. [WDX*24] are impressive in determining simulated 3D head and hand surfaces from a single monocular image. However, both neglect the fidelity of the interaction animation as they are based on the same rather coarse physics-based *surface* simulation. More sophisticated, detailed, and anatomically accurate *volumetric* physics-based simulations of heads have been explored in other contexts [SNF05, IKKP17, Con16, CZ24].

This work introduces a substantially improved physics-based simulation of head-hand interactions and designs more realistic interaction mechanisms. For instance, in contrast to the previous methods, we consider pulling interactions and the influence of the skull. Since this simulation is not real-time capable, we also learn a personalized neural network as an approximation. Both our simulation and the network process tracked head and hand surfaces and thus remain compatible with the tracking concepts of previous approaches [SGPT23, WDX*24]. Another contribution of this work is the creation of a dataset of real head-hand interactions. To this end, we built a multi-view rig with 16 high-resolution and synchronized video cameras with which we recorded several subjects. Unlike the only other comparable dataset available [SGPT23], we did not instruct the participants which head-hand interactions to perform. We simply asked the participants to perform arbitrary interactions and can, therefore, reproduce an even wider range of hand movements in our data. Among other things, we also capture skin pulling, which was previously ignored. Figure 1 is an exemplary illustration that shows a recorded *pulling* frame, the associated simulation, and the approximation by our neural network.

We evaluate our approach qualitatively using visual examples and the accompanying video of dynamic head-hand interaction animations. Furthermore, we conducted an extensive user study that confirms that our approach is perceived more naturally than previous ones. Quantitative experiments demonstrate that the neural approximation can be created in just a few hours and adapted to multiple human identities simultaneously. The trained network achieves around 50 frames-per-second (FPS) even on slower CPUs.

2 Related Work

In this section, we discuss three literature fields related to our approach. First, Section 2.1 presents physics-based facial animations in general. Next, Section 2.2 addresses recent developments focusing specifically on animated head-hand interactions. Finally, Section 2.3 examines work in which neural networks approximate physics-based simulations.

2.1 Physics-Based Facial Animations

Heuristic physics-based facial simulations have been developed for a long time and principally intend to compensate for shortcomings of *simpler* but popular facial animation methods like linear blendshapes [LAR*14]. For instance, artifacts like implausible contortions and self-intersections can be avoided by including volumetric and anatomical constraints. The pioneering work of Sifakis et al. [SNF05] is a volumetric physics-based facial simulation that runs on a personalized tetrahedral mesh. Unfortunately, the tetrahedral mesh can only be of limited resolution due to an associated dense optimization problem. With Phace [IKKP17, IKNDP16], an improved simulation concept has been introduced, which is also defined on a tetrahedral mesh but can handle higher resolutions and considers anatomical structures more precisely. In addition to a tetrahedral mesh, the art-directed muscle models [CF19, BCGF19, Con16] represent muscles as B-splines that steer facial expressions via trajectories of spline control points. A solely inverse model for determining the physical properties of faces was proposed in [KK19].

Thanks to increased computing capabilities, data-driven physics-based facial simulations have also become appealing recently. An example is the model of Yang et al. [YKZ*22] that learns to volumetrically animate a person's face from multi-view videos with the help of differentiable physics [DWM*21]. Although Yang et al. [YZC*23] extend the model to cover several identities, adding a novel identity requires five days of retraining and the inference of one frame runs multiple seconds. While faster alternatives exists [WBS23], generally, heuristic as well as data-driven physics-based simulations are not commonly used in real applications due to their complexity and computational effort. Other data-driven approaches include Animatomy [CEM*22], which represents muscles as curves, and the implicit model of Chandran et al. [CZ24]. The aforementioned data-driven simulations are not designed to handle collisions and external interactions.

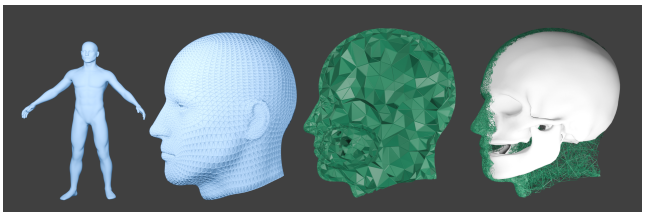
2.2 Head-Hand Interactions

All previously discussed simulations have in common that they are primarily aimed at facial animation, facial retargeting, or face reconstruction, but not at the simulation of external influences such as hands. Although models like Phace [IKKP17] are theoretically applicable in such scenarios, the non-trivial practical implementation of interactions has not happened until lately. Shimada et al. [SGPT23] propose the first head-hand interaction simulation, Decaf, and demonstrate how a neural network can learn the simulation while generalizing over the FLAME head model [LBB*17] and the MANO hand model [RTB17]. Decaf focuses on mapping a single RGB image to interaction deformations, using only a surface-based simulation that, in terms of quality and realism, falls short of the volumetric simulations discussed in Section 2.1. Also, the low resolution and the sometimes too smooth representation of heads in FLAME are often insufficient for demanding applications. Wu et al. [WDX*24] advance Decaf by an extended generalization to in-the-wild images. Unfortunately, the underlying simulation remains the same. Consequently, we focus on a more realistic simulation for personalized and more detailed head avatars.

© 2025 The Author(s).
Computer Graphics Forum published by Eurographics and John Wiley & Sons Ltd.

Variable	Description
$\mathbb{S},\mathbb{J},\mathbb{C}$	Tetrahedral meshes of soft tissue, jaw, and cranium
H,L,R,J,C	Surface meshes of head, left hand, right hand, jaw, and cranium
E_*	Energies
w_*,s_*	Scalar weights
C_*	Vertex targets of hand interactions
I_*	Set or dictionary of vertex indices
$*_t$	Indicates the time step t of a variable
$*^{\text{src}}$	Indicates the source src of a variable
$\mathcal{X}_T$	Sequence $(X_t)_{t=1}^T$ of T surface meshes X_t
$\tilde{X}$	Projection into PCA space of surface mesh X
`v,c,t`	A geom. element like a vertex `v`, a cylinder `c`, or a tetrahedron `t`
`func`	Denotes a function

Table 1: *The notation of the main concepts of Section 3.*



a) *b)* *c)* *d)*

Figure 2: *a) Full-body template which includes the head template surface H shown in b). c) Cross section of the connected tetrahedral meshes $\mathbb{S},\mathbb{J},\mathbb{C}$. d) The template jaw and cranium surfaces J,C embedded in the tetrahedral meshes.*

2.3 Approximating Physics-Based Simulations

As we accelerate our approach with efficient neural networks, we also give a brief overview of the literature on neural approximations of physics-based simulations. On the one hand, there are general methodologies [SWR*21] that also explicitly deal with interactions of two or more objects [RCCO22, RCPO21]. On the other hand, there are methods with a focus on bodies [SGOC20, CO18] or heads [WBS23]. For NePHIM, we adopt the general method of subspace neural physics [HDDN19] that is, in particular, computationally efficient for approximating simulations of interacting objects.

3 Method

This section first outlines the objectives of our approach (Section 3.1) and then presents the formal implementation (Sections 3.2–3.5). To support the reading flow, we slightly misuse the notation in the following derivations by denoting a mesh and the corresponding vector of stacked vertex positions with the same symbol. Table 1 gives a summary of the notation.

3.1 Objectives & Method Overview

We consider an animation at time T with tracked surfaces for the left hand L_T^{tra}, the right hand R_T^{tra}, and the head H_T^{tra} of a person. Given the corresponding neutral head surface mesh H as well as tracked sequences (consisting of all previous frames up to T) for the left hand $\mathcal{L}_T = \left(L_t^{\text{tra}}\right)_{t=1}^T$, right hand $\mathcal{R}_T = \left(R_t^{\text{tra}}\right)_{t=1}^T$, and head $\mathcal{H}_T = \left(H_t^{\text{tra}}\right)_{t=1}^T$, our first objective is to deform the tracked head surface mesh at time T, H_T^{tra}, to

$$H_T^{\texttt{phy}} = \texttt{phy}(\mathcal{R}_T, \mathcal{L}_T, \mathcal{H}_T, H), \tag{1}$$

such that head-hand interactions are resolved *realistically* through a physics-based simulation `phy`. Previous methods [SGPT23, WDX*24] determine deformations through a simple surface-based simulation [MHHR07] incorporating only constraints for the skin surface and (static) pushing hand interactions. We improve realism by implementing `phy` (Section 3.3) as a *volumetric* simulation that additionally respects

- long-term collision *paths* of pushing interactions,
- pulling hand interactions,
- and volumetric anatomical constraints.

Although the resulting $H_T^{\texttt{phy}}$ appears more natural (Section 4.4), our simulation `phy` is not real-time capable and, hence, potential applications are restricted. Therefore, our second objective is to train a neural network `net` (Section 3.5) that approximates $H_T^{\texttt{phy}}$ while being real-time capable even on CPUs.

3.2 Volumetric Template

In the remainder of this section, we will precisely state `phy` and `net`. However, as our approach is intended to reflect volumetric constraints, we first introduce a head template $(H,J,C,\mathbb{S},\mathbb{J},\mathbb{C})$ as the foundation of `phy`. The template includes the neutral head surface mesh $H \subset \mathbb{S}$ that encloses a soft tissue tetrahedral mesh $\mathbb{S}$. The two template surface meshes J,C form the corresponding skull as jaw and cranium and are filled with respective tetrahedral meshes $\mathbb{J}$ and $\mathbb{C}$. All tetrahedral meshes are connected, and the surface vertices of H can be addressed in $\mathbb{S}$ with the same indices. An experienced digital artist designed the template surfaces while the tetrahedral meshes were created with TetGen [Han15].

Figure 2 b–d visualize all template components; all dimensions can be found in Appendix A. The tessellation of H is aligned with a full-body avatar (Figure 2a), which is part of the code release to easily integrate NePHIM into other applications.

To register the volumetric template to a tracked person, we expect the neutral head surface H of this person to be known. Then, we reposition the skull components by a dense linear model that we trained on the computed tomography dataset of Achenbach et al. [ABG*18]. Formally, this model maps from the vertex positions of H to the vertex positions of the jaw J and the cranium C. The vertices of each tetrahedral mesh $\mathbb{S},\mathbb{J},\mathbb{C}$ are placed by radial basis function space warps [BK05] calculated *from* the respective enclosing surfaces in the template *to* the corresponding surfaces of the tracked person.

© 2025 The Author(s).
Computer Graphics Forum published by Eurographics and John Wiley & Sons Ltd.

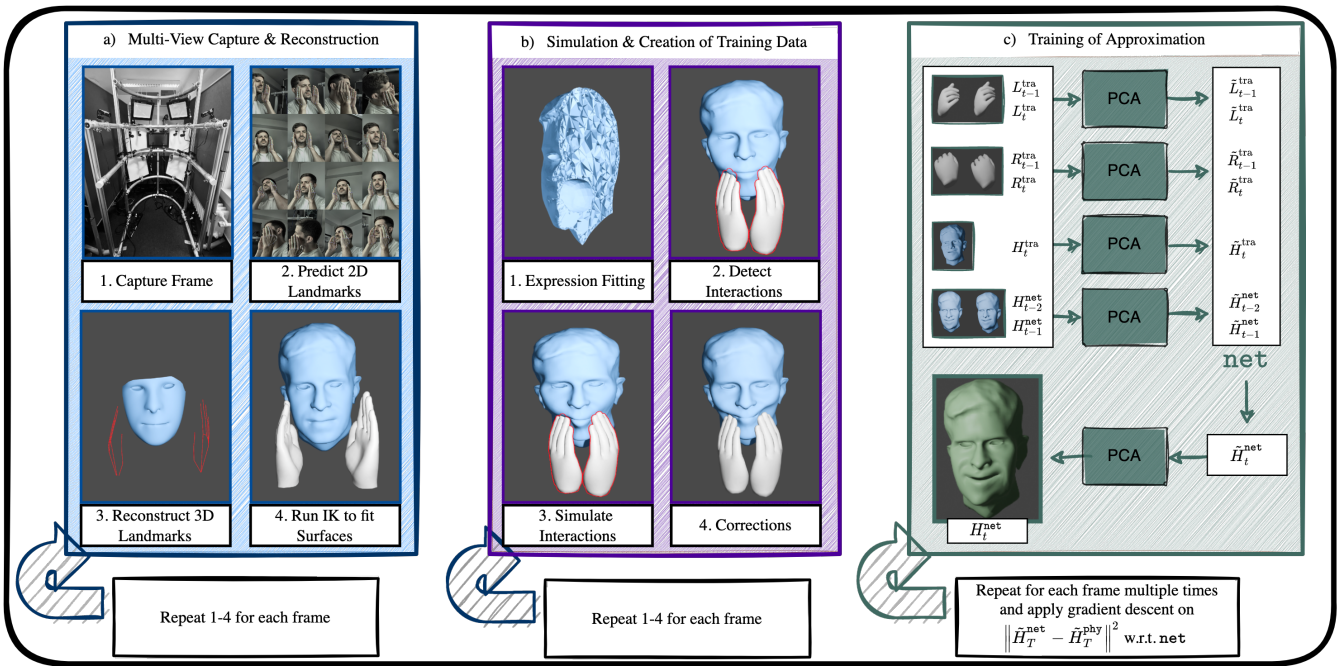


Figure 3: *Overview of the three stages of our approach. a) Data capturing as described in Section 4.1. b) All steps of our physics-based simulation* `phy` *as explained in Section 3.3. c) Efficient neural approximation* `net` *of* `phy` *as explained in Section 3.5.*

3.3 Simulation

Building on the volumetric template, we can now continue with the detailed introduction of our physics-based simulation `phy`. As Algorithm 1 outlines, `phy` conducts four steps that are described in separate subsections from Section 3.3.1 to Section 3.3.4. Figure 3b visualizes an exemplary cycle of all steps. Since we want to take long-term effects such as collision paths and skin pulling into account, it is not sufficient to consider only the last time step T to determine H_T^{phy}. Instead, we start at the beginning of the tracked sequences and run all four simulation steps consecutively for each time step t.

3.3.1 Expression Fitting

As the initial simulation step, we deform the neutral volumetric tetrahedral meshes $\mathbb{S}, \mathbb{J}$, and $\mathbb{C}$ in an anatomically plausible manner to fit the tracked surface H_t^{tra} instead of the neutral surface H. To this end, we minimize a constraint-based energy in the projective dynamics (PD) simulation framework [BML*14]. The first objective

$$E_{\text{target}}\left(H, H_t^{\text{tra}}\right) = \left\| H - H_t^{\text{tra}} \right\|^2 \tag{2}$$

attracts the surface vertices $H \subset \mathbb{S}$ of the soft tissue tetrahedral mesh towards the tracked head surface. The second objective

$$E_{\text{strain}}(\mathbb{S}) = \sum_{\mathtt{t} \in \mathbb{S}} s_{\mathtt{t}} \min_{\mathbf{R} \in SO(3)} \left\| \nabla(\mathtt{t}) - \mathbf{R} \right\|_F^2 \tag{3}$$

models strain for each soft tissue tetrahedron $\mathtt{t} \in \mathbb{S}$. Here, $\mathbf{R} \in SO(3)$ denotes the optimal rotation, $\nabla(\mathtt{t}) \in \mathbb{R}^{3\times 3}$ the deformation gradient of $\mathtt{t}$ (w.r.t. the neutral rest shape), $\|\cdot\|_F$ the Frobenius norm, and $s_{\mathtt{t}} \in (0, 1]$ is a stiffness value calculated as in [SGPT23]. In intuitive and simplified terms, the stiffness decreases the further a tetrahedron is located from the skull. Analogous to the soft tissue strain, we also add strain energies for the jaw $E_{\text{strain}}(\mathbb{J})$ and the cranium $E_{\text{strain}}(\mathbb{C})$. Overall, the weighted energy

$$\begin{aligned} E_{\text{tracked}}\left(H_t^{\text{tra}}, \mathbb{S}, \mathbb{J}, \mathbb{C}\right) = {} & w_{\text{tar}} E_{\text{target}}\left(H, H_t^{\text{tra}}\right) + \\ & w_{\mathbb{S}} E_{\text{strain}}(\mathbb{S}) + \\ & w_{\mathbb{J}} E_{\text{strain}}(\mathbb{J}) + \\ & w_{\mathbb{C}} E_{\text{strain}}(\mathbb{C}) \end{aligned} \tag{4}$$

is minimized. To reflect that both jaw and cranium are rigid, we set the weights $w_{\mathbb{J}}$ and $w_{\mathbb{C}}$ to a high value compared to w_{tar} and $w_{\mathbb{S}}$ and apply a constant stiffness of one. The values of all weights and other simulation parameters can be found in Appendix B. The outputs of the optimization are the tracked tetrahedral meshes

$$\left(\mathbb{S}^{\text{tra}}, \mathbb{J}^{\text{tra}}, \mathbb{C}^{\text{tra}}\right) = \underset{\mathbb{S},\mathbb{J},\mathbb{C}}{\operatorname{argmin}} E_{\text{tracked}}\left(H_t^{\text{tra}}, \mathbb{S}, \mathbb{J}, \mathbb{C}\right). \tag{5}$$

Please note that although there are more detailed simulation methods than PD [LFS*20, IKKP17, YKZ*22], these are often more complex and cannot outweigh the efficiency of PD in our use case.

3.3.2 Detect Interactions

Subsequently, we detect pushing and pulling head-hand interactions and translate them into target positions of the head vertices C_{push} and C_{pull}, which we will simulate in the next step (Section 3.3.3).

Pushing Interactions We first explain how we handle pushing in `phy`. Previous approaches [SGPT23, WDX*24] would simply iterate over the vertices of the head surface H_t^{tra} and if a vertex enters either the left hand L_t^{tra} or the right hand R_t^{tra}, it is moved in the

© 2025 The Author(s).
Computer Graphics Forum published by Eurographics and John Wiley & Sons Ltd.

Algorithm 1 Volumetric Physics-Based Simulation $\texttt{phy}$

Function $\texttt{phy}(\mathcal{R}_T, \mathcal{L}_T, \mathcal{H}_T, H)$

// Section 3.2 Register Template to Neutral
Register volumetric template (Figure 2) to obtain the tracked person's volumetric head description $(H, J, C, \mathbb{S}, \mathbb{J}, \mathbb{C})$.

$t = 1$
while $t \leq T$ **do**

// Section 3.3.1 Expression Fitting
Step 1 Determine the tracked tetrahedral meshes $\mathbb{S}^{\text{tra}}, \mathbb{J}^{\text{tra}}, \mathbb{C}^{\text{tra}}$ by aligning $\mathbb{S}, \mathbb{J}, \mathbb{C}$ with the tracked head surface H_t^{tra} as described in Equation (5).

// Section 3.3.2 Detect Interactions
Step 2 Determine the push and pull target positions $C_{\text{push}}, C_{\text{pull}}$ as described in Algorithm 2 and Algorithm 3, respectively.

// Section 3.3.3 Simulate Interactions
Step 3 Determine the interaction tetrahedral meshes $\mathbb{S}^{\text{int}}, \mathbb{J}^{\text{int}}, \mathbb{C}^{\text{int}}$ by applying the push and pull targets $C_{\text{push}}, C_{\text{pull}}$ to H_t^{tra} as described in Equation (9).

// Section 3.3.4 Corrections
Step 4 Determine the corrected tetrahedral meshes $\mathbb{S}^{\text{cor}}, \mathbb{J}^{\text{cor}}, \mathbb{C}^{\text{cor}}$ by resolving remaining collisions I_{corr} as described in Equation (11). Extract H_t^{phy} from $\mathbb{S}^{\text{cor}}$.

$t = t + 1$

// Return the simulated head surface
return H_T^{phy}

Algorithm 2 Pushing Interaction

Function $\texttt{push}(H_{t-1}^{\text{phy}}, H_t^{\text{tra}}, L_{t-1}^{\text{tra}}, L_t^{\text{tra}}, R_{t-1}^{\text{tra}}, R_t^{\text{tra}})$

// Initialize linear movement directions
$H_{\text{dir}} = H_t^{\text{tra}} - H_{t-1}^{\text{phy}}$
$L_{\text{dir}} = L_t^{\text{tra}} - L_{t-1}^{\text{tra}}$
$R_{\text{dir}} = R_t^{\text{tra}} - R_{t-1}^{\text{tra}}$
// Initialize push targets
$C_{\text{push}} = \{\}$
$I = \{\}$

// Iterate over linear movements
for $\epsilon = 0; \epsilon \leq 1; \epsilon \mathrel{+}= \Delta\epsilon$ **do**

// Iterate over head surface vertices
for $\mathsf{v}_i^H \in \left(H_{t-1}^{\text{phy}} + \epsilon \cdot H_{\text{dir}}\right)$ **do**

// Find collisions with left hand
if v_i^H collides with $\left(L_{t-1}^{\text{tra}} + \epsilon \cdot L_{\text{dir}}\right)$ and $i \notin I$ **then**

// Find nearest neighbor in current left hand
$\mathsf{v}_{j,\epsilon}^L = \texttt{nn}\left(\mathsf{v}_i^H, L_{t-1}^{\text{tra}} + \epsilon \cdot L_{\text{dir}}\right)$
// Add same vertex of final left hand as target position
Add $\left(\mathsf{v}_{j,t}^L, i\right)$ to C_{push}
Add i to I
Repeat the same **if**-clause for the right hand

// Return the push targets
return C_{push}

direction of the corresponding inverted normal until the collision is resolved. Unfortunately, this strategy largely ignores temporal dependencies, and the normal direction only provides an imprecise collision resolution.

For this reason, we rely on the linear movements between H_{t-1}^{phy} and H_t^{tra}, L_{t-1}^{tra} and L_t^{tra}, as well as R_{t-1}^{tra} and R_t^{tra}, i.e., between the previous simulated frame and the current tracked frame, as formally described in Algorithm 2. Expressed in words, we check at short intervals ϵ between the time steps $t-1$ and t whether one of the two hands touches a vertex of the head. If so, the head vertex is dragged from the initial point of contact with the hand at ϵ to the same point on the hand at time t. Please see Figure 4a for a visual explanation. Our way of resolving hand pushing is more *natural* and incorporates long-term effects per construction. Although there are more involved and time-consuming forms of continuous collision detection, these did not yield substantially better results in our experiments.

Pulling Interactions Pulling is considerably more challenging and has not been addressed in prior approaches. We present a heuristic in Algorithm 3 that does not require cumbersome friction calculations but, unfortunately, still has an elaborated notation. Yet, the foundational idea of our heuristics can easily be put into words. First, we form cylinders with radius r between the fingertips of all fingers (index, middle, ring, little) and the thumb as illustrated in Figure 4b. Then, for each cylinder, we determine whether it grabs, i.e., has shortened in length from time step $t-1$ to time step t. If so, and if the length falls below a minimum l_{min}, all head vertices inside the cylinder at time t are marked as *pulled*. We maintain a dictionary I_{pull} over time that stores a set of the *pulled* vertices for each finger.

The target positions of the *pulled* vertices C_{pull} are calculated so that they form smooth ridges within the cylinders (see Figure 4b). The shape of the ridges imitates the skin's natural deformation due to pinching. A *pulled* vertex is unmarked once the corresponding cylinder exceeds l_{min}, i.e., the finger no longer grabs.

3.3.3 Simulate Interactions

For applying the previously determined push and pull targets C_{push} and C_{pull} to the tracked head H_t^{tra}, we again make use of a PD simulation on the fitted tetrahedral meshes $\mathbb{S}^{\text{tra}}, \mathbb{J}^{\text{tra}}$, and $\mathbb{C}^{\text{tra}}$ (Section 3.3.1). Here, we establish anatomical plausibility similar as before by adding strain constraints $E_{\text{strain}}\left(\mathbb{S}^{\text{tra}}\right)$, $E_{\text{strain}}\left(\mathbb{J}^{\text{tra}}\right)$, and $E_{\text{strain}}\left(\mathbb{C}^{\text{tra}}\right)$ to the simulation. Also as before, we add

$$E_{\text{target}}\left(H^{\text{tra}}, H_t^{\text{tra}}\right) = \left\| H^{\text{tra}} - \left(H_t^{\text{tra}} + \frac{\alpha}{s}\left(H_{t-1}^{\text{phy}} - H_{t-2}^{\text{phy}} \right) \right) \right\|^2 \quad (6)$$

to draw the surface vertices $H^{\text{tra}} \subset \mathbb{S}^{\text{tra}}$ of the soft tissue to the tracked surface. This time, however, including damped velocities of the head, where s denotes the size of a time step. A low damping factor α adds natural-looking dynamic effects to the interactions.

New to the simulation are the target constraints

$$\begin{aligned} E_{\text{push}}\left(H^{\text{tra}}, C_{\text{push}}\right) &= \sum_{(\mathsf{p},i) \in C_{\text{push}}} \|\mathsf{p} - \mathsf{v}_i\|^2, \\ E_{\text{pull}}\left(H^{\text{tra}}, C_{\text{pull}}\right) &= \sum_{(\mathsf{p},i) \in C_{\text{pull}}} \|\mathsf{p} - \mathsf{v}_i\|^2, \end{aligned} \quad (7)$$

which draw interacting vertices $\mathsf{v}_i \in H^{\text{tra}}$ to their precalculated target

© 2025 The Author(s).
Computer Graphics Forum published by Eurographics and John Wiley & Sons Ltd.

Algorithm 3 Pulling Interaction

Notation

$\mathsf{c}_t^{L,f}$	Cylinder of finger f of the left hand L at timestep t
`len`	Length of a cylinder
$I_{\text{pull}}[L,f]$	Dictionary entry of key L,f, i.e., a set

Function `pull`$(H_t^{\text{tra}}, L_{t-1}^{\text{tra}}, L_t^{\text{tra}}, R_{t-1}^{\text{tra}}, R_t^{\text{tra}}, I_{\text{pull}})$

// Initialize pull targets
$C_{\text{pull}} = \{\}$
// Check if new vertices are pulled per cylinder
for $f = 1; f \leq 4; f \mathrel{+}= 1$ **do**
  // Pull only if cylinder gets smaller and is small enough
  if $\texttt{len}\left(\mathsf{c}_t^{L,f}\right) < \texttt{len}\left(\mathsf{c}_{t-1}^{L,f}\right)$ and $\texttt{len}\left(\mathsf{c}_t^{L,f}\right) < l_{\text{min}}$ **then**
    // Check for each head vertex if inside cylinder
    for $\mathsf{v}_i \in H_t^{\text{tra}}$ **do**
      if v_i inside $\mathsf{c}_t^{L,f}$ **then**
        Append i to $I_{\text{pull}}[L,f]$
// Check if vertices are no longer pulled per cylinder
for $f = 1; f \leq 4; f \mathrel{+}= 1$ **do**
  if $\texttt{len}\left(\mathsf{c}_t^{L,f}\right) \geq l_{\text{min}}$ **then**
    $I_{\text{pull}}[L,f] = \emptyset$
// Calculate target positions of pulled vertices per cylinder by
// creating a `ridge` *per cylinder as defined in Appendix C*
for $f = 1; f \leq 4; f \mathrel{+}= 1$ **do**
  Append $\texttt{ridge}\left(I_{\text{pull}}[L,f], H_t^{\text{tra}}, \mathsf{c}_t^{L,f}\right)$ to C_{pull}
Repeat same procedure for the right hand
// Return the pull targets
return C_{pull}

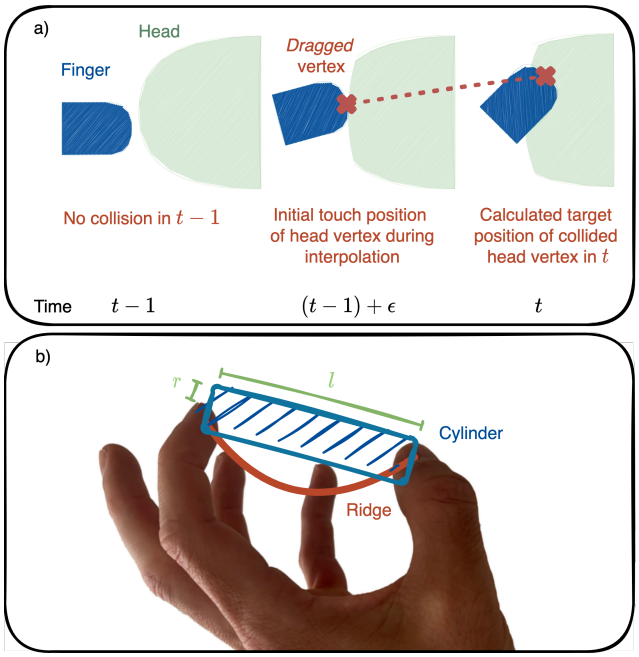


Figure 4: *a) Visualization of* pushing *as described in Section 3.3.2 and Algorithm 2. Here, ϵ is a substep between the time steps $t-1$ and t. b) Illustration of a finger cylinder with radius r, length l, and an exemplary ridge shape that is used for* pulling *as described in Algorithm 3.*

position p. Overall, the weighted energy

$$\begin{aligned} E_{\text{inter}}\left(C_{\text{push}}, C_{\text{pull}}, H_t^{\text{tra}}, \mathbb{S}^{\text{tra}}, \mathbb{J}^{\text{tra}}, \mathbb{C}^{\text{tra}}\right) = \\ w_{\text{push}} E_{\text{push}}\left(H^{\text{tra}}, C_{\text{push}}\right) + \\ w_{\text{pull}} E_{\text{pull}}\left(H^{\text{tra}}, C_{\text{pull}}\right) + \\ w_{\text{tar}} E_{\text{target}}\left(H^{\text{tra}}, H_t^{\text{tra}}\right) + \\ w_{\mathbb{S}} E_{\text{strain}}\left(\mathbb{S}^{\text{tra}}\right) + \\ w_{\mathbb{J}} E_{\text{strain}}\left(\mathbb{J}^{\text{tra}}\right) + \\ w_{\mathbb{C}} E_{\text{strain}}\left(\mathbb{C}^{\text{tra}}\right) \end{aligned} \tag{8}$$

is minimized, where we again set the weights $w_{\mathbb{J}}$ and $w_{\mathbb{C}}$ to a high value for approximating a rigid skull. Likewise, the weights w_{push} and w_{pull} are set to a high value to enforce the target positions, but lower as $w_{\mathbb{J}}, w_{\mathbb{C}}$. By balancing the previously mentioned weights, we achieve a more natural simulation since the bones do not bend in the case of tracking errors and too deeply penetrating hands. The outputs of the optimization are the interaction tetrahedral meshes

$$\begin{aligned} \left(\mathbb{S}^{\text{int}}, \mathbb{J}^{\text{int}}, \mathbb{C}^{\text{int}}\right) = \underset{\mathbb{S}^{\text{tra}}, \mathbb{J}^{\text{tra}}, \mathbb{C}^{\text{tra}}}{\text{argmin}} E_{\text{inter}}\left(C_{\text{push}}, C_{\text{pull}}, \right. \\ \left. H_t^{\text{tra}}, \mathbb{S}^{\text{tra}}, \mathbb{J}^{\text{tra}}, \mathbb{C}^{\text{tra}}\right). \end{aligned} \tag{9}$$

3.3.4 Corrections

The preceding steps of `phy` do not fully resolve all head-hand collisions. For instance, the last step in Section 3.3.3 allows soft tissue vertices that previously did not collide to move inside the hands. To correct most remaining colliding vertices, summarized with their indices in I_{corr}, we perform the previous PD simulation again but add an additional constraint. This constraint

$$E_{\text{corr}}\left(\mathbb{S}^{\text{int}}, I_{\text{corr}}\right) = \sum_{i \in I_{\text{corr}}} \left\| \texttt{nn}\left(\mathsf{v}_i, L_t^{\text{tra}}, R_t^{\text{tra}}\right) - \mathsf{v}_i \right\| \tag{10}$$

draws each colliding vertex $\mathsf{v}_i \in \mathbb{S}^{\text{int}}$ to the nearest neighbor $\texttt{nn}\left(\mathsf{v}_i, L_t^{\text{tra}}, R_t^{\text{tra}}\right)$ of v_i on the left or right hand $L_t^{\text{tra}}, R_t^{\text{tra}}$. The outputs of the optimization are the corrected tetrahedral meshes

$$\begin{aligned} \left(\mathbb{S}^{\text{cor}}, \mathbb{J}^{\text{cor}}, \mathbb{C}^{\text{cor}}\right) = \underset{\mathbb{S}^{\text{int}}, \mathbb{J}^{\text{int}}, \mathbb{C}^{\text{int}}}{\text{argmin}} w_{\text{corr}} E_{\text{corr}}\left(\mathbb{S}^{\text{int}}, I_{\text{corr}}\right) + \\ E_{\text{inter}}\left(C_{\text{push}}, C_{\text{pull}}, H_t^{\text{tra}}, \mathbb{S}^{\text{int}}, \mathbb{J}^{\text{int}}, \mathbb{C}^{\text{int}}\right). \end{aligned} \tag{11}$$

The deformed surface $H_t^{\text{phy}} = \texttt{phy}(\mathcal{R}_t, \mathcal{L}_t, \mathcal{H}_t, H) \subset \mathbb{S}^{\text{cor}}$ can now be extracted as the outer boundary of the soft tissue mesh. After the four steps of `phy` described in Sections 3.3.1–3.3.4 have been carried out consecutively for all time steps through to T, H_T^{phy} is obtained.

3.4 Recursive Formulation

The previous description of `phy` serves the intuitive derivation, but suggests that the computational effort increases linearly with each additional frame. However, this is not the case, since by the design

© 2025 The Author(s).
Computer Graphics Forum published by Eurographics and John Wiley & Sons Ltd.

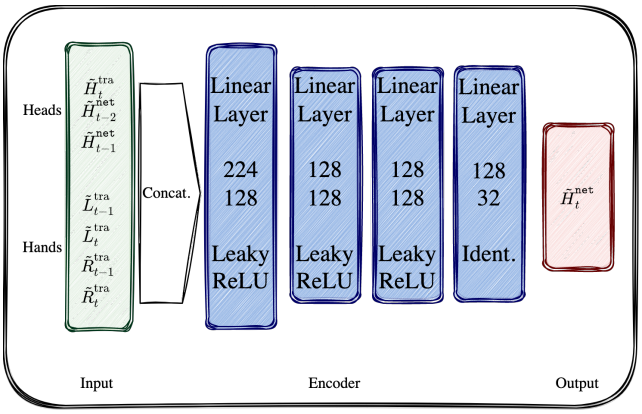


Figure 5: *An overview of the efficient network architecture of* `net`. *Basically, a simple MLP with only 65536 parameters.*

of `phy`, we can rewrite Equation (1) recursively as

$$\begin{aligned} H_T^{\texttt{phy}} = \texttt{phy}(&L_T^{\text{tra}}, R_T^{\text{tra}}, H_T^{\text{tra}}, \\ &L_{T-1}^{\text{tra}}, R_{T-1}^{\text{tra}}, H_{T-1}^{\texttt{phy}}, \\ &H_{T-2}^{\texttt{phy}}). \end{aligned} \tag{12}$$

Hence, we can reuse simulated frames instead of always simulating all time steps.

3.5 Neural Simulation Approximation

As the derivations in the previous sections already indicate, `phy` is not real-time capable. Therefore, we construct `net`, a neural network that can be evaluated even on CPUs with 50 FPS (Table 4) and that closely approximates `phy`. From the wide corpus of techniques that already exist for approximating physic-based simulations (Section 2.3), we adapt subspace neural physics (SNP) [HDDN19] to our needs. Here, we only explain our adapted architecture of `net`, as the original publication extensively describes the training algorithm and we do not modify it.

The principle idea of SNP is to project all inputs and outputs into smaller linear subspaces (e.g. using principal component analysis (PCA)) and to train `net` on the projection. In the following, the pedant of a variable in its respective subspace is referenced with an overlying tilde. The inputs of `net` with regard to `phy` as defined in Equation (12) are

$$L_T^{\text{tra}}, R_T^{\text{tra}}, H_T^{\text{tra}}, L_{T-1}^{\text{tra}}, R_{T-1}^{\text{tra}}, H_{T-1}^{\texttt{net}}, H_{T-2}^{\texttt{net}}. \tag{13}$$

Consequently, we have to create PCA subspaces for the tracked left hand, the tracked right hand, the tracked head, and the simulated head, respectively. The overall training goal is then to minimize

$$\min_{\texttt{net}} \left\| \tilde{H}_T^{\texttt{net}} - \tilde{H}_T^{\texttt{phy}} \right\|^2, \tag{14}$$

where

$$\begin{aligned} \tilde{H}_T^{\texttt{net}} = \texttt{net}(&\tilde{L}_T^{\text{tra}}, \tilde{R}_T^{\text{tra}}, \tilde{H}_T^{\text{tra}}, \\ &\tilde{L}_{T-1}^{\text{tra}}, \tilde{R}_{T-1}^{\text{tra}}, \tilde{H}_{T-1}^{\texttt{net}}, \\ &\tilde{H}_{T-2}^{\texttt{net}}). \end{aligned} \tag{15}$$

A visual illustration of the inputs and outputs of `net` is depicted in Figure 3c and our architecture can be found in Figure 5. To recover $H_T^{\texttt{net}}$ from $\tilde{H}_T^{\texttt{net}}$, the PCA of the simulated heads is applied. By selecting an appropriate number of components of the subspace, we prevent the loss of geometric details.

4 Results

The result section is organized as follows. First, we outline how we capture and process real head-hand interactions to form training and test data (Section 4.1). The same subsection also contains a description of the resulting dataset and details on training and evaluation protocols. In Section 4.2, we discuss qualitative characteristics of the simulation `phy` and the approximation `net` using visual examples. In Section 4.3, we examine quantitative characteristics and also take a closer look at running times as well as training times. Finally, we present the results of a user study (Section 4.4) that supports the more natural perception of our approach.

4.1 Dataset & Training

To capture real head-hand interactions, we use a multi–view rig consisting of 16 synchronized and calibrated XIMEA [Xim24] RGB cameras generating 12-megapixel images at 20 FPS. In each captured image, we predict 2D landmarks for both hands and head using existing tracking methods [BT17, ZBV*20]. For the hands, a landmark is predicted for each joint, each fingertip, and the wrists. For heads, we only capture the contours of the eyes and the mouth, as can be seen in Figure 1. From the 2D landmarks, we generate 3D landmarks per frame using a basic bundle adjustment algorithm.

Since our simulation `phy` is conceptualized to work on tracked surfaces, the last step in the capturing pipeline is to fit appropriate template surfaces to the 3D landmarks. Regarding the head, we initially create a high-resolution personalized head avatar for the recorded person with an automated 3D reconstruction and (nonlinear) template fitting pipeline [WAB*20]. Afterward, we add linear blendshapes to the avatar by an *automated* volumetric deformation transfer [WSB24, SP04] of a set of template blendshapes. The template blendshapes represent the 52 ARKit expressions [App24] and were *manually* sculpted *once* by a professional digital artist.[†] Finally, we optimize per frame a set of corresponding blendshape weights, a translation vector, and a rotation matrix to fit the head surface to the respective 3D landmarks. Regarding the hands, we adopt a similar approach. Here, however, we do not use a personalized hand model but optimize the pose and shape parameters of the MANO [RTB17] hand model to match the respective 3D landmarks. Contrary to the pose parameters, the shape parameters are the same for each frame. We use gradient descent as the optimizer for the surface fittings. Figure 3a illustrates all steps of the capturing pipeline.

The dataset we compiled contains up to 10 recordings of each of 8 individuals. The individuals are Caucasian males aged 26 to 54 with a body mass index ranging from slightly underweight to obese. Each recording lasts approximately 30 seconds and captures arbitrary head-hand interactions. In particular, we did not instruct

[†] The template blendshapes are part of the code release.

© 2025 The Author(s).
Computer Graphics Forum published by Eurographics and John Wiley & Sons Ltd.

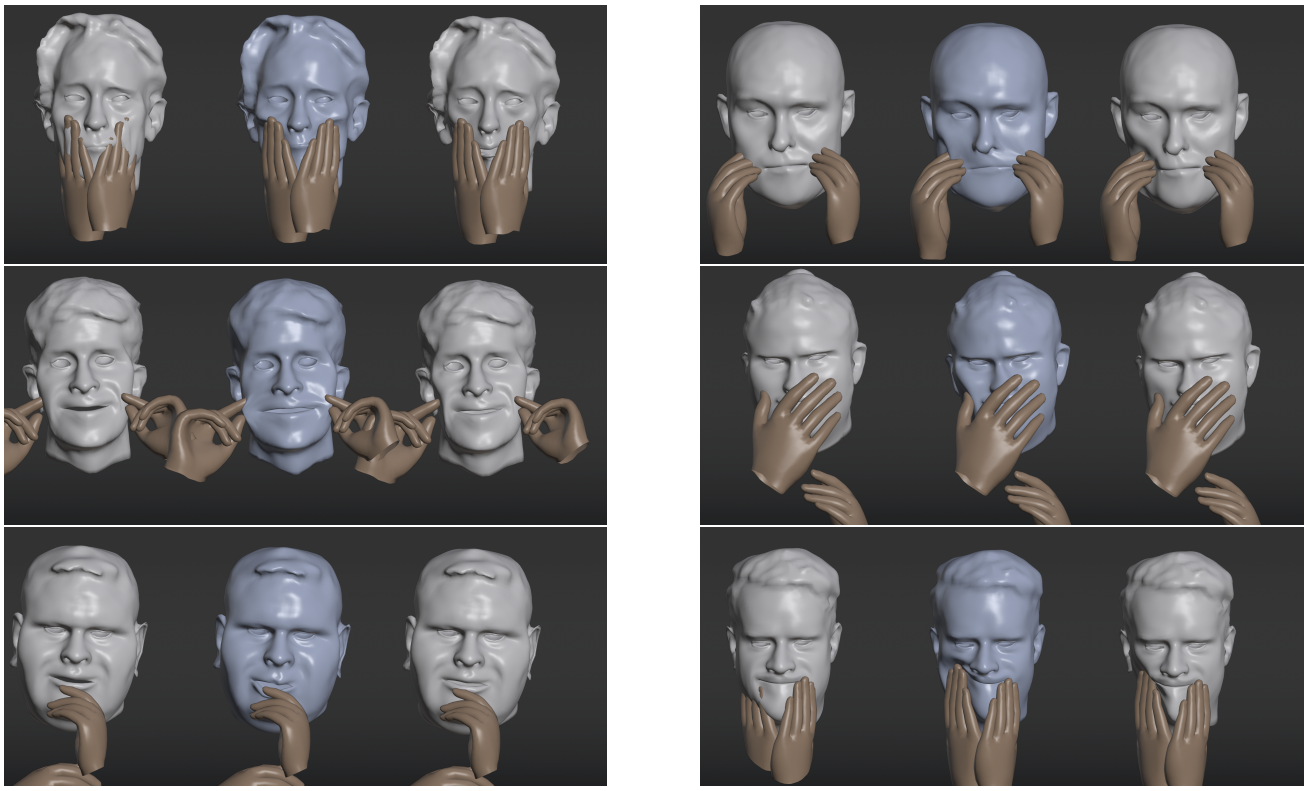


Figure 6: *The figure shows examples of our simulation* `phy` *and compares them to the tracked surfaces as well as the simulation of Decaf [SGPT23]. In the top left, for example, the advantage of simulating the skull becomes apparent near the cheekbone. In the top right image, a pulling interaction is shown and the lower images demonstrate the importance of (time-dependent) collision paths.*

the individuals on which hand movements or facial expressions they should perform. Appendix D summarizes the frequency and the types of interactions. Overall, we captured, reconstructed, and simulated around 50 000 frames for this work. All of the following experiments concerning the neural network `net` are always stated as an average of five runs, and we uniformly (i.e., non-consecutively) draw random train/test splits (90%/10%) for each run. All PCA subspaces have 32 components, which is sufficient in our case as we do not intend to generalize over large head or hand models. We rebuilt the subspaces for each run on the respective training data, and if several individuals are part of an experiment, we form joint subspaces. Neural networks and the inference of PCAs are implemented with PyTorch [PGM*19] while PCA subspaces are constructed with the default implementation of Scikit [PVG*11].

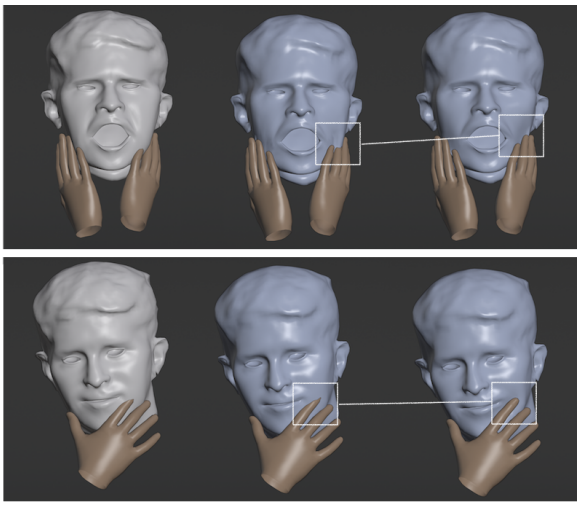
4.2 Qualitative Evaluation

Figure 6 (and additional examples in Appendix E) display instances of the simulation `phy` in comparison to the tracked surfaces as well as the simulation of the current state-of-the-art Decaf [SGPT23]. Please note that we implemented the latter simulation ourselves as the announced implementations are not (yet) available. Decaf results sometimes appear slightly different to those from [SGPT23], which mainly stems from the fact that our head avatars are more detailed than FLAME [LBB*17] and that we did not instruct the recorded persons which head-hand interactions they should perform. In the shown examples, it is especially striking that our temporal processing of hand pushes leads to effects such as a bent nose, a pushed-up mouth corner, or even a pushed-down lip. Moreover, the pulling of skin is readily recognizable and appears natural. None of these effects can be observed with the other methods. The accompanying video demonstrates the advantages of our method for dynamic scenes.

Besides the more general examples, we show further visual comparisons to inspect individual stages of our approach. Figure 7 emphasizes the necessity of the correction step of `phy` (Section 3.3.4) while Figure 8 stresses the relevance of modeling temporal effects in our simulation. However, Figure 8 not only exhibits the impact of temporal effects on our simulation but also contrasts our simulation without temporal effects to the Decaf [SGPT23] simulation. Finally, Figure 9 underpins the advantage of a volumetric anatomy simulation by contrasting bendable and rigid bones.

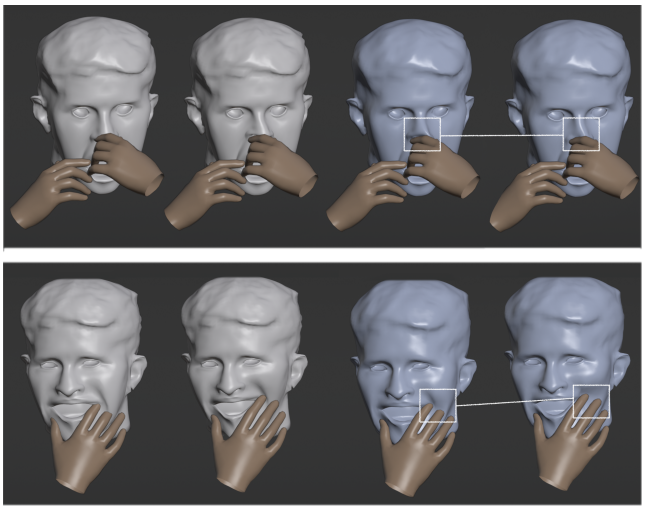
Examples of our simulation `phy` along with the learned approximation `net` are depicted in Figure 10. For this purpose, we trained `net` on all identities in our dataset simultaneously. Although minor discrepancies can be recognized, these do not appear to be decisive for visual perception. Moreover, the quality of the approximation is

© 2025 The Author(s).
Computer Graphics Forum published by Eurographics and John Wiley & Sons Ltd.



Tracked *Without* *With*

Figure 7: *Examples of our simulation* `phy` *without and with the correction step (Section 3.3.4). Errors due to the missing correction step can accumulate over time.*



Tracked *Decaf [SGPT23]* *Without* *With*

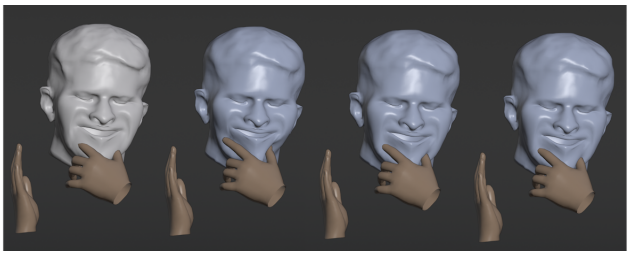
Figure 8: *Examples of our simulation* `phy` *without and with temporal effects as well as the Decaf [SGPT23] simulation.*

not affected by whether the head-hand interactions are pushing or pulling. Again, the accompanying video contains further examples.

4.3 Quantitative Evaluation

4.3.1 Accuracy

This section mainly investigates the quantitative properties of the network `net` and the simulation `phy`. To begin with, we have a look at the approximation accuracy of `net`. For this purpose, Table 2 summarizes average train and test subspace errors (mean squared error) of `net` as well as the average and maximum reconstruction errors on the actual surfaces (ℓ^2 error). There is also a breakdown by the number of identities with which we trained and tested `net`. The table indicates that the reconstruction test errors are never greater than a millimeter on average, and our implementation of `net` has sufficient capacity to generalize over several identities. Moreover, the likewise small maximum reconstruction errors indicate that all kinds of simulated deformations can be adequately approximated by `net` without hallucinating non-existent interactions.



Tracked *0.0* *0.5* *1.0*

Figure 9: *Example of our simulation* `phy` *applying either 0%, 50%, or 100% of the bone weights* $w_{\mathbb{J}}, w_{\mathbb{C}}$.

Dataset	# Identities	Subspace *Mean MSE*	Reconstruction *Mean* ℓ^2	*Max* ℓ^2
Train	One	0.011	0.02 cm	0.11 cm
	Eight	0.041	0.04 cm	0.18 cm
Test	One	0.052	0.09 cm	0.23 cm
	Eight	0.056	0.10 cm	0.35 cm

Table 2: *Train and test errors of the neural approximation* `net` *of the simulation* `phy`*. The table is separated by the number of identities* `net` *was trained on. The errors stated for* one *identity are the average over separate networks for all identities in our dataset.*

4.3.2 Plausibility

In Table 3, we compare the plausibility of our network `net`, our simulation `phy`, and the simulation of Decaf [SGPT23] by means of quantitative metrics introduced in Shimada et al. [SGPT23]. Among them is the *Non Collisions* metric, which captures the number of collision-free frames after applying a method. We also state the *Collision Distance*, which measures the average per-vertex depth of the remaining collisions. We complement the existing metrics with the *Deformation Distance*, which calculates the average per-vertex deformation of the tracked head caused by a method. Table 3 indicates that all methods significantly reduce the number of colliding frames, and the remaining collisions are less deep. Although Decaf appears to better resolve collisions at first glance, this is to be expected, as it is able to bend bones unnaturally, for instance. This expectation is also supported by the *Deformation Distance*, which demonstrates that Decaf tends to apply larger deformations in general.

4.3.3 Timings

Table 4 summarizes the average running times of `phy` and `net`. On the one hand, with a runtime of 876 ms per frame on an AMD

© 2025 The Author(s).
Computer Graphics Forum published by Eurographics and John Wiley & Sons Ltd.

Figure 10: *Examples of our simulation* `phy` *along with the tracked surfaces as well as the learned neural approximation* `net` *(trained on all identities in our dataset). The quality of the approximation is independent of whether it is a pushing or a pulling interaction.*

Method	Non Collisions	Collision Dist.	Deformation Dist.
Tracked	53 %	1.20 cm	0.00 cm
`phy`	69 %	0.19 cm	0.11 cm
`net`	68 %	0.20 cm	0.09 cm
Decaf	8 %	0.09 cm	0.16 cm

Table 3: *Plausibility metrics to compare our simulation* `phy`*, the Decaf simulation [SGPT23], and our network* `net` *to the tracked input. Non Collisions is the percentage of frames without collisions, Collision Distance measures the average per-vertex penetration depth of the remaining collisions, and Deformation Distance indicates the average per-vertex deformation by the respective method.*

Input Size	`phy` CPU	`net` CPU	`net` GPU
1 ×	876 ms	19.2 ms	5.1 ms
2 ×	2248 ms	34.6 ms	7.3 ms
4 ×	6553 ms	79.2 ms	9.4 ms

Table 4: *The average inference times of the simulation* `phy` *and the neural approximation* `net` *depending on the input size, i.e., the number of surface vertices, the number of volumetric vertices, and the size of the PCA subspaces.*

Tracking + `net`	Decaf	Dice
15.3 ms / 66 FPS	88 ms / 11.5 FPS	19590 ms / 0.05 FPS

Table 5: *GPU inference times of our neural approximation* `net` *compared to Decaf [SGPT23] and Dice [WDX*24]. For a fair comparison, since Decaf and Dice include tracking components, we added Mediapipe's [BT17, ZBV*20] head and hand tracking ahead of our network. The times were measured on a 128-core AMD Ryzen CPU and a NVIDIA A6000 GPU.*

Ryzen Threadripper PRO 3995WX, `phy` is evidently not realtime-capable. On the other hand, `net` can be executed not only on a consumer-grade GPU (NVIDIA RTX 3090) but also on a weaker CPU (Intel i5 12600K) with more than 50 FPS. In comparison, our implementation of the simulation of Decaf [SGPT23] runs in 178 ms per frame on the Threadripper CPU. The entire pipeline, as shown in Figure 3, from data acquisition to training `net` only takes about 20 hours for eight identities and 3.5 hours for one identity. We trained on a NVIDIA A6000 GPU for four hours (eight identities) or one and a half hours (one identity). The short training time is mainly due to the efficient network architecture.

The aforementioned running times depend on the resolution of the underlying template. Although our template is already able to capture detailed deformations, Table 4 also shows that we can still efficiently execute `net` if the template is further refined. To that end, we doubled and quadrupled the number of surface and volumetric template vertices (remeshing) as well as the size of the PCA subspaces. On the CPU, `net` still runs at interactive rates if the resolution is doubled, whereas on the GPU, even a quadrupling is feasible. Nevertheless, as can also be seen in Table 4, the running time of the simulation increases substantially, and so does the time needed for generating training data.

We have intentionally designed `net` to be independent of any particular tracking method, and the running times stated in Table 4 imply that it can readily be integrated with other applications. However, in order to compare the inference times with those of Decaf [SGPT23] and Dice [WDX*24], we trained a slightly modified `net`. For this modification, we input 2D head and hand landmarks tracked by Mediapipe [BT17, ZBV*20] on the most frontal camera of our multi–view rig instead of PCA subspace representations of the undeformed surfaces. The test errors of this network are close

© 2025 The Author(s).
Computer Graphics Forum published by Eurographics and John Wiley & Sons Ltd.

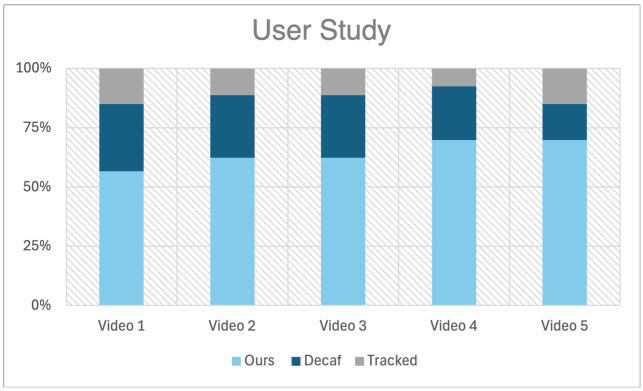


Figure 11: *A user study among 53 participants supports that our approach is recognized as more natural. For each video shown in the user study, our approach received the most votes.*

to those stated in Table 2 (see Appendix F). Since the other two methods are not intended to run on the CPU, we only compare the GPU (A6000) running times listed in Table 5. It becomes apparent that, including visual tracking, our approach is still around 6 times faster than Dice and 1300 times faster than Decaf.

4.4 User Study

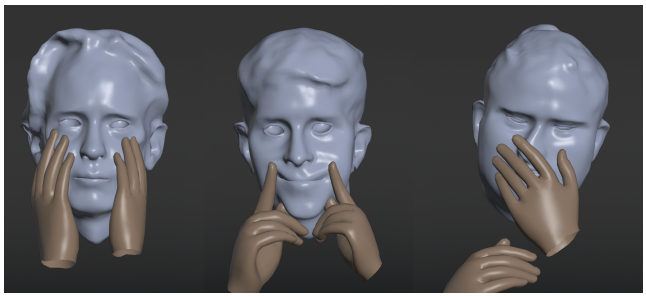
To support the qualitative results, we conducted an online user study with 53 participants from two universities. Each participant watched five random example videos that compared the tracked surfaces, the simulated surfaces of Shimada et al. [SGPT23], and our (neural) approximated surfaces as in Figure 6. The videos are randomly drawn from the sequences in our dataset. Participants were asked to choose the most natural-looking of the three variants for each video. To avoid any bias, we rendered all surfaces in the same color and arranged the variants in random orders. To ensure independent documentation, we used `survio.com` for the technical implementation.

Figure 11 summarizes the outcome of the user study as the proportion of votes each variant received. Our approach achieved the most votes for all videos by a considerable margin.

5 Limitations

The most significant limitations of our approach result from missing details in the foundational physics-based simulation `phy` as demonstrated in Figure 12. For instance, tracking errors can cause hands to move too deep into the head such that the skull is penetrated. In this case, we consider it more natural to not fully resolve collisions rather than bend bones (Figure 12a). We also do not resolve self-collisions between lips or lips and teeth (Figure 12b). Finally, in our anatomical head model, cartilage components are not sufficiently taken into account, causing the nose or ears to bend a bit too much when the hands push firmly (Figure 12c).

Regarding the efficient approximation of `phy` by the neural network `net`, one can consider a lack of generalization over an extensive set of head shapes as a limitation. However, in contrast to previous work [SGPT23, WDX*24] our focus is on personalized head avatars that exhibit a much higher level of detail and authenticity than commonly used head models [LBB*17, FFBB21]. Moreover, our experiments with multiple head shapes (Section 4.3) indicate generalization capacities of `net`, and the short training time of our approach should be sufficient in most scenarios to train `net` to a given personalized head avatar.



(a) *(b)* *(c)*

Figure 12: *(a) displays remaining collisions due to rigid bones, (b) self-intersections of lips, and (c) a too bendy nose due to missing cartilage.*

Finally, a greater diversity in our dataset would be desirable. Although we cover a wide range of head shapes with different anatomical compositions, a more diverse coverage of genders and ethnicities would strengthen our results.

6 Conclusion

In this work, we presented NePHIM, a neural physics-based head-hand interaction model. NePHIM extends previous interaction simulations [SGPT23, WDX*24] with various features such as time-dependent collision paths, pulling of skin, and a higher anatomical precision. Comprehensive experiments and a user study show that our approach is perceived as being considerably closer to reality than the previous state-of-the-art [SGPT23]. Furthermore, we successfully learned a neural approximator of the simulation that allows for rapid inference even on consumer-grade devices.

Nevertheless, we also discussed limitations that provide various starting points for future work. For instance, more detailed anatomical structures and physical properties may enhance the simulation. Moreover, learning the deformation of interactions directly from multi-view videos can contribute to further improvements.

Acknowledgments

This research was supported by the German Federal Ministry of Education and Research (BMBF) through the project HiAvA (ID 16SV8785). Open Access funding enabled and organized by Projekt DEAL.

References

[ABG*18] ACHENBACH J., BRYLKA R., GIETZEN T., ZUM HEBEL K., SCHÖMER E., SCHULZE R., BOTSCH M., SCHWANECKE U.: A multilinear model for bidirectional craniofacial reconstruction. In *Proceedings*

© 2025 The Author(s).
Computer Graphics Forum published by Eurographics and John Wiley & Sons Ltd.

of the Eurographics Workshop on Visual Computing for Biology and Medicine (2018), pp. 67–76. 3

[App24] APPLE INC.:, September 2024. https://developer.apple.com/augmented-reality/arkit/. 7

[AXS*22] ATHAR S., XU Z., SUNKAVALLI K., SHECHTMAN E., SHU Z.: RigNeRF: Fully controllable neural 3D portraits. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 20364–20373. 2

[BCGF19] BAO M., CONG M., GRABLI S., FEDKIW R.: High-quality face capture using anatomical muscles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 10802–10811. 2

[BK05] BOTSCH M., KOBBELT L.: Real-time shape editing using radial basis functions. *Computer Graphics Forum 24*, 3 (2005). 3

[BML*14] BOUAZIZ S., MARTIN S., LIU T., KAVAN L., PAULY M.: Projective dynamics: fusing constraint projections for fast simulation. *ACM Transactions on Graphics (ToG) 33*, 4 (2014), 1–11. 4

[BT17] BULAT A., TZIMIROPOULOS G.: How far are we from solving the 2D & 3D face alignment problem?(and a dataset of 230,000 3D facial landmarks). In *Proceedings of the IEEE International Conference on Computer Vision* (2017), pp. 1021–1030. 7, 10, 14

[CEM*22] CHOI B., EOM H., MOUSCADET B., CULLINGFORD S., MA K., GASSEL S., KIM S., MOFFAT A., MAIER M., REVELANT M., ET AL.: Animatomy: an Animator-centric, Anatomically Inspired System for 3D Facial Modeling, Animation and Transfer. In *SIGGRAPH Asia Conference Papers* (2022), pp. 1–9. 2

[CF19] CONG M., FEDKIW R.: Muscle-based facial retargeting with anatomical constraints. In *ACM SIGGRAPH 2019 Talks* (New York, NY, USA, 2019), SIGGRAPH '19, Association for Computing Machinery. 2

[CO18] CASAS D., OTADUY M. A.: Learning nonlinear soft-tissue dynamics for interactive avatars. *Proceedings of the ACM on Computer Graphics and Interactive Techniques 1*, 1 (2018), 1–15. 3

[Con16] CONG M. D.: *Art-directed muscle simulation for high-end facial animation*. PhD thesis, Stanford University, 2016. 2

[CZ24] CHANDRAN P., ZOSS G.: Anatomically constrained implicit face models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2024), pp. 2220–2229. 2

[DWM*21] DU T., WU K., MA P., WAH S., SPIELBERG A., RUS D., MATUSIK W.: DiffPD: differentiable projective dynamics. *ACM Transactions on Graphics (ToG) 41*, 2 (2021), 1–21. 2

[FFBB21] FENG Y., FENG H., BLACK M. J., BOLKART T.: Learning an animatable detailed 3D face model from in-the-wild images. *ACM Transactions on Graphics (ToG) 40*, 4 (2021), 1–13. 11

[GKE*22] GARBIN S. J., KOWALSKI M., ESTELLERS V., SZYMANOWICZ S., REZAEIFAR S., SHEN J., JOHNSON M., VALENTIN J.: VolTeMorph: realtime, controllable and generalisable animation of volumetric representations. *arXiv preprint arXiv:2208.00949* (2022). 2

[Han15] HANG S.: Tetgen, a delaunay-based quality tetrahedral mesh generator. *ACM Trans. Math. Softw 41*, 2 (2015), 11. 3

[HDDN19] HOLDEN D., DUONG B. C., DATTA S., NOWROUZEZAHRAI D.: Subspace neural physics: fast data-driven interactive simulation. In *Proceedings of the 18th annual ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2019), pp. 1–12. 3, 7

[IKKP17] ICHIM A.-E., KADLEČEK P., KAVAN L., PAULY M.: Phace: Physics-based face modeling and animation. *ACM Transactions on Graphics (ToG) 36*, 4 (2017), 1–14. 2, 4

[IKNDP16] ICHIM A. E., KAVAN L., NIMIER-DAVID M., PAULY M.: Building and animating user-specific volumetric face rigs. In *Symposium on Computer Animation* (2016), pp. 107–117. 2

[KGM15] KWOK Y. L. A., GRALTON J., MCLAWS M.-L.: Face touching: a frequent habit that has implications for hand hygiene. *American journal of infection control 43*, 2 (2015), 112–114. 1

[KK19] KADLEČEK P., KAVAN L.: Building accurate physics-based face models from data. *Proceedings of the ACM on Computer Graphics and Interactive Techniques 2*, 2 (2019), 1–16. 2

[LAR*14] LEWIS J. P., ANJYO K., RHEE T., ZHANG M., PIGHIN F. H., DENG Z.: Practice and theory of blendshape facial models. *Eurographics (State of the Art Reports) 1*, 8 (2014), 2. 2

[LBB*17] LI T., BOLKART T., BLACK M. J., LI H., ROMERO J.: Learning a model of facial shape and expression from 4D scans. *ACM Transactions on Graphics (ToG) 36*, 6 (2017). 2, 8, 11

[LFS*20] LI M., FERGUSON Z., SCHNEIDER T., LANGLOIS T. R., ZORIN D., PANOZZO D., JIANG C., KAUFMAN D. M.: Incremental potential contact: intersection-and inversion-free, large-deformation dynamics. *ACM Trans. Graph. 39*, 4 (2020), 49. 4

[MHHR07] MÜLLER M., HEIDELBERGER B., HENNIX M., RATCLIFF J.: Position based dynamics. *Journal of Visual Communication and Image Representation 18*, 2 (2007), 109–118. 3

[MMG19] MUELLER S. M., MARTIN S., GRUNWALD M.: Self-touch: contact durations and point of touch of spontaneous facial self-touches differ depending on cognitive and emotional load. *PloS one 14*, 3 (2019), e0213677. 1

[MWSZ24] MA S., WENG Y., SHAO T., ZHOU K.: 3d gaussian blendshapes for head avatar animation. In *ACM SIGGRAPH 2024 Conference Papers* (2024), SIGGRAPH '24, Association for Computing Machinery. 2

[PGM*19] PASZKE A., GROSS S., MASSA F., LERER A., BRADBURY J., CHANAN G., KILLEEN T., LIN Z., GIMELSHEIN N., ANTIGA L., DESMAISON A., KOPF A., YANG E., DEVITO Z., RAISON M., TEJANI A., CHILAMKURTHY S., STEINER B., FANG L., BAI J., CHINTALA S.: Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. 8

[PVG*11] PEDREGOSA F., VAROQUAUX G., GRAMFORT A., MICHEL V., THIRION B., GRISEL O., BLONDEL M., PRETTENHOFER P., WEISS R., DUBOURG V., VANDERPLAS J., PASSOS A., COURNAPEAU D., BRUCHER M., PERROT M., DUCHESNAY E.: Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830. 8

[QKS*24] QIAN S., KIRSCHSTEIN T., SCHONEVELD L., DAVOLI D., GIEBENHAIN S., NIESSNER M.: Gaussian avatars: photorealistic head avatars with rigged 3D gaussians. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2024), pp. 20299–20309. 2

[RCCO22] ROMERO C., CASAS D., CHIARAMONTE M. M., OTADUY M. A.: Contact-centric deformation learning. *ACM Transactions on Graphics (ToG) 41*, 4 (2022), 1–11. 3

[RCPO21] ROMERO C., CASAS D., PÉREZ J., OTADUY M.: Learning contact corrections for handle-based subspace dynamics. *ACM Transactions on Graphics (ToG) 40*, 4 (2021), 1–12. 3

[RMF20] RAHMAN J., MUMIN J., FAKHRUDDIN B.: How frequently do we touch facial t-zone: a systematic review. *Annals of Global Health 86*, 1 (2020). 1

[RTB17] ROMERO J., TZIONAS D., BLACK M. J.: Embodied hands: Modeling and capturing hands and bodies together. *ACM Transactions on Graphics (ToG) 36*, 6 (2017). 2, 7

[SGOC20] SANTESTEBAN I., GARCES E., OTADUY M. A., CASAS D.: SoftSMPL: data-driven modeling of nonlinear soft-tissue dynamics for parametric humans. *Computer Graphics Forum 39*, 2 (2020). 3

[SGPT23] SHIMADA S., GOLYANIK V., PÉREZ P., THEOBALT C.: Decaf: monocular deformation capture for face and hand interactions. *ACM Transactions on Graphics (ToG) 42*, 6 (2023), 1–16. 1, 2, 3, 4, 8, 9, 10, 11, 14

[SNF05] SIFAKIS E., NEVEROV I., FEDKIW R.: Automatic determination of facial muscle activations from sparse motion capture marker data. *ACM Trans. Graph. 24*, 3 (July 2005), 417425. 2

© 2025 The Author(s).
Computer Graphics Forum published by Eurographics and John Wiley & Sons Ltd.

[SP04] SUMNER R. W., POPOVIĆ J.: Deformation transfer for triangle meshes. *ACM Transactions on Graphics (ToG) 23*, 3 (2004). 7

[SWR*21] SRINIVASAN S. G., WANG Q., ROJAS J., KLÁR G., KAVAN L., SIFAKIS E.: Learning active quasistatic physics-based models from data. *ACM Transactions on Graphics (ToG) 40*, 4 (2021), 1–14. 3

[WAB*20] WENNINGER S., ACHENBACH J., BARTL A., LATOSCHIK M. E., BOTSCH M.: Realistic virtual humans from smartphone videos. In *Proceedings of the 26th ACM Symposium on Virtual Reality Software and Technology* (2020), pp. 1–11. 7

[WBS23] WAGNER N., BOTSCH M., SCHWANECKE U.: Softdeca: Computationally efficient physics-based facial animations. In *Proceedings of the 16th ACM SIGGRAPH Conference on Motion, Interaction and Games* (2023), pp. 1–11. 2, 3

[WDX*24] WU Q., DOU Z., XU S., SHIMADA S., WANG C., YU Z., LIU Y., LIN C., CAO Z., KOMURA T., ET AL.: Dice: end-to-end deformation capture of hand-face interactions from a single image. *arXiv preprint arXiv:2406.17988* (2024). 2, 3, 4, 10, 11

[WSB24] WAGNER N., SCHWANECKE U., BOTSCH M.: Anacondar: Anatomically-constrained data-adaptive facial retargeting. *Computers & Graphics 122* (2024), 103988. 7

[Xim24] XIMEA:, September 2024. https:www.ximea.com. 7

[YKZ*22] YANG L., KIM B., ZOSS G., GÖZCÜ B., GROSS M., SOLENTHALER B.: Implicit neural representation for physics-driven actuated soft bodies. *ACM Transactions on Graphics (ToG) 41*, 4 (2022), 1–10. 2, 4

[YZC*23] YANG L., ZOSS G., CHANDRAN P., GOTARDO P., GROSS M., SOLENTHALER B., SIFAKIS E., BRADLEY D.: An implicit physical face model driven by expression and style. In *SIGGRAPH Asia 2023 Conference Papers* (2023), Association for Computing Machinery. 2

[ZBT23] ZIELONKA W., BOLKART T., THIES J.: Instant volumetric head avatars. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2023), pp. 4574–4584. 2

[ZBV*20] ZHANG F., BAZAREVSKY V., VAKUNOV A., TKACHENKA A., SUNG G., CHANG C.-L., GRUNDMANN M.: Mediapipe hands: on-device real-time hand tracking. *arXiv preprint arXiv:2006.10214* (2020). 7, 10, 14

Appendix

A Template Dimensions

Mesh	H	J	C	$\mathbb{S}$	$\mathbb{J}$	$\mathbb{C}$
# Vertices	6688	886	4220	11001	899	3354
# Faces / Tets	13372	1768	8444	31456	4190	15634

Table 6: *The dimensions of all template components in our experiments.*

B Weights & Parameters

w_{tar}	$w_{\mathbb{S}}$	$w_{\mathbb{J}}$	$w_{\mathbb{C}}$	w_{push}	w_{pull}	w_{corr}
10^2	10^1	10^4	10^4	10^2	10^2	10^2

Table 7: *The weights of the physics-based simulations of* `phy`.

Proj. Dyn. Iterations	α	l_{min}	r	s	$\Delta\epsilon$
10	0.01	2.5 cm	0.5 cm	50 ms	0.05

Table 8: *The parameters of the physics-based simulations of* `phy`.

C Ridge Calculation

Algorithm 4 Cylinder Ridge

c Cylinder
H Head Surface
I Indices of vertices that are in c
v_i^H Vertex i of H
$\texttt{len}, \texttt{start}, \texttt{end}$ Length, start, end of a cylinder
$\texttt{plane}(\mathsf{r}, \mathsf{n})$ Plane in normal form
$\texttt{mean}$ Mean of vertices
$\texttt{proj}(\mathsf{v}, \mathsf{p})$ Project vertex v on plane p

Function $\texttt{ridge}(I, H, \mathsf{c})$

// Initialize ridge targets
$C_{\text{ridge}} = \{\}$
// Calculate mean of cylinder
$\mathsf{r} = (\texttt{start}(\mathsf{c}) + \texttt{end}(\mathsf{c}))/2$
// Calculate normal of cylinder plane
$\mathsf{n} = \mathsf{r} - \texttt{mean}(H)$
$\mathsf{n} \mathrel{/}= \|\mathsf{n}\|$
// Calculate cylinder plane
$\mathsf{p} = \texttt{plane}(\mathsf{r}, \mathsf{n})$
// Calculate targets
for $i \in I$ **do**
// Calculate plane position
$\mathsf{v}_i^{\mathsf{p}} = \texttt{proj}(\mathsf{v}_i^H, \mathsf{p})$
// Calculate a height offset factor
$\kappa = \min\left(\left\|\texttt{start}(c) - \mathsf{v}_i^{\mathsf{p}}\right\|, \left\|\texttt{end}(c) - \mathsf{v}_i^{\mathsf{p}}\right\|\right) / (\texttt{len}(c)/2)$
// Add ridge target
Add $\left(\mathsf{v}_i^p + \kappa \cdot \texttt{len}(c) \cdot \mathsf{n}, i\right)$ to C_{ridge}
// Return the ridge targets
return C_{ridge}

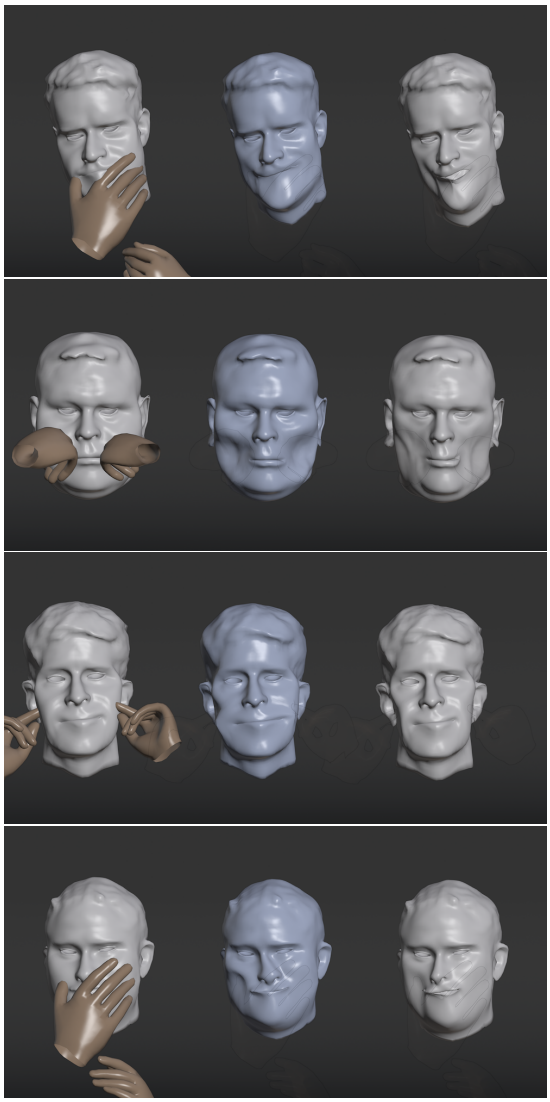
© 2025 The Author(s).
Computer Graphics Forum published by Eurographics and John Wiley & Sons Ltd.

D Dataset

Movement	Single Finger		Multiple Fingers		Open Palm		Closed Fist	
	I	*II*	*I*	*II*	*I*	*II*	*I*	*II*
Poke / Touch								
Cheeks	-	4	2	3	-	4	-	-
Nose	2	-	-	-	-	-	-	-
Forehead	-	-	1	-	-	-	-	-
Chin	-	-	-	-	-	-	4	-
Pinch / Squeeze								
Lips	-	-	5	-	-	-	-	-
Cheeks	-	-	13	-	-	2	-	3
Rub / Stroke								
B → F	-	2	6	1	1	12	-	1
F → B	-	-	-	2	-	5	-	2
D → U	-	-	-	-	4	7	-	2
U → D	-	2	3	3	2	5	-	1
L → R	-	-	-	-	-	1	-	-
R → L	-	-	-	-	-	1	-	-
Circle	-	2	-	7	-	7	-	-
Punch								
Cheeks	-	-	-	-	6	-	4	3
Pull / Tug								
Lips	1	3	-	-	-	-	-	-
Cheeks	-	7	-	8	-	-	-	-
Nose	2	-	-	-	-	-	-	-
Sum	5	20	33	24	13	44	8	12

Table 9: *Quantitative description of the captured hand-head interactions. The number of involved hands is indicated by I and II. A direction is indicated by → where B,F,D,U,L, and R abbreviate back, front, down, up, left, and right, respectively.*

E Additional Simulation Examples



Tracked *Ours* *Decaf [SGPT23]*

Figure 13: *The figure shows examples of our simulation* `phy` *and compares them to the tracked surfaces as well as the simulation of Decaf [SGPT23]. Here, the hands are solely rendered for the tracked meshes to accentuate the simulated deformations.*

F Tracking Network

Dataset	# Identities	Reconstruction	
		Mean ℓ^2	*Max* ℓ^2
Test	One	0.11 cm	0.29 cm
	Eight	0.14 cm	0.46 cm

Table 10: *Test errors of the neural approximation* `net` *of the simulation* `phy` *paired with Mediapipe [BT17, ZBV*20] tracking as described in Section 4.3.3.*

© 2025 The Author(s).
Computer Graphics Forum published by Eurographics and John Wiley & Sons Ltd.