Improving Facial Rig Semantics for Tracking and Retargeting

DALTON OMENS, Stanford University, USA and Epic Games, USA
ALLISE THURMAN, Stanford University, USA
JIHUN YU, Epic Games, USA
RONALD FEDKIW, Stanford University, USA and Epic Games, USA

In this paper, we consider retargeting a tracked facial performance to either another person or to a virtual character in a game or virtual reality (VR) environment. We remove the difficulties associated with identifying and retargeting the semantics of one rig framework to another by utilizing the same framework (3DMM, FLAME, MetaHuman, etc.) for both subjects. Although this does not constrain the choice of framework when retargeting from one person to another, it does force the tracker to use the game/VR character rig when retargeting to a game/VR character. We utilize volumetric morphing in order to fit facial rigs to both performers and targets; in addition, a carefully chosen set of Simon-Says expressions is used to calibrate each rig to the motion signatures of the relevant performer or target. Although a uniform set of Simon-Says expressions can likely be used for all person to person retargeting, we argue that person to game/VR character retargeting benefits from Simon-Says expressions that capture the distinct motion signature of the game/VR character rig. The Simon-Says calibrated rigs tend to produce the desired expressions when exercising animation controls (as expected). Unfortunately, these well-calibrated rigs still lead to undesirable controls when tracking a performance (a well-behaved function can have an arbitrarily ill-conditioned inverse), even though they typically produce acceptable geometry reconstructions. Thus, we propose a fine-tuning approach that modifies the rig used by the tracker in order to promote the output of more semantically meaningful animation controls, facilitating high efficacy retargeting. In order to better address real-world scenarios, the fine-tuning relies on implicit differentiation so that the tracker can be treated as a (potentially non-differentiable) black box.

1 INTRODUCTION

Faces are ubiquitous in digital media such as games and film. Performance-driven facial animation (as opposed to manually keyframed animation) is often used to improve quality, to reduce manual labor, and to facilitate real-time applications. The impact of performance-driven capture is significantly bolstered by the ability to transfer, or *retarget*, captured performances to other human or character models. Although facial motion retargeting has traditionally been limited to big-budget feature films and AAA games due to the need for artist-driven cleanup and intervention, more robust (and democratized) methods would have an enormous impact on a large variety of real-time applications especially given the surging interest in the so-called metaverse.

The computer vision community has made leaps and bounds in the reconstruction of faces using a variety of techniques, and the higher fidelity approaches tend to utilize parameterized models (such as 3DMM [Blanz and Vetter 1999], FLAME [Li et al. 2017], etc.) in order to regularize the reconstruction. Although it is not entirely clear whether or not a computer graphics style semantically-meaningful animation rig (such as is available in Maya [Autodesk, Inc. 2025], in Blender's Rigify [Blender Online Community 2025], or in Unreal Engine's MetaHuman plugins [Epic Games 2025], etc.) could achieve better geometry reconstructions than a PCA-based rig (such as 3DMM, FLAME, etc.), it is clear that most of the progress has occurred with these latter non-semantic rigs that are prevalent in the computer vision community. If the goal were merely to reconstruct a real-world performance in a virtual world, perhaps with different lighting and textures as well as novel views, then non-semantic PCA-style geometry reconstruction is likely sufficient; however, retargeting of that performance even if only to a younger/older version of the performer can be problematic without a semantically-meaningful rig that can disentangle reconstruction geometry from semantic intention.

Whether it be PCA-based or semantic, a rig typically separates the identity parameters N that define the geometry in a neutral expressionless pose from the set of expression parameters θ. Given an image I, a geometry reconstruction $\mathcal{V}$ is obtained by solving an inverse problem to find geometry parameters $\mathcal{S}(I;N,\theta,\psi)$ where ψ represents parameters internal to the solver. Viewing this from the standpoint of an animator, they use animation controls to determine geometry parameters $\mathcal{S}$ based on an (actual or mentally envisioned) image I. They choose the animation controls based on knowledge of how the rig works (even if this is acquired via trial and error), meaning that $\mathcal{S}$ still depends on N and θ; in this scenario, ψ represents parameters internal to the brain of the animator. Regardless of whether $\mathcal{S}$ is determined via a solver or via an animator, $\mathcal{V}(\mathcal{S}(I;N,\theta,\psi);N,\theta)$ represents the reconstructed geometry.

Assuming that the same rig framework is used for both subjects, in order to remove the difficulties associated with identifying and retargeting the semantics of one rig framework to another, retargeting can be expressed as $\mathcal{V}(\mathcal{S}(I;N,\theta,\psi);\hat{N},\hat{\theta})$ where $\hat{N}$ and $\hat{\theta}$ are the identity and expression parameters of the target. This expression highlights the main issue with retargeting, i.e. that $\mathcal{S}$ is unaware of $\hat{N}$ and $\hat{\theta}$. To elucidate this, let $\mathcal{V}_o = \mathcal{V}(\mathcal{S}(I;N,\theta,\psi);N,\theta)$ be geometry reconstructed from an input image I of a performer, and let $\hat{\mathcal{V}}_o$ be the preferred geometry on the target; then, $\hat{\mathcal{V}}_o = \mathcal{V}(\mathcal{S}(\hat{I};\hat{N},\hat{\theta},\psi);\hat{N},\hat{\theta})$ where $\hat{I}$ is an image of $\hat{\mathcal{V}}_o$. Unfortunately, $\mathcal{V}(\mathcal{S}(I;N,\theta,\psi);\hat{N},\hat{\theta})$ will typically differ significantly from $\hat{\mathcal{V}}_o$ due to the disparity between $\mathcal{S}(I;N,\theta,\psi)$ and $\mathcal{S}(\hat{I};\hat{N},\hat{\theta},\psi)$.

An expert animator working with a high quality rig would determine $\mathcal{S}$ by choosing animation controls while thinking about N and θ abstractly in a manner that includes a dependence on $\hat{N}$, $\hat{\theta}$, and $\hat{I}$. Some high-end trackers are designed to mimic this approach in various ways; for example, certain degrees of freedom may be preferred or penalized in the objective function. Some trackers aim

Authors' addresses: Dalton Omens, domens@stanford.edu, Stanford University, Stanford, California, USA and Epic Games, Cary, North Carolina, USA; Allise Thurman, allise@stanford.edu, Stanford University, Stanford, California, USA; Jihun Yu, jihun.yu@epicgames.com, Epic Games, Cary, North Carolina, USA; Ronald Fedkiw, rfedkiw@stanford.edu, Stanford University, Stanford, California, USA and Epic Games, Cary, North Carolina, USA.

for an optimal geometry reconstruction in the first pass while subsequently reworking the degrees of freedom with preferences and penalties in a second pass. In fact, it is sometimes useful to allow for a varying identity in the first pass (as this can be used to explain errors in camera extrinsics, etc.), while rigidifying the identity in the second pass; in contrast, rigidifying the identity in the first pass could cause erroneous values for expression parameters as they aim to explain various errors.

In spite of the various strategies used to improve the output of high-end trackers, expert animators still spend a great deal of time cleaning the output (i.e. modifying $\mathcal{S}$ via animator controls) before it can be used for retargeting. Thus, in this paper, we aim to improve high-end trackers even further by eliminating the disparity between $\mathcal{S}(I; N, \theta, \psi)$ and $\mathcal{S}(\hat{I}; \hat{N}, \hat{\theta}, \psi)$. The maturity of reconstruction techniques means that N and $\hat{N}$ are well-determined. In addition, a high-end tracker likely has a well-engineered solver with optimal parameters ψ. Thus, we focus on the expression parameters stressing that motion signatures can vary greatly from person to person. That is, our main idea is to modify the original θ to a new $\theta_{\mathcal{S}}$ in order to obtain $\mathcal{S}(I; N, \theta_{\mathcal{S}}, \psi) \approx \mathcal{S}(\hat{I}; \hat{N}, \hat{\theta}, \psi)$. In order to cover the full range of expression parameters, a large number of image pairs I and $\hat{I}$ would be required. Unfortunately, a different $\theta_{\mathcal{S}}$ needs to be determined for every performer/target pair. Once again, we rely on animator mimicry in order to address this problem. The main idea is to use agreement in animator controls as a proxy for agreement in geometry degrees of freedom. The prior implies the latter but not necessarily vice-versa, so this is typically a stricter condition.

In summary, we address the scenario where real-world performance is used to drive a virtual model of another person or game/VR character. In the first case, the same rig framework is volumetrically morphed to 3D reconstructions of both the performer and the target; in the second case, the carefully crafted game/VR character rig is volumetrically morphed to a 3D reconstruction of the performer. A carefully chosen set of Simon-Says expressions (see [Zhu et al. 2024]) is used to calibrate each rig to the motion signatures of the relevant performer or target; for performer to game/VR character retargeting, the Simon-Says expressions should capture the motion signatures of the game/VR character rig. Unfortunately, these well-calibrated rigs still lead to undesirable controls when tracking a performance. Thus, we propose a fine-tuning approach that modifies the expression parameters used by the tracker in order to promote the output of more semantically meaningful animation controls, facilitating high efficacy retargeting. The fine-tuning relies on implicit differentiation so that a high-end potentially non-differentiable black box tracker can be utilized.

We summarize our novel contributions as follows:

- We provide a mathematical analysis of rigs and trackers via linearization in order to rigorously demonstrate that a tracker can be thought of as a geometry reconstruction followed by the inverse of a rig operation.
- In the context of Simon-Says, we choose a set of expressions that covers the most important rig parameters while still remaining minimal with expressions that are straightforward for an average person to make. In addition, we ensure that the numerical optimization respects the various constraints expert riggers utilize in the rig design process.
- Our implicit function theorem style approach allows for modification of the tracker parameters via an implicit approximation of the derivative of the tracker output with respect to those parameters without requiring access to or knowledge of the internal workings of the tracker (the tracker can be treated as a black box) and without requiring differentiability.
- We present a reformulation of Broyden's method illustrating its connection to finite difference estimates and subsequently show how to substitute in more robust finite differences that minimize truncation error while avoiding roundoff errors. These observations allow us to improve the search direction.
- We illustrate that our approach is both feasible on and has high efficacy on real-world problems by utilizing best practices at each stage of the process: for example, we propose techniques aimed at obtaining accurate reconstructed geometry, we use volumetric morphing to obtain tight fitting high-end rigs that produce high fidelity motion, we worked with animators and riggers to get domain-expert feedback, etc.

2 RELATED WORK

There is a plethora of work aimed at representing faces in digital worlds. Although the seminal efforts relied quite heavily on the ability of artists to sculpt, paint, and animate both virtual characters and digital doubles, the impact of data-driven approaches has increased significantly over the years. A number of works mostly aim primarily to *disentangle the camera view* from a virtual model of the subject of interest, rasterizing photorealistic images from novel views. The most notable, e.g. [Buehler et al. 2024, 2023; Cao et al. 2022; Duan et al. 2023; Gafni et al. 2021; Grassal et al. 2021; Kirschstein et al. 2023; Lombardi et al. 2018; Sarkar et al. 2023; Teotia et al. 2024; Trevithick et al. 2023; Wu et al. 2023; Xiang et al. 2024; Xu et al. 2024; Yang et al. 2024; Zheng et al. 2022, 2023; Zielonka et al. 2022a], utilize either NeRFs [Mildenhall et al. 2021] or Gaussian Splats [Kerbl et al. 2023]. See also [Ma et al. 2021].

Going one step further, a plethora of work has been aimed at geometry reconstruction, which additionally (i.e. in addition to disentangling the camera viewpoint) *disentangles three dimensional geometry from texture and lighting*. Many such works utilize a PCA representation, typically either a 3DMM [Blanz and Vetter 1999; Egger et al. 2020] or FLAME [Li et al. 2017], for regularization. See e.g. [Danecek et al. 2022; Hewitt et al. 2024; Laine et al. 2017; Li et al. 2009; Retsinas et al. 2024; Taubner et al. 2024; Wang et al. 2024; Wood et al. 2022; Zhang et al. 2023; Zielonka et al. 2022b]. Other works use anatomical constraints [Beeler and Bradley 2014; Wu et al. 2016] or sculpted blend shapes [Garrido et al. 2013] for regularization. Similar in spirit to our approach, [Baert et al. 2024] points out that geometric reconstruction results can be improved by personalizing the FLAME expression coefficients.

Independent of geometry, various works aim to additionally *disentangle texture from lighting*, see e.g. [Athar et al. 2024; Bharadwaj et al. 2023; Deng et al. 2019; Dib et al. 2023, 2021a, 2024, 2021b; Han

et al. 2024; Lei et al. 2023; Saito et al. 2024; Tewari et al. 2017; Zhou et al. 2024]. The high-end approaches utilize a light stage [Debevec et al. 2000; Ghosh et al. 2011; Saito et al. 2017].

When retargeting, it is important to *disentangle semantic intention from geometry*. Although there is a plethora of computer vision work interested in semantic intention for the sake of scene interpretation, these works are rarely interested in the 3D reconstruction of facial geometry. Those who are focused on both typically work in the field of computer graphics and make use of so-called facial animation rigs, see e.g. [Orvalho et al. 2012]. Notable work on facial animation rigs includes [Bailey et al. 2020; Bao et al. 2019; Choi et al. 2022; Cong et al. 2015; Debevec et al. 2000; Kavan et al. 2024; Li et al. 2010; Ming et al. 2024; Yang et al. 2023a,b; Zhang et al. 2004]. Publicly available options for facial animation software include MetaHumans in the Unreal Engine [Epic Games 2025] as well as various tools suites in Maya [Autodesk, Inc. 2025] and Blender [Blender Online Community 2025].

For a review of prior work on the *retargeting* of facial performances to other persons or characters, see [Zhu and Joslin 2024b]. Although some works have found a degree of success by transferring *per-vertex* geometry displacements from one mesh to another, see e.g. [Cha et al. 2025; Choi et al. 2025; Noh and Neumann 2001; Qin et al. 2023; Seol et al. 2012; Sumner and Popović 2004], most focus on the underlying rig degrees of freedom. Although some works have aimed to find reasonable mappings between *disparate rig frameworks*, see e.g. [Deng et al. 2006; Song et al. 2011; Wang et al. 2004], this is still a difficult problem. Various authors have aimed to address this difficult problem via neural network techniques, see e.g. [Aneja et al. 2018; Kim et al. 2021; Zhang et al. 2022], while others have relied on constrained numerical optimization, e.g. [Chandran et al. 2022] uses anatomical constraints.

Some approaches have aimed to skip the retargeting of geometry and/or rig controls entirely by instead using *image-based retargeting* followed by reconstruction or rig inversion from the retargeted image. [Moser et al. 2021] uses a network to translate an image of the performer to a semantically similar image of the target and regress animation controls from the new image. [Qiu et al. 2024] encodes an image of the performer into a latent space and subsequently uses a target-specific decoder to obtain animation controls. With the recent surge in popularity of LLMs, there is also some interest in audio-based retargeting followed by reconstruction or rig inversion from the retargeted audio. In this vein, [Pan et al. 2025] recently proposed a method for regressing MetaHuman animator controls for an audio signal.

High-end approaches tend to use the *same rig frameworks* on both the performer and target, aiming for a one-to-one remapping of rig control parameters. In order to obtain high efficacy results, the semantic intention of the performance needs to be disentangled from the geometry. [Tran et al. 2024] and [Bai et al. 2024] focus on achieving realistic person-to-person retargeting within the constrained input domain of VR headset cameras. [Racković et al. 2024] proposes a new optimization framework aiming to obtain semantically-meaningful animation controls that retarget well. Similar in spirit to our approach, [Ribera et al. 2017] uses a Range of Motion (ROM) video to modify expression blendshapes aiming to obtain semantically-meaningful animation controls. [Zhu and Joslin 2024a] proposes a new FACS-based rig designed to improve the retargeting efficacy of regressed animation controls.

3 PRELIMINARIES

Similar to $\mathcal{V}$ (from Section 1), animation rigs $\mathcal{R}(c; N, \theta_{\mathcal{R}})$ output geometry. The main difference is that the geometry degrees of freedom in a rig are specified directly (instead of indirectly, i.e. Section 1 assumes that $\mathcal{S}$ is derived from c) by the animation controls c. Importantly, the animation controls are typically designed to have interpretable semantic meaning, whereas PCA coefficients simply represent geometric modes. Given a 3D geometric reconstruction of the neutral pose N and a set of corresponding animation controls versus geometry pairs (c_k, v_k),

$$\min_{\theta_{\mathcal{R}}} \sum_k ||\mathcal{R}(c_k; N, \theta_{\mathcal{R}}) - v_k||_2^2 \tag{1}$$

can be used to determine the internal parameters $\theta_{\mathcal{R}}$ of the rig. Each of the v_k can either be sculpted by an artist or reconstructed from an image I_k chosen to correspond with c_k. Various methods (e.g. volumetric morphing [Zhu et al. 2024]) can be used to retarget a rig to a new character, allowing an artist to co-animate both characters with the same artist controls. Unfortunately, the expressions on the new character will closely resemble those of the old character. This can be remedied by laboriously sculpting geometry v_k for a number of key poses (each specified by a c_k) on the new character; then, Equation 1 can be solved to determine new internal parameters for the new character's rig. When retargeting to another person, sculpting can be avoided by using a Simon-Says approach (see Section 8) to create semantically correlated images I_k that can be reconstructed into geometry v_k via inverse rendering (see Section 9) and other techniques.

Similar to $\mathcal{S}$ (from Section 1), a tracker $\mathcal{T}(I; \psi)$ solves an inverse problem to find animation controls c that produce geometry when evaluated in the rig $\mathcal{R}(c; N, \theta_{\mathcal{R}})$. Generally speaking, a tracker does not necessarily need to depend on a rig; thus, N and $\theta_{\mathcal{R}}$ are temporarily (for the sake of this section) absorbed into ψ if the tracker depends on them. Given a set of corresponding animation controls versus image pairs (c_k, I_k),

$$\min_{\psi} \sum_k ||\mathcal{T}(I_k; \psi) - c_k||_2^2 \tag{2}$$

can be used to determine the internal tracker parameters. In order to improve the tracker's ability to robustly generalize to unseen images, regularization is typically required. This is sometimes accomplished by considering the geometry $\mathcal{R}(\mathcal{T}(I; \psi); N, \theta_{\mathcal{R}}) + \tilde{v}_{\mathcal{T}}(I; \psi)$ obtained by evaluating the tracker output in the corresponding rig. Here, $\tilde{v}_{\mathcal{T}}$ is a correction to the geometry, often included when a rig is thought to have limited expressivity. This motivates a modification of Equation 2 to

$$\begin{aligned} \min_{\psi} \sum_k & \gamma_1 ||\mathcal{T}(I_k; \psi) - c_k||_2^2 + \\ & \gamma_2 ||\mathcal{R}(\mathcal{T}(I_k; \psi); N, \theta_{\mathcal{R}}) + \tilde{v}_{\mathcal{T}}(I_k; \psi) - v_k||_2^2 + \\ & \gamma_{\tilde{v}_{\mathcal{T}}} ||\tilde{v}_{\mathcal{T}}(I_k; \psi)||_2^2 \end{aligned} \tag{3}$$

in order to match both the artist controls and the geometry. The last term in Equation 3 regularizes $\tilde{v}_{\mathcal{T}}$ to be small, so that the geometry v_k

is mostly matched by the rig. With regard to terminology, a would-be tracker that outputs good geometry but poor artist controls (focusing on the second term in Equation 3 at the expense of the first) is typically thought of as a reconstruction method, not an expression tracker.

Note that the first term in Equation 3 expresses the "agreement in animator controls" mentioned in Section 1 as a proxy for agreement in geometry degrees of freedom. That is, changing the minimization in Equation 3 to be over "$\theta_{\mathcal{S}}$" instead of ψ would essentially achieve the desired result. Importantly, this only has to be done once per performer/rig combination independent of the number of target characters.

4 A MOTIVATIONAL LINEARIZATION

For the sake of simplicity, assume that the rig output v consists of vertex displacements from the neutral pose so that $\mathcal{R}$ does not necessarily depend on N; in addition, assume that the rig is linear in c implying

$$v = A(\theta_{\mathcal{R}})c \tag{4}$$

where $A(\theta_{\mathcal{R}})$ is a size $m \times n$ matrix. Stacking all the (c_k, v_k) pairs into separate matrices C and V yields

$$V = A(\theta_{\mathcal{R}})C \tag{5}$$

$$C^T A^T(\theta_{\mathcal{R}}) = V^T \tag{6}$$

$$A(\theta_{\mathcal{R}}) = VC^T(CC^T)^{-T} = VC^T(CC^T)^{-1} \tag{7}$$

illustrating that A is unique as long as the c_k are chosen to make C^T full rank. If the expression set does not make C^T full rank, then Equation 6 can be modified to be full rank by adding an additional set of equations $\epsilon I_{nxn} A^T(\theta_{\mathcal{R}}) = \epsilon A^T(\theta_0)$ where θ_0 are default values for the rig (perhaps from a template rig). As is typical for Levenberg-Marquardt, ϵ is chosen to limit the interference of the regularizing equations with $C^T A^T = V^T$.

Assume that the tracker executes an error-free geometry reconstruction to obtain geometry v from an image I; in addition, assume that the tracker is linear in the reconstructed geometry v so that

$$c = B(\psi)v \tag{8}$$

where $B(\psi)$ is a size $n \times m$ matrix. B satisfies

$$C = B(\psi)V \tag{9}$$

$$CC^T(CC^T)^{-1} = B(\psi)VC^T(CC^T)^{-1} \tag{10}$$

$$I_{nxn} = B(\psi)A(\theta_{\mathcal{R}}) \tag{11}$$

implying that B is a size $n \times m$ left inverse of A. *This notion of a tracker consisting of a geometry reconstruction (from I to v) followed by the "inverse" of a rig operation (Equation 8) is conceptually useful.* If C^T is not full rank, the additional rows of C^T and V^T that were added to Equation 6 need to be included as additional columns in Equation 9 in order to derive Equation 11. These can be seen as additional (c_k, v_k) pairs.

Now, instead suppose that the tracker erroneously reconstructs geometry $\hat{v}_k$ from the images I_k; then, Equation 9 instead becomes

$$C = \hat{B}(\psi)\hat{V} \tag{12}$$

where $\hat{B}$ rectifies errors in geometry reconstruction. Similar to Equations 5 to 7, Equation 12 leads to

$$\hat{B}(\psi) = C\hat{V}^T(\hat{V}\hat{V}^T)^{-1} \tag{13}$$

when $\hat{V}^T$ is full rank; otherwise, equations of the form $\epsilon I_{mxm}\hat{B}^T(\psi) = \epsilon\hat{B}^T(\psi_0)$ can be included. Note that the regularization of $\hat{V}^T$ adds columns to C and $\hat{V}$ that are entirely different from the columns that were added to C and V in order to regularize C^T in Equation 6. Thus, there are two reasons that $\hat{B}$ in Equation 13 is no longer a left inverse of A in Equation 7. Not only have reconstruction errors caused V to change to $\hat{V}$, but the rig-solve based regularization used to obtain Equation 11 is different from the tracker-solve based regularization used to obtain Equation 13. Since $\hat{B}$ is no longer a left inverse of A, the rig used for the "inverse" operation in the tracker *should* be inconsistent with the animation rig.

Rewriting Equation 12 from the standpoint of a solver that "inverts" the rig, the tracker solves a linear system of the form

$$\hat{A}(\theta_{\mathcal{T}})C = \hat{V} \tag{14}$$

to determine C. This leads to

$$\hat{A}(\theta_{\mathcal{T}}) = \hat{V}C^T(CC^T)^{-1} \neq VC^T(CC^T)^{-1} = A(\theta_{\mathcal{R}}) \tag{15}$$

where the regularization of C and $\hat{V}$ is the same as the regularization of C and V in Equation 6; thus, the only difference between the left hand side and right hand side of Equation 15 is $\hat{V}$ versus V. Here, $\hat{A}$ denotes the perturbed rig used by the tracker; thus, $\hat{A}$ is written to depend on parameters $\theta_{\mathcal{T}} \neq \theta_{\mathcal{R}}$. Starting with Equation 14 instead of Equation 5 leads to an Equation 11 assertion that B should be the left inverse of $\hat{A}$. However, actually finding such a B (equivalent to running a tracker) requires regularization similar to what was used to obtain Equation 13. Thus, $\hat{A}$ should be perturbed even further (beyond Equation 15) in order to account for regularization in the solver.

5 DETERMINING A RIG FOR THE TRACKER'S "SOLVE"

Going forward, we will write $\mathcal{T}(I; \theta_{\mathcal{T}}, \psi)$ to stress that the tracker depends on the rig parameters in some perhaps unknown or complex fashion. We drop the explicit mention of the dependence on the neutral N, since it has no bearing on the discussion. In fact, ψ could be omitted as well, since only $\theta_{\mathcal{T}}$ will be modified; however, ψ is left in as a placeholder for non-modified tracker parameters (technically, N could be absorbed into ψ).

Although the linearization in section 4 provides motivation, the vast majority of state-of-the-art trackers will be nonlinear, complex, and not readily differentiable. We only assume that the tracker has been trained to work via Equation 3, or a similar in spirit approach. Thus, we propose using a modified version of Equation 3, i.e.

$$\begin{aligned}\min_{\theta_{\mathcal{T}}} \sum_k \Big(&\gamma_1 ||\mathcal{T}(I_k; \theta_{\mathcal{T}}, \psi) - c_k||_2^2 + \\ &\gamma_2 ||\mathcal{R}(\mathcal{T}(I_k; \theta_{\mathcal{T}}, \psi); \theta_{\mathcal{R}}) - v_k||_2^2\Big) + \\ &\gamma_\epsilon ||\theta_{\mathcal{T}} - \theta_{\mathcal{R}}||_2^2\end{aligned} \tag{16}$$

in order to fine-tune the tracker by modifying the $\theta_{\mathcal{T}}$ parameters of the underlying animation rig. Note that $\tilde{v}_{\mathcal{T}}$ has been eliminated, so that modifying $\theta_{\mathcal{T}}$ does not encourage the rig to drift away from its ability to match the ground-truth geometry; in addition, this

removes the need to estimate derivatives of $\tilde{v}_{\mathcal{T}}$ with respect to $\theta_{\mathcal{T}}$. Finally, the γ_ϵ term is added.

In our experience, most trackers have been trained to provide good geometric reconstruction. This means that the γ_2 term in Equation 16 will be small when $\theta_{\mathcal{T}} = \theta_{\mathcal{R}}$. However, in our experience, trackers often struggle to output semantically correct rig controls. This means that the γ_1 term in Equation 16 will typically be large when $\theta_{\mathcal{T}} = \theta_{\mathcal{R}}$. Thus, our goal is to minimize the γ_1 term while staying close to the constraint surface that keeps the γ_2 term small. The γ_ϵ term emphasizes that the rig is being fine-tuned, softly constraining the minimization of the γ_1 term on or near the γ_2 constraint surface to occur within a ball about $\theta_{\mathcal{R}}$; otherwise, $\theta_{\mathcal{T}}$ can drift too much (which we have observed in our experiments).

The γ_2 term is important since it emphasizes that the output of the tracker will subsequently be used in the animation rig, which depends on $\theta_{\mathcal{R}}$. The observant reader might wonder whether the rig used internal to the tracker also requires geometry regularization. This can be accomplished by including

$$\gamma_3 ||\mathcal{R}(\mathcal{T}(I_k; \theta_{\mathcal{T}}, \psi); \theta_{\mathcal{T}}) - v_k||_2^2 \tag{17}$$

in Equation 16. Our preliminary tests (see Section 7.2) obviate the need for this term.

6 DIFFERENTIATING THE TRACKER

In some scenarios, one may not have access to the internal workings of the tracker. In other scenarios, the tracker may be open-source, albeit nonlinear, complex, and/or not readily differentiable. In either scenario, $\frac{\partial \mathcal{T}}{\partial \theta}$ cannot be computed explicitly. Thus, in the spirit of Equation 17, we make the reasonable ansatz that the tracker actually uses its internal rig in order to provide some sort of regularization; that is, *we assume that the controls output by the tracker yield credible geometry when evaluated on the rig internal to the tracker.* This can be written as

$$\hat{v}(I; \theta_{\mathcal{T}}, \psi) = v(I) + \tilde{v}(I; \theta_{\mathcal{T}}, \psi) = \mathcal{R}(\mathcal{T}(I; \theta_{\mathcal{T}}, \psi); \theta_{\mathcal{T}}) \tag{18}$$

where v is the ground-truth geometry and $\tilde{v}$ is an erroneous perturbation away from the ground-truth geometry; then, the ansatz that the tracker uses its internal rig to provide reasonable regularization can be restated as a claim that $\tilde{v}$ should be small. Differentiating Equation 18 with respect to θ leads to

$$\left.\frac{\partial \tilde{v}(I;\theta,\psi)}{\partial \theta}\right|_{\theta=\theta_{\mathcal{T}}} = \left.\frac{\partial \mathcal{R}(c;\theta_{\mathcal{T}})}{\partial c}\right|_{c=\mathcal{T}(I;\theta_{\mathcal{T}},\psi)} \left.\frac{\partial \mathcal{T}(I;\theta,\psi)}{\partial \theta}\right|_{\theta=\theta_{\mathcal{T}}} + \left.\frac{\partial \mathcal{R}(\mathcal{T}(I;\theta_{\mathcal{T}},\psi);\theta)}{\partial \theta}\right|_{\theta=\theta_{\mathcal{T}}} \tag{19}$$

and thus

$$\left.\frac{\partial \mathcal{R}(c;\theta_{\mathcal{T}})}{\partial c}\right|_{c=\mathcal{T}(I;\theta_{\mathcal{T}},\psi)} \left.\frac{\partial \mathcal{T}(I;\theta,\psi)}{\partial \theta}\right|_{\theta=\theta_{\mathcal{T}}} = - \left.\frac{\partial \mathcal{R}(\mathcal{T}(I;\theta_{\mathcal{T}},\psi);\theta)}{\partial \theta}\right|_{\theta=\theta_{\mathcal{T}}} + \left.\frac{\partial \tilde{v}(I;\theta,\psi)}{\partial \theta}\right|_{\theta=\theta_{\mathcal{T}}} \tag{20}$$

as an implicit equation for $\frac{\partial \mathcal{T}}{\partial \theta}$. Note that $\frac{\partial v}{\partial \theta} = 0$, since v is the ground-truth geometry. When $\frac{\partial \mathcal{R}}{\partial c}$ is not full rank, equations of the form $\epsilon I_{nxn} \frac{\partial \mathcal{T}}{\partial \theta} = 0$ can be included in order to regularize the gradients of the controls in the unconstrained subspace; alternatively, when tractable, the pseudo-inverse can be used as well.

Although one could assume that the tracker produces a $\hat{v}$ very close to v making $\tilde{v}$ small enough to remove it from Equation 18, subsequently removing its derivative from Equations 19 and 20, it is also possible to estimate $\frac{\partial \tilde{v}}{\partial \theta}$. In order to estimate $\frac{\partial \tilde{v}}{\partial \theta}$, we utilize the finite difference update at the heart of Broyden's method [Broyden 1965], which is also the motivation for many other optimization schemes including SR1 [Nocedal and Wright 2006], DFP [Davidon 1991; Fletcher and Powell 1963], BFGS [Fletcher 1987], and L-BFGS [Liu and Nocedal 1989].

For the sake of exposition, let u be some instance of $\tilde{v}$ for a given I and ψ, so that u only varies with $\theta_{\mathcal{T}}$ during optimization. Given distinct values θ_α and θ_β with $\Delta\theta = \theta_\beta - \theta_\alpha$, an estimate to $\frac{\partial u}{\partial \theta}$ at θ_α can be updated via

$$\left(\left.\frac{\partial u}{\partial \theta}\right|_{\theta=\theta_\alpha}\right)^{new} = \left(\left.\frac{\partial u}{\partial \theta}\right|_{\theta=\theta_\alpha}\right)^{old} + \frac{1}{(\Delta\theta)^T \Delta\theta}\left(u(\theta_\beta) - u(\theta_\alpha) - \left(\left.\frac{\partial u}{\partial \theta}\right|_{\theta=\theta_\alpha}\right)^{old} \Delta\theta\right)(\Delta\theta)^T \tag{21}$$

so that

$$\left(\left.\frac{\partial u}{\partial \theta}\right|_{\theta=\theta_\alpha}\right)^{new} \Delta\theta = u(\theta_\beta) - u(\theta_\alpha) \tag{22}$$

is satisfied. Equation 22 shows that the estimate for the derivative satisfies a secant-type equation in the direction of $\Delta\theta$, as is typical for Broyden-style methods. In fact, letting $s = ||\Delta\theta||_2 = \sqrt{(\Delta\theta)^T \Delta\theta}$ allows Equations 21 and 22 to be written as

$$\left(\left.\frac{\partial u}{\partial \theta}\right|_{\theta=\theta_\alpha}\right)^{new} = \left(\left.\frac{\partial u}{\partial \theta}\right|_{\theta=\theta_\alpha}\right)^{old} + \left(\frac{u(\theta_\alpha + s\widehat{\Delta\theta}) - u(\theta_\alpha)}{s} - \left(\left.\frac{\partial u}{\partial \theta}\right|_{\theta=\theta_\alpha}\right)^{old} \widehat{\Delta\theta}\right)\widehat{\Delta\theta}^T \tag{23}$$

$$\left(\left.\frac{\partial u}{\partial \theta}\right|_{\theta=\theta_\alpha}\right)^{new} \widehat{\Delta\theta} = \frac{u(\theta_\alpha + s\widehat{\Delta\theta}) - u(\theta_\alpha)}{s} \tag{24}$$

where $\widehat{\Delta\theta}$ is a unit vector in the direction of $\Delta\theta$. Equations 23 and 24 illustrate that Equation 21 is updating $\frac{\partial u}{\partial \theta}$ to be consistent with a finite difference estimate of $\frac{\partial u}{\partial \theta}$ in the direction $\widehat{\Delta\theta}$.

Unlike the typical approach which compiles a running estimate of $\frac{\partial u}{\partial \theta}$ as the optimization steps through parameter space, we use Equation 23 (equivalent to Equation 21) in order to construct an estimate of $\frac{\partial u}{\partial \theta}$ at θ_α. This is accomplished by using Equation 23 repeatedly with different $\widehat{\Delta\theta}$ directions. The various $\widehat{\Delta\theta}$ can be chosen randomly, importance sampled, etc., noting that the estimate of $\frac{\partial u}{\partial \theta}$ is merely used to aid in the choice of the search direction; thus, this approach is perhaps best described as a predictor-corrector method similar in spirit to Nesterov or second-order Runge-Kutta. Finally, note that each $\widehat{\Delta\theta}$ direction can use a varying s in order to ascertain a reasonable finite difference approximation that avoids round-off error while minimizing truncation error.

The initial guess for $\frac{\partial u}{\partial \theta}$ in the first iteration of Equation 23 can be set to be the final value obtained for $\frac{\partial u}{\partial \theta}$ when iterating Equation 23 in the prior optimization step. This warm start bears resemblance to Broyden's method, and it greatly accelerated convergence in all of our tests. In the first optimization iteration where there is no prior estimate for $\frac{\partial u}{\partial \theta}$, an initial guess of zero can be used. This differs from Broyden's method, which uses the identity because the estimate is typically used in conjunction with matrix inversion.

Note that u cannot actually be an instance of $\tilde{v}$, since $\tilde{v}$ represents unknown error; however, letting u be an instance of $\hat{v}$, which is equivalent to the rig output (see Equation 18), still leads to an estimate of $\frac{\partial \tilde{v}}{\partial \theta}$ since the exact solution v does not vary with θ. Formally,

$$\begin{aligned}\frac{\partial \mathcal{R}(c;\theta_{\mathcal{T}})}{\partial c}\bigg|_{c=\mathcal{T}(I;\theta_{\mathcal{T}},\psi)} \frac{\partial \mathcal{T}(I;\theta,\psi)}{\partial \theta}\bigg|_{\theta=\theta_{\mathcal{T}}} = \\ -\frac{\partial \mathcal{R}(\mathcal{T}(I;\theta_{\mathcal{T}},\psi);\theta)}{\partial \theta}\bigg|_{\theta=\theta_{\mathcal{T}}} + \frac{\partial \hat{v}(I;\theta,\psi)}{\partial \theta}\bigg|_{\theta=\theta_{\mathcal{T}}}\end{aligned} \tag{25}$$

can be used as a replacement for Equation 20.

Note that it may be more appropriate to use

$$\mathcal{R}(\mathcal{T}(I;\theta_{\mathcal{T}},\psi);\theta_{\mathcal{T}}) + \tilde{v}_{\mathcal{T}}(I;\theta_{\mathcal{T}},\psi) \tag{26}$$

on the far right side of Equation 18 when $\tilde{v}_{\mathcal{T}}$ is used as a geometry correction for rigs with limited expressivity (see the discussion leading up to Equation 3). This leads to

$$\begin{aligned}\hat{v}(I;\theta_{\mathcal{T}},\psi) - \tilde{v}_{\mathcal{T}}(I;\theta_{\mathcal{T}},\psi) &= v(I) + \tilde{v}(I;\theta_{\mathcal{T}},\psi) - \tilde{v}_{\mathcal{T}}(I;\theta_{\mathcal{T}},\psi) \\ &= \mathcal{R}(\mathcal{T}(I;\theta_{\mathcal{T}},\psi);\theta_{\mathcal{T}})\end{aligned} \tag{27}$$

emphasizing that $\tilde{v} - \tilde{v}_{\mathcal{T}}$ plays the same role here as $\tilde{v}$ played in Equations 18 and 20. In some scenarios, $\frac{\partial \tilde{v}_{\mathcal{T}}}{\partial \theta}$ might be known, and $\tilde{v}$ might be negligible when compared to $\tilde{v}_{\mathcal{T}}$. Otherwise, $\frac{\partial}{\partial \theta}(\tilde{v} - \tilde{v}_{\mathcal{T}})$ can be estimated by letting u be an instance of $\tilde{v} - \tilde{v}_{\mathcal{T}}$ in the prior discussion. Of course, similar to the discussion proceeding Equation 25, u would actually need to be an instance of $\hat{v} - \tilde{v}_{\mathcal{T}}$, which is the rig output from Equation 27, in order to accomplish this.

7 SIMPLE LINEAR EXAMPLES

Assuming linearity of the rig according to Equation 4 gives

$$\frac{\partial \mathcal{R}(c,\theta)}{\partial c} = A(\theta) \tag{28}$$

$$\frac{\partial \mathcal{R}(c,\theta)}{\partial \theta} = \begin{bmatrix} c^T & 0_{1x3} & 0_{1x3} \\ 0_{1x3} & c^T & 0_{1x3} \\ 0_{1x3} & 0_{1x3} & c^T \end{bmatrix} \tag{29}$$

where $\frac{\partial \mathcal{R}}{\partial \theta}$ is size 3×9 and θ_ℓ for $\ell = 1 \ldots 9$ are the entries of A enumerated in row-major order. Equation 16, including Equation 17, can be written as

$$\begin{aligned}\min_{\theta_{\mathcal{T}}} \sum_k \Big(&\gamma_1 ||\mathcal{T}(v_k;\theta_{\mathcal{T}}) - c_k||_2^2 + \\ &\gamma_2 ||A(\theta_{\mathcal{R}})\mathcal{T}(v_k;\theta_{\mathcal{T}}) - v_k||_2^2 + \\ &\gamma_3 ||\hat{A}(\theta_{\mathcal{T}})\mathcal{T}(v_k;\theta_{\mathcal{T}}) - v_k||_2^2\Big) + \\ &\gamma_\epsilon ||\theta_{\mathcal{T}} - \theta_{\mathcal{R}}||_2^2\end{aligned} \tag{30}$$

after removing the unused tracker parameters ψ and assuming that the tracker perfectly reconstructs v from I, eliminating I entirely. Recall that $\hat{A} \neq A$ from Equation 15 and the discussion thereafter.

Assuming linearity of the tracker according to Equation 8 led to the notion of a tracker that solves a linear system $\hat{A}c = v$. Letting $\hat{A}^{-1}v$ represent the result obtained by the tracker when solving this linear system leads to

$$\begin{aligned}\min_{\theta_{\mathcal{T}}} \sum_k \Big(&\gamma_1 ||\hat{A}^{-1}(\theta_{\mathcal{T}})v_k - c_k||_2^2 + \\ &\gamma_2 ||A(\theta_{\mathcal{R}})\hat{A}^{-1}(\theta_{\mathcal{T}})v_k - v_k||_2^2 + \\ &\gamma_3 ||\hat{A}(\theta_{\mathcal{T}})\hat{A}^{-1}(\theta_{\mathcal{T}})v_k - v_k||_2^2\Big) + \\ &\gamma_\epsilon ||\theta_{\mathcal{T}} - \theta_{\mathcal{R}}||_2^2\end{aligned} \tag{31}$$

in place of Equation 30. Note that errors in the tracker solve, regularization, etc. typically cause $\hat{A}$ and $\hat{A}^{-1}$ to not necessarily cancel. In the space spanned by the v_k, the second term aims to make $\hat{A}^{-1}$ the right inverse of A (driving $\theta_{\mathcal{T}}$ to $\theta_{\mathcal{R}}$ in a subspace), while the third term aims to make $\hat{A}$ and $\hat{A}^{-1}$ the left/right inverses of each other. When $A(\theta_{\mathcal{R}})c_k = v_k$, the second term is merely a scaling of the first term. When $A(\theta_{\mathcal{R}})c_k \neq v_k$, which arises when an inconsistent set of (c_k, v_k) pairs makes Equation 6 overdetermined, the second term provides information beyond that of the first term (penalizing the solution to better match the geometry).

Solving Equation 30 requires the computation of $\frac{\partial \mathcal{T}}{\partial \theta}$ for each of the k terms in the sum. In this linearized scenario, Equations 18 and 25 become

$$\hat{v}(v;\theta_{\mathcal{T}}) = \hat{A}(\theta_{\mathcal{T}})\mathcal{T}(v;\theta_{\mathcal{T}}) \tag{32}$$

$$\begin{aligned}\hat{A}(\theta_{\mathcal{T}})\frac{\partial \mathcal{T}(v;\theta_{\mathcal{T}})}{\partial \theta_{\mathcal{T}}} = \\ -\begin{bmatrix} \mathcal{T}(v;\theta_{\mathcal{T}})^T & 0_{1x3} & 0_{1x3} \\ 0_{1x3} & \mathcal{T}(v;\theta_{\mathcal{T}})^T & 0_{1x3} \\ 0_{1x3} & 0_{1x3} & \mathcal{T}(v;\theta_{\mathcal{T}})^T \end{bmatrix} + \frac{\partial \hat{v}(v;\theta_{\mathcal{T}})}{\partial \theta_{\mathcal{T}}}\end{aligned} \tag{33}$$

where $\mathcal{T}(v;\theta_{\mathcal{T}})$ is found by solving $\hat{A}c = v$. This solve can be accomplished via the inverse (when it exists), least squares (when $\hat{A}$ is full rank), or the pseudo-inverse (to obtain the minimum norm solution, when $\hat{A}$ is not full rank); optionally, the equations can be augmented with Levenberg-Marquardt regularization and solved via the normal equations. In order to estimate $\frac{\partial \hat{v}}{\partial \theta}$, Equation 23 is iterated for various $\widehat{\Delta\theta}$ directions; afterwards, Equation 33 can be solved independently for each column of $\frac{\partial \mathcal{T}}{\partial \theta}$.

For completeness, the derivatives of the four terms in Equation 30 are

$$\frac{\partial \mathcal{L}_{\gamma_1}}{\partial \theta_{\mathcal{T}}} = 2\gamma_1 \sum_k (\mathcal{T}(v_k;\theta_{\mathcal{T}}) - c_k)^T \frac{\partial \mathcal{T}(v_k;\theta_{\mathcal{T}})}{\partial \theta_{\mathcal{T}}} \tag{34}$$

$$\frac{\partial \mathcal{L}_{\gamma_2}}{\partial \theta_{\mathcal{T}}} = 2\gamma_2 \sum_k (A(\theta_{\mathcal{R}})\mathcal{T}(v_k;\theta_{\mathcal{T}}) - v_k)^T A(\theta_{\mathcal{R}}) \frac{\partial \mathcal{T}(v_k;\theta_{\mathcal{T}})}{\partial \theta_{\mathcal{T}}} \tag{35}$$

$$\begin{aligned}\frac{\partial \mathcal{L}_{\gamma_3}}{\partial \theta_{\mathcal{T}}} = 2\gamma_3 \sum_k (\hat{A}(\theta_{\mathcal{T}})\mathcal{T}(v_k;\theta_{\mathcal{T}}) - v_k)^T \Bigg(&\hat{A}(\theta_{\mathcal{T}}) \frac{\partial \mathcal{T}(v_k;\theta_{\mathcal{T}})}{\partial \theta_{\mathcal{T}}} + \\ &\begin{bmatrix} \mathcal{T}(v_k;\theta_{\mathcal{T}})^T & 0_{1x3} & 0_{1x3} \\ 0_{1x3} & \mathcal{T}(v_k;\theta_{\mathcal{T}})^T & 0_{1x3} \\ 0_{1x3} & 0_{1x3} & \mathcal{T}(v_k;\theta_{\mathcal{T}})^T \end{bmatrix}\Bigg)\end{aligned} \tag{36}$$

$$\frac{\partial \mathcal{L}_{\gamma_\epsilon}}{\partial \theta_{\mathcal{T}}} = 2\gamma_\epsilon (\theta_{\mathcal{T}} - \theta_{\mathcal{R}})^T \tag{37}$$

where the term in parentheses in Equation 36 would be replaced by

$$\frac{\partial \hat{A}}{\partial \theta_{\mathcal{T}}} = \frac{\partial \hat{A}(c;\theta_{\mathcal{T}})}{\partial c}\bigg|_{c=\mathcal{T}(v_k;\theta_{\mathcal{T}})} \frac{\partial \mathcal{T}(v_k;\theta_{\mathcal{T}})}{\partial \theta_{\mathcal{T}}} + \frac{\partial \hat{A}(c;\theta_{\mathcal{T}})}{\partial \theta_{\mathcal{T}}}\bigg|_{c=\mathcal{T}(v_k;\theta_{\mathcal{T}})} \tag{38}$$

when $\hat{A}(\mathcal{T}(v_k;\theta_{\mathcal{T}});\theta_{\mathcal{T}})$ is nonlinear.

Motivated by the γ_1 term in Equation 31, the (c_k, v_k) pairs can be stacked together to obtain $\hat{A}^{-1}V - C$ where $V^T\hat{A}^{-T} = C^T$ can be solved separately for each column of $\hat{A}^{-T}$. Since Equation 30 solves for $\hat{A}$, not $\hat{A}^{-1}$, a better alternative is to solve $C^T\hat{A}^T = V^T$ using least squares, minimum norm, or Levenberg-Marquardt. This direct method minimizes

$$\mathcal{L}_D = \sum_i ||C^T\hat{A}^T(\theta_{\mathcal{T}})e_i - V^T e_i||_2^2 \tag{39}$$

where i loops through the rows of $\hat{A}$. Equation 39 will be used to evaluate the efficacy of various approaches to solving Equation 30.

7.1 Avoiding ∞ times 0 singularities

Consider a single pair $(c_1, v_1) = (1, -1)$ where $\hat{A} = \begin{bmatrix}-1\end{bmatrix}$. For now, consider only the γ_1 term. Starting with an initial guess of $\hat{A}_0 = 1$, the tracker solves $\hat{A}_0 C_0 = V$ to obtain $C_0 = \begin{bmatrix}-1\end{bmatrix}$; then, the optimization attempts to drive C from $\begin{bmatrix}-1\end{bmatrix}$ to $\begin{bmatrix}1\end{bmatrix}$ by sending $\hat{A} \to \begin{bmatrix}\infty\end{bmatrix}$ to obtain $C \to \begin{bmatrix}0\end{bmatrix}$. See Figure 1. Unfortunately, the origin is a non-removable singularity, and there is no mechanism to cross over the origin changing the entry in C from negative to positive while flipping the entry in $\hat{A}$ from ∞ to $-\infty$.

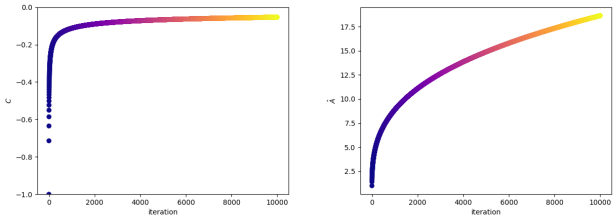

Fig. 1. As $C \to \begin{bmatrix}0\end{bmatrix}, \hat{A} \to \begin{bmatrix}\infty\end{bmatrix}$.

Next, consider

$$\hat{A} = \begin{bmatrix}-1 & 0\\ 0 & -1\end{bmatrix}, C = \begin{bmatrix}1 & 1\\ 2 & 1\end{bmatrix}, V = \begin{bmatrix}-1 & -1\\ -2 & -1\end{bmatrix} \tag{40}$$

where $\hat{A}C = V$. When starting with an initial guess of $\hat{A}_0 = I$, the optimization suffers from the same issue as was discussed in the prior example. See Figure 2.

Changing the initial guess to

$$\hat{A} = \begin{bmatrix}1 & 1\\ 0 & 1\end{bmatrix} \tag{41}$$

achieves the expected results, since the origin is avoided. See Figure 3. Note that perturbing the initial guess is a viable strategy because the optimization is done offline (and only once) in order to determine a suitable tracking rig. Moreover, using $\theta_{\mathcal{R}}$ as an initial guess for $\theta_{\mathcal{T}}$ is likely a good strategy, especially when the tracking rig does not need to be perturbed too much from the animation rig.

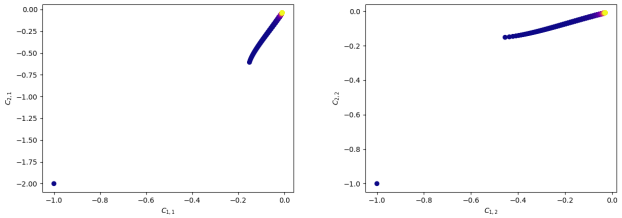

Fig. 2. Starting with $\hat{A}_0 = I$ leads to $C_0 = V$. The left/right sub-figure shows the first/second column of C with each row entry on a separate axis. As the iteration proceeds, the results are color-coded from blue to yellow. All four entries are driven to zero; likewise, all four entries of $\hat{A}$ blow up.

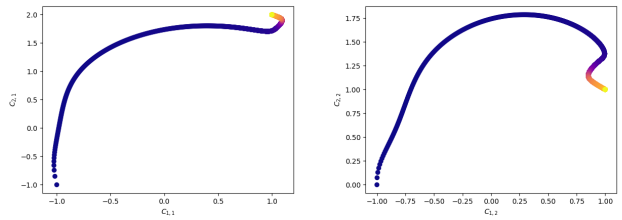

Fig. 3. Changing the initial guess allows both C and $\hat{A}$ to converge correctly.

7.2 Solver Efficacy

Consider the set of pairs

$$C = \begin{bmatrix}1 & 2 & 3 & 1\\ 2 & -1 & 1 & 1\\ 3 & -1 & -2 & 1\end{bmatrix}, V = \begin{bmatrix}-1 & -2 & -3 & -1\\ 4 & -2 & 2 & 2\\ -2 & \frac{2}{3} & \frac{4}{3} & -\frac{2}{3}\end{bmatrix} \tag{42}$$

consistent with a rig

$$A = \begin{bmatrix}-1 & 0 & 0\\ 0 & 2 & 0\\ 0 & 0 & -\frac{2}{3}\end{bmatrix} \tag{43}$$

in order to demonstrate the efficacy of Equation 30 and our approach to solving it. Starting with $\hat{A} = I$, Table 1 demonstrates that $\hat{A} \to A$ using either the γ_1 term or the γ_2 term; moreover, either term drives all three of the γ_1, γ_2, and γ_3 terms to zero. The results obtained via the direct least squares method are shown for the sake of comparison.

	$\mathcal{L}_D$	$\mathcal{L}_{\gamma_1} + \mathcal{L}_{\gamma_2} + \mathcal{L}_{\gamma_3}$	$\|\|\hat{A} - A\|\|_F^2$
Direct	4.58E−12	4.85E−12	4.26E−13
γ_1 only	2.05E−8	2.58E−8	1.31E−8
γ_2 only	1.54E−9	4.15E−9	9.82E−10

Table 1

The results in Table 1 were obtained using all four columns of C and V. The results obtained using only three columns of C and V are similar. Table 2 shows the results of using only the first two columns of C and V with initial guess $\hat{A} = I$. Here, A_2 is the minimum norm solution, unobtainable by Equation 30 without regularizing $\theta_{\mathcal{T}} \to 0$ and undesirable since $\theta_{\mathcal{T}} \to \theta_{\mathcal{R}}$ is preferred. The γ_ϵ term regularizes the unconstrained degree of freedom resulting in $\hat{A} \to A$.

	$\mathcal{L}_D$	$\mathcal{L}_{\gamma_1}+\mathcal{L}_{\gamma_2}+\mathcal{L}_{\gamma_3}$	$\|\|\hat{A}-A_2\|\|_F^2$	$\|\|\hat{A}-A\|\|_F^2$
Direct	1.71E−12	6.89E−13	0	2.77E+0
γ_1 only	4.228E−9	4.90E−12	7.72E+1	6.75E+1
γ_2 only	2.56E−8	1.66E−7	1.62E+2	1.46E+2
γ_1 & γ_ϵ	1.43E−12	7.33E−12	2.77E0	5.74E−9
γ_2 & γ_ϵ	9.94E−14	2.13E−12	2.77E0	2.05E−10

Table 2

Next, consider randomly perturbing the entries of V by 1% to obtain

$$\hat{V}=\begin{bmatrix}-0.990 & -2.00 & -2.97 & -1.00\\ 3.97 & -1.98 & 1.98 & 2.01\\ -1.98 & 0.667 & 1.33 & -0.667\end{bmatrix} \tag{44}$$

so that pairing C with $\hat{V}$ would yield a perturbation of A. The results obtained starting with $\hat{A}=I$ are shown in Table 3, omitting $\mathcal{L}_{\gamma_3}$ since it was on the order of round-off error. Since the pairs over-determine $\hat{A}$, the direct method cannot achieve a small residual and the γ_1 term is bounded from below; however, they both result in the correct least squares solution (specified by $\hat{A}_4$ in the table). Using only the first three columns of C and $\hat{V}$ allows both the direct method and the γ_1 term to achieve significantly smaller errors (as expected) since $\hat{A}$ is no longer over-determined. The γ_2 drives $\hat{A}\rightarrow A$ as expected.

	$\mathcal{L}_D$	$\mathcal{L}_{\gamma_1}$	$\mathcal{L}_{\gamma_2}$	$\|\|\hat{A}-\hat{A}_4\|\|_F^2$	$\|\|\hat{A}-A\|\|_F^2$
Direct	6.60E−4	3.47E−4	2.55E−3	0	5.57E−4
γ_1 only	6.60E−4	3.47E−4	2.54E−3	9.25E−10	5.57E−4
γ_2 only	3.21E−3	2.38E−3	5.65E−9	5.58E−4	3.63E−9

Table 3

Table 4 shows the results obtained using only the first two columns of C and $\hat{V}$ with an initial guess $\hat{A}=I$, again omitting $\mathcal{L}_{\gamma_3}$ since it was on the order of round-off error. The minimum norm solution is unobtainable via Equation 30, as discussed above. Instead, $\hat{A}_{2^*}$ is computed by specifying a third independent column of C, multiplying it by A, and adding it to $\hat{V}$ before solving $C^T\hat{A}_{2^*}^T=\hat{V}^T$. This makes $\hat{A}_{2^*}^T$ agree with the minimum norm solution in the two constrained degrees of freedom and agree with A in the remaining degree of freedom. The γ_1 and γ_2 terms converge as expected, and neither converges to A or $\hat{A}_{2^*}$, due to the unconstrained degree of freedom; however, an initial guess of $\hat{A}=A$ does result in the γ_1 term and the γ_2 terms driving $\hat{A}\rightarrow\hat{A}_{2^*}$ and $\hat{A}\rightarrow A$ respectively (as expected). The γ_ϵ term regularizes the unconstrained degree of freedom driving $\hat{A}\rightarrow A$; however, it competes with the γ_1 term that aims to drive $\hat{A}\rightarrow\hat{A}_{2^*}$ in a two-dimensional subspace.

	$\mathcal{L}_D$	$\mathcal{L}_{\gamma_1}$	$\mathcal{L}_{\gamma_2}$	$\|\|\hat{A}-\hat{A}_{2^*}\|\|_F^2$	$\|\|\hat{A}-A\|\|_F^2$
Direct	1.71E−12	1.20E−12	1.80E−3	2.77E0	2.77E0
γ_1 only	9.89E−9	8.56E−9	1.80E−3	6.66E+1	6.66E+1
γ_2 only	1.13E−2	1.33E−3	8.24E−9	1.13E+2	1.13E+2
γ_1 & γ_ϵ	7.19E−5	1.97E−5	1.24E−3	7.23E−6	9.26E−5
γ_2 & γ_ϵ	1.80E−3	1.33E−3	7.62E−11	1.39E−4	8.44E−9

Table 4

7.3 Improving the search direction via $\frac{\partial\hat{v}}{\partial\theta_\mathcal{T}}$

Consider a tracker $\mathcal{T}(v;\theta_\mathcal{T})=\hat{A}^\dagger v$ where $\hat{A}^\dagger$ depends on the method being used to solve $\hat{A}c=v$. Next, perturb the output of the tracker to

$$\mathcal{T}_1(v;\theta_\mathcal{T})=\hat{A}^\dagger v+\tilde{c} \tag{45}$$

where $\tilde{c}$ captures the fact that trackers often (intentionally, for robustness) do not correctly/precisely invert the rig. Equation 18 becomes

$$\hat{v}(v;\theta_\mathcal{T})=v+\tilde{v}(v;\theta_\mathcal{T})=\hat{A}(\theta_\mathcal{T})\left(\hat{A}^\dagger(\theta_\mathcal{T})v+\tilde{c}(v)\right) \tag{46}$$

where $\tilde{v}\neq 0$. When $\hat{A}$ happens to be invertible,

$$\tilde{v}(v;\theta_\mathcal{T})=\hat{A}(\theta_\mathcal{T})\tilde{c}(v) \tag{47}$$

$$\frac{\partial\hat{v}(v;\theta_\mathcal{T})}{\partial\theta_\mathcal{T}}=\frac{\partial\tilde{v}(v;\theta_\mathcal{T})}{\partial\theta_\mathcal{T}}=\begin{bmatrix}\tilde{c}(v)^T & 0_{1x3} & 0_{1x3}\\ 0_{1x3} & \tilde{c}(v)^T & 0_{1x3}\\ 0_{1x3} & 0_{1x3} & \tilde{c}(v)^T\end{bmatrix} \tag{48}$$

leading to

$$\mathcal{L}_{\hat{v}'}=\sum_k\left\|\frac{\partial\hat{v}_k(v_k;\theta_\mathcal{T})}{\partial\theta_\mathcal{T}}-\begin{bmatrix}\tilde{c}_k(v_k)^T & 0_{1x3} & 0_{1x3}\\ 0_{1x3} & \tilde{c}_k(v_k)^T & 0_{1x3}\\ 0_{1x3} & 0_{1x3} & \tilde{c}_k(v_k)^T\end{bmatrix}\right\|_F^2 \tag{49}$$

as a way of evaluating the efficacy of approximations to $\frac{\partial\hat{v}}{\partial\theta_\mathcal{T}}$.

Plugging Equation 45 into the γ_1 term in Equation 30 and stacking columns leads to $\hat{A}^\dagger V+\tilde{C}-C=0$ or $V-\hat{A}C+\hat{A}\tilde{C}=0$ when $\hat{A}$ happens to be invertible; thus, minimizing the γ_1 term leads to a small value for $V-\hat{A}C+\hat{A}\tilde{C}$ instead of a small value for $V-\hat{A}C$. Table 5 shows the results obtained with and without using $\frac{\partial\hat{v}}{\partial\theta_\mathcal{T}}$ to improve the search direction. In this simple example, auto-diff can be used to compute $\frac{\partial\hat{v}}{\partial\theta_\mathcal{T}}$. The average $\mathcal{L}_{\hat{v}'}$ error obtained when using auto-diff for $\frac{\partial\hat{v}}{\partial\theta_\mathcal{T}}$ was 6.44E−10; for the sake of comparison, the average value of $\sum_k\left\|\frac{\partial\hat{v}_k}{\partial\theta_\mathcal{T}}\right\|_F^2$ was 7.05E−3. Only the first three columns of C and V were used, $\tilde{C}=A^{-1}(\hat{V}-V)$ was chosen so that the perturbations in C resemble those used for V, and the initial guess was $\hat{A}=I$. The γ_2 term gave similar results, as expected since $AC=V$ here.

	$\|\|V-\hat{A}C+\hat{A}\tilde{C}\|\|_F^2$	iters
γ_1 only	3.86E−8	13868
γ_1 & $\frac{\partial\hat{v}}{\partial\theta_\mathcal{T}}$	3.82E−8	12960

Table 5

Next, consider perturbing the output of the tracker to

$$\mathcal{T}_2(v;\theta_\mathcal{T})=\hat{A}^\dagger v+\hat{A}(\theta_\mathcal{T})\tilde{c}(v) \tag{50}$$

instead of Equation 45. Equation 18 becomes

$$\hat{v}(v;\theta_\mathcal{T})=v+\tilde{v}(v;\theta_\mathcal{T})=\hat{A}(\theta_\mathcal{T})(\hat{A}^\dagger(\theta_\mathcal{T})v+\hat{A}(\theta_\mathcal{T})\tilde{c}(v)) \tag{51}$$

and

$$\tilde{v}(v;\theta_\mathcal{T})=\hat{A}(\theta_\mathcal{T})\hat{A}(\theta_\mathcal{T})\tilde{c}(v) \tag{52}$$

$$\begin{aligned}\frac{\partial\hat{v}(v;\theta_\mathcal{T})}{\partial\theta_\mathcal{T}}=&\hat{A}(\theta_\mathcal{T})\begin{bmatrix}\tilde{c}(v)^T & 0_{1x3} & 0_{1x3}\\ 0_{1x3} & \tilde{c}(v)^T & 0_{1x3}\\ 0_{1x3} & 0_{1x3} & \tilde{c}(v)^T\end{bmatrix}+\\ &\begin{bmatrix}\tilde{c}(v)^T\hat{A}^T(\theta_\mathcal{T}) & 0_{1x3} & 0_{1x3}\\ 0_{1x3} & \tilde{c}(v)^T\hat{A}^T(\theta_\mathcal{T}) & 0_{1x3}\\ 0_{1x3} & 0_{1x3} & \tilde{c}(v)^T\hat{A}^T(\theta_\mathcal{T})\end{bmatrix}\end{aligned} \tag{53}$$

replaces Equations 47 and 48 when $\hat{A}$ happens to be invertible. Equation 53 can be used to define a new $\mathcal{L}_{\hat{v}'}$ that replaces Equation 49. Plugging Equation 50 into the γ_1 term in Equation 30 and stacking

columns leads to the notion that minimizing the γ_1 term should lead to a small value of $V - \hat{A}C + \hat{A}^2\tilde{C}$, assuming that $\hat{A}$ happens to be invertible. Table 6 shows the results obtained with and without using $\frac{\partial \hat{v}}{\partial \theta_{\mathcal{T}}}$ to improve the search direction. Once again, this example is simple enough to use auto-diff. The average $\mathcal{L}_{\hat{v}'}$ error obtained when using auto-diff for $\frac{\partial \hat{v}}{\partial \theta_{\mathcal{T}}}$ was 2.54E−10; for the sake of comparison, the average value of $\sum_k \left\| \frac{\partial \hat{v}_k}{\partial \theta_{\mathcal{T}}} \right\|_F^2$ was 1.91E−1.

	$\|\|V - \hat{A}C + \hat{A}^2\tilde{C}\|\|_F^2$	iters
γ_1 only	3.82E−8	29083
γ_1 & $\frac{\partial \hat{v}}{\partial \theta_{\mathcal{T}}}$	3.79E−8	5329

Table 6

7.4 Estimating $\frac{\partial \hat{v}}{\partial \theta_{\mathcal{T}}}$

A general black-box tracker is not necessarily amenable to auto-diff; in such a scenario, $\frac{\partial \hat{v}}{\partial \theta_{\mathcal{T}}}$ would need to be estimated via Equation 23. In order to demonstrate the efficacy of Equation 23, consider minimizing the γ_1 term using either $\mathcal{T}_1$ from Equation 45 or $\mathcal{T}_2$ from Equation 46 along with the analytic values for $\frac{\partial \hat{v}}{\partial \theta_{\mathcal{T}}}$ given in Equations 48 and 53. After about 13000 optimization steps, the analytic solution for $\frac{\partial \hat{v}_k}{\partial \theta_{\mathcal{T}}}$ is used about 39000 times since C and V have three columns.

In order to ascertain the sensitivity of the parameter s in the finite difference approach, an approximation to $\frac{\partial \hat{v}_k}{\partial \theta_{\mathcal{T}}}$ was computed via one iteration of Equation 23 using a randomly chosen $\widehat{\Delta\theta}$ direction and the following values of s: 1E−7, 1E−6, 1E−5, 1E−4, 1E−3, 1E−2, 1E−1, 1E0, 1E1, 1E2. The finite difference approximations for any two values of s can be compared via a Frobenius norm. Let $\mathcal{L}_\Delta(s; \theta_{\mathcal{T}}, \widehat{\Delta\theta})$ be the sum of the Frobenius norms obtained by comparing s to both its next higher and next lower values; then, $\mathcal{L}_\Delta$ indicates the relative sensitivity of the finite difference approximations to perturbations in s. Figure 4 shows the results obtained on all 39000 sub-iterations. The red regions at the bottom of the figures represent round-off errors, and the red region at the top of the right figure represents truncation error (as expected, see e.g. [Gear 1971]). There is no red region at the top of the left figure, since $\frac{\partial \hat{v}_k}{\partial \theta_{\mathcal{T}}}$ is constant (see Equation 48). The large blue region illustrates the ease at which s can be chosen.

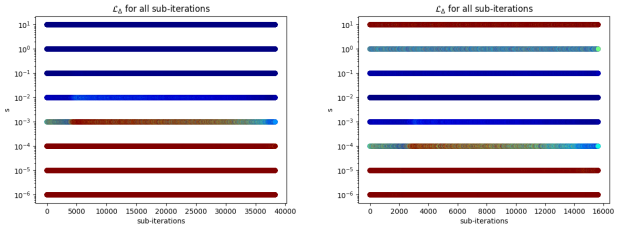

Fig. 4. Color-coded from lower error (blue) to higher error (red). Left: $\mathcal{T}_1$. Right: $\mathcal{T}_2$.

Next, consider the same tests with the analytic solution replaced by various approximations to the $\frac{\partial \hat{v}_k}{\partial \theta_{\mathcal{T}}}$. For the finite difference approximations, s was chosen to minimize $\mathcal{L}_\Delta$. Tables 7 and 8 show the results obtained for $\mathcal{T}_1$ and $\mathcal{T}_2$ respectively using either a number of randomly-chosen $\widehat{\Delta\theta}$ directions, auto-diff, or a single $\widehat{\Delta\theta}$ chosen in the direction of steepest descent. The third column of the tables averages the error (see Equations 49 and 53) over all iterations; for the sake of comparison, the second column averages the solution. As expected, increasing the number of randomly chosen $\widehat{\Delta\theta}$ improves the estimate of $\frac{\partial \hat{v}_k}{\partial \theta_{\mathcal{T}}}$; however, improved estimates of $\frac{\partial \hat{v}_k}{\partial \theta_{\mathcal{T}}}$ did not tend to improve convergence. On the other hand, choosing $\widehat{\Delta\theta}$ in the steepest descent direction did improve convergence. This makes sense from the viewpoint of a predictor-corrector approach, even though $\frac{\partial \hat{v}_k}{\partial \theta_{\mathcal{T}}}$ is poorly estimated as compared to the other approaches. Unfortunately, iterating on steepest descent choices for $\widehat{\Delta\theta}$ did not further improve convergence.

$\mathcal{T}_1$	iters	$\mu\left(\sum_k \left\| \frac{\partial \hat{v}_k}{\partial \theta_{\mathcal{T}}} \right\|_F^2\right)$	$\mu(\mathcal{L}_{\hat{v}'})$
1 random	12781	7.04E−3	4.57E−6
10 random	12921	7.05E−3	2.86E−7
100 random	12964	7.05E−3	4.63E−8
auto-diff	12960	7.05E−3	1.74E−11
steepest descent	9089	2.09E−3	6.41E−3

Table 7

$\mathcal{T}_2$	iters	$\mu\left(\sum_k \left\| \frac{\partial \hat{v}_k}{\partial \theta_{\mathcal{T}}} \right\|_F^2\right)$	$\mu(\mathcal{L}_{\hat{v}'})$
1 random	4763	1.71E−1	3.50E−4
10 random	5307	1.93E−1	6.32E−6
100 random	5329	1.94E−1	1.24E−6
auto-diff	5329	1.94E−1	2.57E−10
steepest descent	4825	7.28E−2	1.93E−1

Table 8

8 SIMON SAYS ANIMATION RIG CREATION

The industry-standard rig calibration pipeline begins by placing an actor into a light-stage where they are asked to create a variety of expressions, each of which can be defined by a set of animator controls c. Afterwards, the data is used to create 3D geometry v corresponding to each captured expression. This is a time-consuming process that requires both artist expertise and subjectivity. In fact, some of the expressions are inferred and/or sculpted as opposed to being reconstructed, e.g. `smile` would be split into `smile_left` and `smile_right`. To simplify rig calibration, we utilize the Simon-Says approach proposed in [Zhu et al. 2024].

In order to guide the user, a version of "Simon" is required. This can be achieved by reconstructing the user's neutral geometry N and subsequently fitting it with an animation rig $\mathcal{R}$. The ability to reconstruct neutral geometry is becoming more common, see Section 2, and commercial software is becoming available. Given a rig fitted to any preexisting geometry, volumetric morphing can be used to transfer that rig to any newly constructed geometry once a topological correspondence is determined. See e.g. [Zhu et al.

2024] for more details. In particular, we utilize the MetaHuman animation rig framework (see [Epic Games 2025]); however, since their framework includes a blendshape offset in the neutral pose that can lead to various inaccuracies when making expressions (see [Zhu et al. 2024]), we volumetrically morph the rig to tightly fit the actual geometry. Note that auxiliary attributes such as textures, hair, eyebrows, etc. are selected from a pre-defined database since they are not the focus of this paper.

A carefully chosen set of Simon-Says expressions is used to calibrate each rig to the motion signatures of the relevant user. The expression set should cover the most important rig parameters while still remaining minimal with expressions that are straightforward for an average person to make. In addition, expressions that would typically require artist intervention need to be omitted. In total, nineteen expressions

- neutral, brows down, brows up, eyes wide, eyes close, nose wrinkle, cheek puff, teeth grimace, corner pull, mouth stretch, corner depress, lip press, pursed lips, mouth funnel, lip bite, jaw open, jaw open extreme, jaw left, jaw right

are used for person-to-person retargeting. The nineteen expressions along with their associated animation controls are shown in Figure 5. For person-to-game/VR character retargeting, the expressions should be chosen in a manner that captures the distinct motion signatures of the game/VR character.

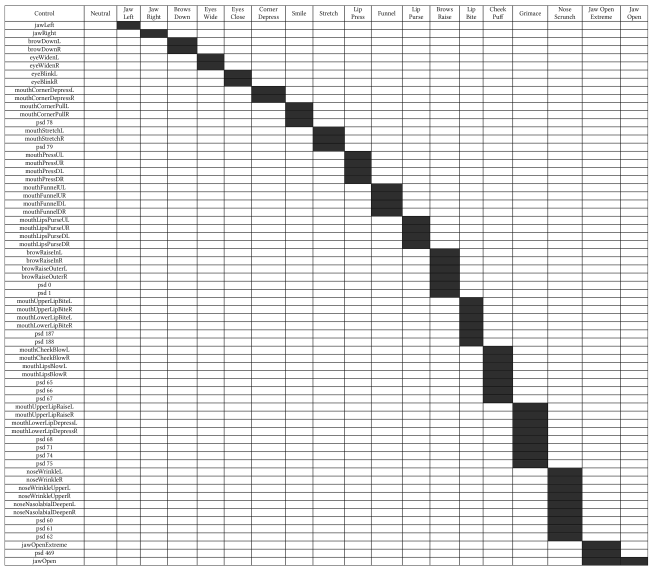

Fig. 5. For each expression, the associated column shows the active MetaHuman PSD rig controls.

For each expression, an image $\phi(\mathcal{R}(c; N, \theta_{\mathcal{R}}))$ is shown to the user. Here, ϕ represents a renderer with camera parameters set to mimic the user's webcam. The user then attempts to replicate each expression. When they feel as though they are correctly reproducing the expression, an image I of their face is recorded. See Figure 6. Visual cues enable non-experts to quickly understand each expression, especially when the cues are similar in appearance to the user (as they are here). Importantly, since the animation rig is unlikely to already contain the correct semantic motion signatures of the user, it is incorrect for the user to simply aim to match each image. For example, `jaw_open` and `jaw_open_extreme` should match the user's actual biomechanics, which may differ significantly from those in the rig. Thus, concise verbal descriptions are also provided for each pose.

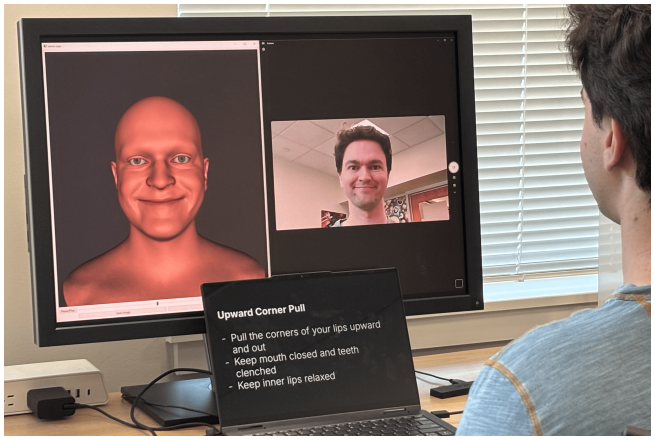

Fig. 6. Our proposed Simon-Says capture setup.

Once the Simon-Says session is complete, the captured images can be used to reconstruct geometry (see section 9). Given the (c, v) pairs, Equation 1 can be solved to determine rig parameters $\theta_{\mathcal{R}}$ that better capture the user's motion signatures. Expert riggers typically utilize a number of constraints during the rig design process. For example, `jaw_open` plus `lips_together` should properly seal the lips together. It is straightforward to incorporate such constraints by including additional (c, v) pairs, representing these constraints, into the Equation 1 solve. Similarly, we can also include surrogate (c, v) pairs for any expressions that were omitted because they would typically require artist intervention. Although these surrogate pairs would not be user-specific, they do help with regularization.

8.1 Incorrect Expressions

When the user struggles to make the correct expression (such as accidentally flexing eyebrows while smiling), the reconstructed geometry v will not correctly correspond to the expression specified by c. It is possible to roughly identify the additional controls that need to be activated in order to explain these spurious geometric deformations. Using the current best guess for the animation rig parameters $\theta_{\mathcal{R}}$, a tracker can be used to compute

$$c^+ = c + (I_{n\times n} - \mathcal{H}(c))\mathcal{T}(I, \theta_{\mathcal{R}}) \tag{54}$$

where $\mathcal{H}(c)$ is a diagonal matrix of Heaviside functions $H(|c_i|)$. Equation 54 leaves the nonzero entries of c unmodified while augmenting the other entries to agree with $\mathcal{T}(I, \theta_{\mathcal{R}})$. The newly added (c^+, v) pairs help to prevent the inappropriate modifications to $\theta_{\mathcal{R}}$ that would have resulted from the (c, v) pairs that they replace when solving Equation 1. Alternatively, the geometry can be constrained to mask out regions of the face that should not deform for the given expression, replacing (c, v) pairs with (c, v^-) pairs.

The augmentation of c to c^+ can be problematic when the added controls interfere with the primary expression under consideration; thus, it is desirable to leave interfering controls unmodified. An

alternative strategy consists of constraining $\theta_{\mathcal{R}}$ so that the degrees of freedom irrelevant to c do not change. This is plausible when solving Equation 1, since $\mathcal{R}$ is typically a straightforward deterministic function; however, it can be daunting when solving Equation 16, since $\mathcal{T}$ contains both an ill-posed geometric reconstruction and an inverse problem (rig inversion).

8.2 Rig Co-Animation

It is worth briefly addressing the importance of well-calibrated rigs when it comes to retargeting. It makes no sense to strive for an accurate tracking solve on a performer if those controls are not going to be sensical on the target. This consideration emphasizes the importance of the Simon-Says rig calibration. For example, if the performer opens their jaw as far as they can and `jaw_open` is 0.8 instead of 1, then the performer will be unable to fully open the jaw of the target. If the performer misunderstood the instructions and did not open their jaw fully, then (in the spirit of Equation 54) the `jaw_open` = 1 in c should be lowered to a more appropriate value.

8.3 Validation with Synthetic Data

In this section, we describe the results of an experiment that used a MetaHuman with hand-crafted animation rig parameters θ_{GT} that could be considered a ground truth (or close to it). Starting with a different MetaHuman, we morphed its rig onto the MetaHuman being used for this test. Labeling the morphed rig parameters as θ_M, we used a Simon-Says expression set to generate geometry with θ_{GT} and subsequently optimized $\theta_M \rightarrow \theta_S$. For the Simon-Says expressions, θ_{GT} and θ_S behaved similarly, as expected. Generalization of our Simon-Says approach can be tested by comparing the geometry produced by θ_{GT} to that produced by θ_S for other expressions unseen by the optimization. In particular, the benefits of the approach can be quantified as the decrease in errors obtained by using θ_S instead of θ_M. See Table 9.

Expression	θ_M	θ_S
Happy Speaking	0.487	0.285
Partial Stretch	0.693	0.370
Funnel Asymmetric	0.308	0.143
Funneled Speech	0.393	0.183
"Ur"	0.507	0.337
"Em"	0.211	0.176
"Vee"	0.247	0.208
Frown	0.329	0.119

Table 9. Average error in mm for synthetic expressions unseen by the Simon-Says optimization.

9 EXPRESSION CAPTURE

Assuming the existence of a well-constructed neutral geometry, this section focuses on reconstructing geometry that corresponds to the images captured in a Simon-Says session. Although any of the approaches discussed in Section 2 could be used, higher quality geometry reconstruction likely leads to an animation rig that better captures the user's motion signature. Thus, we briefly address geometry reconstruction in this section. Moreover, we discuss geometry reconstruction in the context of improving upon (or bootstrapping from) state-of-the-art techniques, as opposed to starting from scratch. See Figure 8 for some representative results, and Figure 7 for an overall outline of the approach. Finally, although we intentionally treat trackers via a black box ansatz, it is worth noting that improved geometry reconstruction techniques could be used to compute $\tilde{v}_{\mathcal{T}}$ (see Section 3).

Even though the light-stage is often discussed as being the top end data acquisition method for constructing neutral geometry, textures, and animation rigs, the overly confining equipment can adversely impact the authenticity of the user's expressions, hindering the ability to construct an accurate basis for their motion signatures. In fact, we prefer monocular, multi-camera, or head-mounted camera (HMC) setups because they are all less restrictive on the user's movements and expressions. A monocular approach is the least preferable option, since the data term (without robust priors) only includes feature alignment in the image plane instead of in $\mathbb{R}^3$.

9.1 Bridging the Domain Gap

It can be difficult to ascertain correspondences between synthetic geometry and real-world images, and the lack of a proper correspondence leads to inaccuracies in the geometry reconstruction. Even after calibrating the virtual camera to match the real-world camera's intrinsics and estimate the scene extrinsics (typically with the aid of a landmark detector), inaccuracies in the texture and lighting lead to a domain gap that hinders proper correspondence.

We address the domain gap via the screen-space warping technique from [Zhu et al. 2024]. Given a synthetically rendered image created to match a real-world image as closely as possible, a neural segmentation of both images is used to construct an optical flow field that warps the real-world image to better match the synthetic image; then, the photon mapping method from [Lin et al. 2022] is used to project samples from the warped real-world image into the synthetic geometry's texture where the samples can be robustly gathered to texels. This baking of the entangled real-world texture and lighting into the synthetic geometry's texture better enables an inverse rendering pipeline to bridge the domain gap. Typically, this process is repeated every time (e.g. every iteration of optimization) the geometry is updated.

Since the Simon-Says calibration of the animation rig (and our modification of the tracking rig) only needs to be done once, it is worth considering a slightly more invasive, yet still democratizable, approach. Easy-to-remove dots can be placed on the face using liquid eyeliner or other makeup. After manually selecting their screen-space locations in the neutral pose, their locations can subsequently be tracked throughout a video (see e.g. [Doersch et al. 2023; Karaev et al. 2023]). During extreme poses, dot locations may be lost and require re-labeling; alternatively, problematic frames can be sandwiched by tracking forward and backward through time starting from frames of higher confidence. Dot correspondences between the real and synthetic images can be used in the place of neural segmentation in order to construct an improved optical flow field for warping. This requires that dots must also be present in the synthetic geometry's texture. Although one could label them manually, we instead use a well-aligned neutral expression in order to project the dots from the real-world image into the synthetic geometry's texture (note that this assumes an accurate neutral geometry).

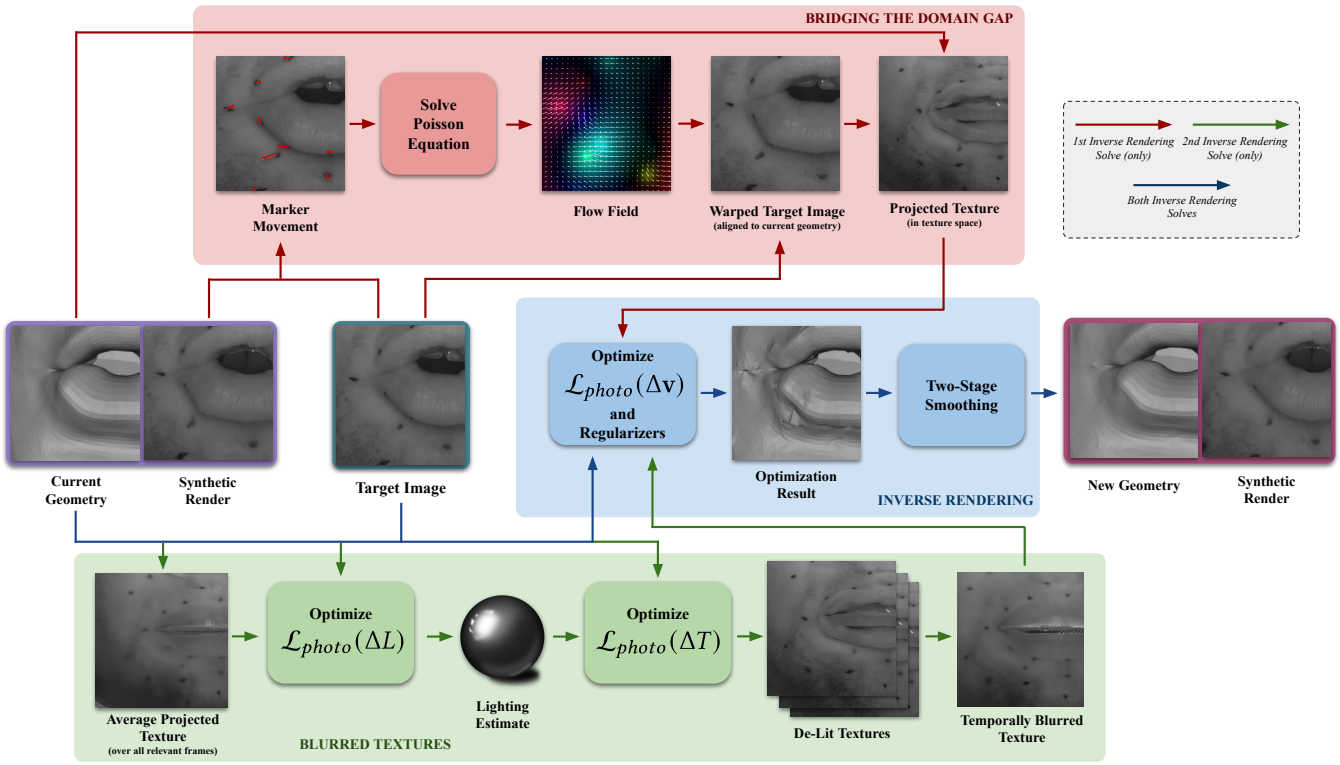

Fig. 7. A high-level overview of our expression capture pipeline.

The dot-based optical flow field is constructed by solving a Poisson equation (using a standard 5-point stencil) on the synthetic image with Neumann boundary conditions on the image edges and Dirichlet boundary conditions at the locations of the synthetic dots. It is important to note that point-based Dirichlet boundary conditions rapidly decay (as compared to line-based conditions) in $\mathbb{R}^2$, as was discussed in [Cong et al. 2015]. We developed a number of techniques aimed at alleviating this issue to some degree. Firstly, and perhaps most importantly, the dots should be placed in the key areas of interest where sliding would lead to large errors. For example, dots should be placed around the edge of the lip line and near the nasolabial folds. In addition, dots should be placed with regular density throughout the face. Secondly, in order to increase the support and decrease the falloff, the entire dot (not just the dot center) should be used for Dirichlet boundary conditions. Since only a single pixel is tracked for each dot, the velocity from that pixel is redistributed to all the surrounding pixels associated with that dot in order to provide values for the Dirichlet boundary conditions.

9.2 Inverse Rendering

Given a sequence of frames, the MetaHuman Animator [Epic Games 2025] is used to initialize a rough approximation to the reconstructed geometry on every frame. Each iteration, every real-world image in the sequence is warped and projected onto its corresponding synthetic geometry (see Section 9.1). Since differentiable renderers work better with smaller perturbations, only a fraction (increasing linearly from 10% to 100% as the iterations proceed) of the optical flow is used for the warp. Next, an off-the-shelf differentiable renderer [Ravi et al. 2020] is used to improve the current estimate to the reconstructed geometry in each frame. The data term is a per-pixel photometric loss $\mathcal{L}_{photo}(\Delta \mathbf{v}) = \sum_i \left[\Psi(\mathbf{v} + \Delta \mathbf{v}, T, L)_i - \mathcal{I}_i^R\right]^2$ that compares the synthetic image obtained from a differentiable renderer Ψ to the real-world image $\mathcal{I}^R$. Here, $\mathbf{v}$ are the vertex positions of the geometry, and $\Delta \mathbf{v}$ is a perturbation to the vertex positions. Since the texture T computed according to Section 9.1 already contains baked-in lighting, only a simple model of ambient lighting is required for L; importantly, the ambient lighting should leave the texture colors unchanged. In addition to the data term, a number of regularization terms are used: a penalty on changes in edge lengths (as compared to the initial rough approximation to the geometry), a penalty on the Laplacian (using a one-ring stencil) of vertex displacements, a penalty on deviations of face normals (as compared to the initial rough approximation to the geometry), and a penalty on the strain energy of interior tetrahedra when a volumetric model is used. The strain energy term is quite useful for preserving lip volume and thus for creating lip bulge.

The regularization terms tend to overly prevent the data term from reaching a result consistent with matching the marker positions between the real and synthetic images. Thus, we use weaker weights on the regularization terms sacrificing mesh quality in order

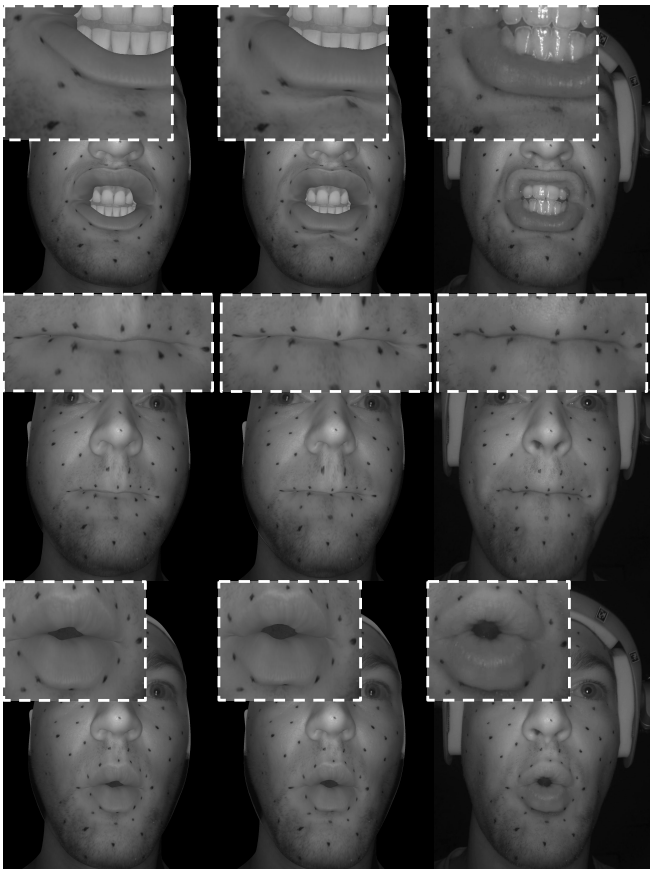

Fig. 8. Geometry reconstruction results on some difficult expressions. 1st column: Initial reconstructions (from MetaHuman Animator). 2nd column: Our results. 3rd column: Target images. 1st row: The lip bulge has been corrected to better match a "funnel" expression. 2nd row: The inward lip roll has been increased to better match a "lip bite" expression. 3rd row: The inner lip lines have been corrected to better match an "oh" expression.

to obtain a better result for marker matching. Afterwards, a post-process is used to improve the quality of the mesh without adversely affecting the matching of the markers. This is achieved via a two-stage Laplacian smoothing algorithm. In the first stage, the vertex deltas corresponding to any vertices within a topological threshold distance to a facial marker are fixed in place while all other vertex deltas are smoothed. In the second stage, the complementary set of vertex deltas are held fixed, and the marker-associated deltas are smoothed. This results in a mesh with high marker reconstruction accuracy without high-frequency artifacts.

9.3 Blurred Textures

Although the process outlined in Section 9.2 creates geometry that matches the marker positions well, some regions (such as the inner lip) end up with overly noisy geometry. This can be alleviated by a second inverse rendering solve, which is only used to optimize specific problematic regions of the face such as the inner lip area.

For each frame, the real-world image is (optionally) warped before projecting it onto the corresponding initial rough approximation to the geometry. Then, a single blurred texture is created for a sequence of frames by averaging the projected texel values over all relevant frames. This blurred texture is used along with the rough approximation to the geometry in order to determine a lighting approximation L for the sequence by minimizing $\mathcal{L}_{photo}(\Delta L) = \sum_i \left[\Psi(\mathbf{v}, T, L+\Delta L)_i - \mathcal{I}_i^R\right]^2$ across all frames. A spherical harmonics approximation [Ramamoorthi and Hanrahan 2001] to the lighting L was used. In texels where the projected values do not vary much over the sequence, the averaged texture is close to the unaveraged values causing the lighting model to aim for an ambient-only explanation. In texels where the averaged texture varies greatly from the unaveraged values, the diffuse components of the lighting model strive to explain those variations in a manner that is consistent with the animated normals of the geometry. Similarly, specular components (if used) would strive to explain variations in a manner consistent with both animated normals and the camera viewing angle.

After estimating the lighting L, a per-frame de-lit texture is calculated by minimizing $\mathcal{L}_{photo}(\Delta T) = \sum_i \left[\Psi(\mathbf{v}, T + \Delta T, L)_i - \mathcal{I}_i^R\right]^2$. Subsequently, each per-frame de-lit texture is blurred by temporally averaging (with a kernel) the texel values in a time window. Generally speaking, temporally blurred textures can improve the results obtained via inverse rendering by creating additional feature overlap (similar in spirit to the incremental warping) and by smoothing out spurious sharp features (such as specular highlights) facilitating differentiability in image space. For similar reasons, it may also be helpful to blur parts of the target image.

The second inverse rendering solve uses $\mathcal{L}_{photo}(\Delta \mathbf{v})$ with the lighting obtained from $\mathcal{L}_{photo}(\Delta L)$ and the texture obtained from $\mathcal{L}_{photo}(\Delta T)$. It is important to stress that temporal blurring typically hinders the all-important matching of marker positions. This is why we restrict the second inverse rendering solve to problematic regions of the face such as the lip region where high curvature can lead to ill-conditioned specular highlights and the inner lip region where the occlusion boundary can cause discontinuities in the projected texture.

9.4 Inner Mouth and Teeth

When the mouth opens, the ability to match a synthetic rendering of the inner mouth and teeth with the corresponding areas of the real-world image helps to avoid confusing those areas with occlusion boundaries of the lips. Thus, we use template teeth, gums, and inner mouth geometry and textures. For each frame, scalar intensity modifiers are calculated and used to uniformly scale the template textures. One scalar is calculated for the teeth/gums region and a separate scalar is calculated for the remainder of the inner mouth. Although segmentation of the ground-truth image works well to identify the inner occlusion boundaries of the lips, allowing for a straightforward comparison of the inner mouth with the corresponding pixels in the image, segmenting out the teeth and/or the teeth and gums together proved to be more difficult. Thus, the scalar multiplier for the teeth/gums textures is calculated by assuming

that the teeth/gums region is aligned with the associated region in the image; obviously, this leads to a source of error.

10 OPTIMIZING THE TRACKING RIG

Assuming that the performer has had their rig calibrated to some $\theta_{\mathcal{R}}$ using Simon-Says in Section 8, we further optimize that rig to a $\theta_{\mathcal{T}}$ more appropriate for tracking. The Simon-Says images or any other images can be used for this, as long as aspirational (I, c) pairs exist. Of course, c should be modified to c^+ via Equation 54 and other modifications, as discussed in Section 8.1. Using the (I, c^+) pairs, Equation 16 can be written as

$$\begin{aligned}\min_{\theta_{\mathcal{T}}} \sum_k \Big(& \gamma_1 ||\mathcal{T}(I_k;\theta_{\mathcal{T}}) - c_k^+||_2^2 + \\ & \gamma_2 ||\mathcal{R}(\mathcal{T}(I_k;\theta_{\mathcal{T}});\theta_{\mathcal{R}}) - v_k^+||_2^2 \Big) + \\ & \gamma_\epsilon ||\theta_{\mathcal{T}} - \theta_{\mathcal{R}}||_2^2\end{aligned} \tag{55}$$

where $v^+ = \mathcal{R}(c^+;\theta_{\mathcal{R}})$. Equations 18 and 25 can be written as

$$\hat{v}(I;\theta_{\mathcal{T}}) = \mathcal{R}(\mathcal{T}(I;\theta_{\mathcal{T}});\theta_{\mathcal{T}}) \tag{56a}$$

$$\begin{aligned}\left.\frac{\partial \mathcal{R}(c;\theta_{\mathcal{T}})}{\partial c}\right|_{c=\mathcal{T}(I;\theta_{\mathcal{T}})} & \left.\frac{\partial \mathcal{T}(I;\theta)}{\partial \theta}\right|_{\theta=\theta_{\mathcal{T}}} = \\ & - \left.\frac{\partial \mathcal{R}(\mathcal{T}(I;\theta_{\mathcal{T}});\theta)}{\partial \theta}\right|_{\theta=\theta_{\mathcal{T}}} + \left.\frac{\partial \hat{v}(I;\theta)}{\partial \theta}\right|_{\theta=\theta_{\mathcal{T}}}\end{aligned} \tag{56b}$$

in order to differentiate Equation 55. The pseudo-inverse of the normal equations can be used to solve for $\frac{\partial \mathcal{T}}{\partial \theta}$ in Equation 56b.

10.1 MetaHuman Framework

The MetaHuman Animator (MHA) tools suite was chosen because its size and complexity is representative of similar industry frameworks. For the animation controls, $c \in \mathbb{R}^{174}$ can be divided into 97 primary controls and 77 so-called "tweaker" controls. They are combined in various ways to create 814 pose-space deformation (PSD) controls. The MetaHuman animation rig controls 24,049 vertices via 870 joints, each with 9 degrees of freedom (3 translation, 3 rotation, 3 scaling). For the sake of implementation, we implement the rig to output the 7,830 degrees of freedom associated with the joints instead of the 72,147 degrees of freedom associated with the vertices. This optimization ignores the fact that the PSD controls can also affect blendshape correctives; instead, it assumes that the final mesh can be obtained merely by skinning the joints. The discussions throughout this paper are valid in either case, but this simplification makes the optimization more tractable. The animation rig determines the joint degrees of freedom by multiplying the joint matrix by the PSD controls. The size $7{,}830 \times 814$ joint matrix has only 745,284 nonzero entries, meaning that $\theta_{\mathcal{R}} \in \mathbb{R}^{745284}$.

In order to differentiate the tracker via Equation 56b, $\frac{\partial \mathcal{R}}{\partial c} \in \mathbb{R}^{(7830,174)}$ and $\frac{\partial \mathcal{R}}{\partial \theta} \in \mathbb{R}^{(7830,745284)}$ would need to be computed. In order to make $\frac{\partial \mathcal{R}}{\partial \theta}$ tractable, each term in the sum in Equation 55 is minimized over only the nonzero θ in the columns of the joint matrix corresponding to the PSD controls that are not identically zero according to c^+. This greatly reduces the dimension of θ for any reasonable (c^+, v^+) pair. For example, the nineteen expressions (ignoring the neutral) used for the Simon-Says capture have their dimensionality reduced to a far more tractable 1589, 6456, 1470, 1470, 9470, 15252, 11465, 7890, 7939, 2868, 4050, 3198, 12936, 4051, 7262, 3758, 7506, 2594, 2594, respectively. Of course, reducing each expression to a few thousand parameters means that hundreds of expressions (depending on expression overlap) would be needed to cover the full rig. We circumvent this by limiting our expression set to cover only the most important controls, noting that this leaves $\theta_{\mathcal{T}}$ fixed to its $\theta_{\mathcal{R}}$ values for the parameters that do not appear in the expression set (in the spirit of the last term of Equation 55).

10.2 Tracker Variants

Our optimization strategy will make use of the following variations of Equation 55. For each expression, i.e. each term in the sum in Equation 55, let $\mathcal{H}$ be a diagonal matrix of Heaviside functions $H(|c_i^+|)$. Let $\mathcal{T}_{\mathcal{H}} = \mathcal{H}\mathcal{T}$ represent the filtering of the tracker to zero out entries that are zero in c^+. Let $\mathcal{H}_D$ be the decimation of $\mathcal{H}$ into a wide matrix via the elimination of rows that are entirely full of zeros; then, $c_D = \mathcal{H}_D c$ is the subset of c containing all relevant controls, and $\mathcal{H}_D\mathcal{T}$ is a similarly decimated tracker. Although $\mathcal{T}_{\mathcal{H}}$ and $\mathcal{H}_D\mathcal{T}$ are essentially equivalent, $\mathcal{H}_D\mathcal{T}$ facilitates code optimizations. Let θ_D be the subset of the rig parameters that depends on c_D, let $\mathcal{R}_D(c_D;\theta_D)$ be the subset of the rig that can be modified by c_D, and let $\mathcal{T}_D(I;\theta_D)$ represent a reduced tracker that only considers the reduced degrees of freedom from c_D. In contrast to $\mathcal{T}_{\mathcal{H}}$, $\mathcal{T}_D$ is not allowed to use filtered out degrees of freedom in order to explain the geometry.

The γ_1 term in Equation 55 can use any of the three trackers, decimating c^+ for $\mathcal{T}_D$. The γ_2 term in Equation 55 can use any of the three trackers modifying $\mathcal{R}$ to $\mathcal{R}_D$ for $\mathcal{T}_D$; in addition, v^+ should be decimated if the output of $\mathcal{R}_D$ is a decimated version of the output of $\mathcal{R}$. Superscripts on γ (i.e. γ, $\gamma^{\mathcal{H}}$, γ^D) will be used to indicate which tracker was used in a specific term in Equation 55. Equation 56 can be used for any of the three trackers, modifying $\mathcal{R}$ to $\mathcal{R}_D$ for $\mathcal{T}_D$; in addition, v^+ should be decimated if the output of $\mathcal{R}_D$ is. Inserting $\mathcal{T}_{\mathcal{H}}$ into Equation 56b and using the normal equations on $\frac{\partial \mathcal{R}}{\partial c}\mathcal{H}$ leads to a coefficient matrix with nonzero entries corresponding to the normal equations for $\mathcal{R}_D$, i.e. $\frac{\partial \mathcal{T}_{\mathcal{H}}}{\partial \theta}$ agrees with $\frac{\partial \mathcal{T}_D}{\partial \theta}$ when it is nonzero. This highlights the aforementioned code optimizations for $\frac{\partial \mathcal{T}_{\mathcal{H}}}{\partial \theta}$, which consist of replacing $\mathcal{R}$ with $\mathcal{R}_D$ and $\mathcal{T}_{\mathcal{H}}$ with $\mathcal{H}_D\mathcal{T}$.

10.3 Optimization Strategy

A typical rig contains primary controls that retarget well and auxiliary controls that one might want to ignore when retargeting to a new character. Unfortunately, when a tracker ignores the auxiliary controls, the primary controls end up polluted as they work too hard to explain geometry that would have been better explained by auxiliary controls. Thus, a tracker should aim to use the entire set of controls; however, it should lean more heavily on the primary controls as opposed to the auxiliary controls. The trick is to figure out how to use just enough of the auxiliary controls in order to obtain non-polluted values for the primary controls. Motivated by this, we propose modifying the rig used by the tracker (i.e. $\theta_{\mathcal{T}}$) so that the primary controls properly explain the geometry and so that the auxiliary controls do not attempt to explain the geometry. This can be seen as an attempt to "orthogonalize" the rig used by the tracker.

11 OPEN-SOURCE TRACKER

We begin by considering an open-source tracker (using gradient descent, L-BFGS [Fletcher 1987], etc.) where one is able to modify $\mathcal{T}$ to $\mathcal{T}_D$; in Section 12, we consider a black-box tracker where one cannot make use of $\mathcal{T}_D$. Let θ_A be the internal rig parameters obtained from the MetaHuman auto-rigging service, and let θ_S be the result obtained by modifying the rig parameters via Simon-Says (see Section 8). Either θ_A or θ_S can be used as $\theta_{\mathcal{R}}$ in the rig (and as $\theta_{\mathcal{R}}$ in the γ_2 term); in addition, either can be used as the initial guess for $\theta_{\mathcal{T}}$. In this section, we show how to improve the semantics of the tracker by modifying $\theta_A \to \hat{\theta}_A$ and $\theta_S \to \hat{\theta}_S$. All the trackers use the same reconstructed geometry from Titan, again assuming that obtaining that geometry reconstruction is more straightforward than rig inversion; thus, the trackers will be functions of the geometry, not the images. As shown in Figures 9 and 10, changing $\theta_A \to \theta_S$ via Simon-Says or modifying $\theta_A \to \hat{\theta}_A$ and $\theta_S \to \hat{\theta}_S$ has little effect on the ability of the tracker to invert the rig and match the geometry; however, the tracker output and thus the semantic interpretation can vary significantly. All the Simon-Says expressions were used in the optimization.

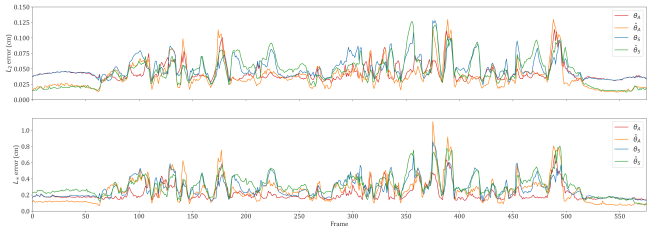

Fig. 9. A 573 frame pangram was used to test the tracker's ability to invert the rig and match the reconstructed geometry. The tracker was mostly able to adequately minimize geometry errors using any of θ_A, $\hat{\theta}_A$, θ_S, $\hat{\theta}_S$, even though the output controls (and thus the semantic interpretation) can vary significantly. See also Figure 10.

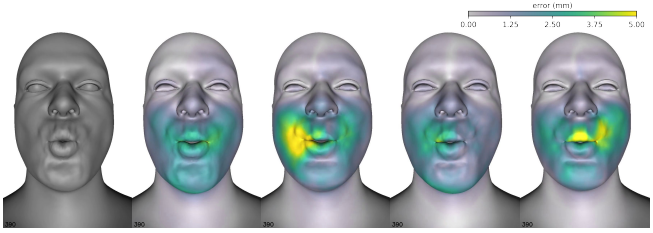

Fig. 10. Ground-truth reconstructed geometry (left) as compared to the geometry output by the tracker using θ_A, $\hat{\theta}_A$, θ_S, $\hat{\theta}_S$, respectively. This corresponds to frame 390, which was chosen because it has relatively large errors, of Figure 9. Colored regions indicate reconstruction error according to the colorbar.

Our proposed strategy for improving the tracker, either $\theta_A \to \hat{\theta}_A$ or $\theta_S \to \hat{\theta}_S$, begins by utilizing a limited set of rig parameters θ_D along with $\mathcal{R}_D$ and $\mathcal{T}_D$ in order to explain the geometry via a limited set of controls c_D. Thus, γ_1^D and γ_2^D are used in Equation 55. See Figure 11. Importantly, we do not treat the results of Equation 55 as the final solution along the lines of solving an inverse problem. Instead, motivated by machine learning, we use the optimization simply to generate samples in parameter space. In machine learning applications, holdout data is used to pick the most salient sample. We instead use the full tracker $\mathcal{T}$ in order to supervise the process. That is, modifying $\theta_{\mathcal{T}}$ to $\hat{\theta}_{\mathcal{T}}$ by replacing the parameters corresponding to θ_D with those of a generated sample, the various $\mathcal{T}(v; \hat{\theta}_{\mathcal{T}})$ can be analyzed to determine the best $\hat{\theta}_{\mathcal{T}}$. Given the best $\hat{\theta}_{\mathcal{T}}$, as supervised by $\mathcal{T}$, this is used to bootstrap improvement of $\mathcal{T}$ itself.

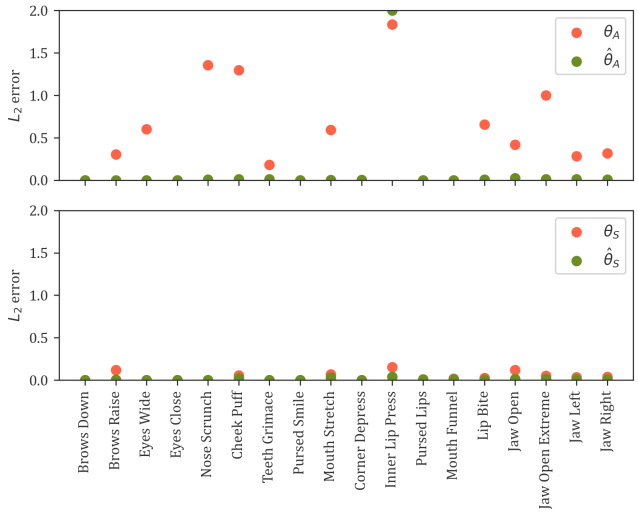

Fig. 11. L_2 errors (according to γ_1^D) on the various expressions before (red) and after (green) optimizing $\theta_A \to \hat{\theta}_A$ (top) and $\theta_S \to \hat{\theta}_S$ (bottom) using only the primary controls on each expression via γ_1^D and γ_2^D.

In the second stage of the process, we switch from $\mathcal{T}_D$ to $\mathcal{T}_{\mathcal{H}}$, still optimizing only the same columns as before (i.e. for the primary controls only) but aiming to get the full tracker to also give the desired output. Thus, $\gamma_1^{\mathcal{H}}$ and $\gamma_2^{\mathcal{H}}$, where $\mathcal{H}$ selects only the primary controls, are used in Equation 55. See Figure 12. In the third stage, again optimizing the same columns, we aim to minimize the contributions of the spurious controls by adding a second $\gamma_1^{\mathcal{H}}$ term that selects problematic spurious controls aiming to set them to zero. Multiple such $\gamma_1^{\mathcal{H}}$ terms can be used to strategically drive spurious controls to zero based on various priorities. Note that the $\gamma_2^{\mathcal{H}}$ term is not changed from the previous stage, i.e. it still only selects the primary controls. See Figure 13. At this point in the process, the hope would be that the columns corresponding to the primary controls have been adjusted to explain as much of the expression as possible. Since the Simon-Says expression set was chosen to be minimal for the sake of efficiency, the expressions have minimal overlap allowing each term in Equation 55 to be run separately and in parallel. The only overlap in Figure 5 is between `jaw_open` and `jaw_open_extreme`, and it was not problematic to consider them independently. In the final (fourth) stage of the process, columns corresponding to spurious controls are considered. Since the spurious controls can have a lot of overlap, every relevant expression should be included in the sum in Equation 55 when considering them. Again, multiple $\gamma_1^{\mathcal{H}}$ terms can be used in order to select the controls of interest. Note that the $\gamma_2^{\mathcal{H}}$ still remains unchanged, i.e. it still only selects the primary controls. See Figure 14.

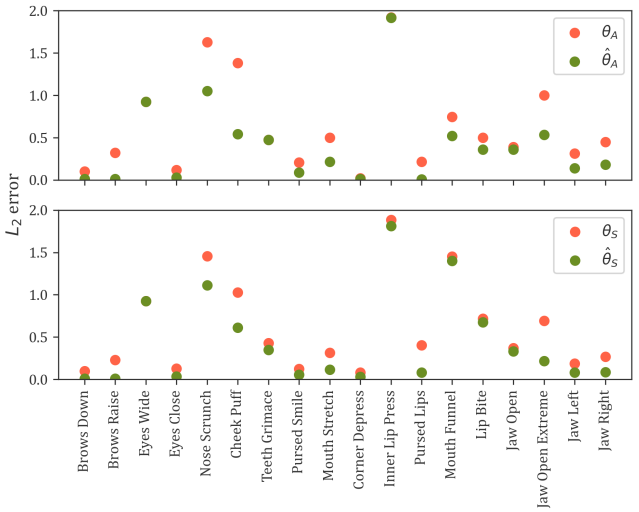

Fig. 12. L_2 errors (according to $\gamma_1^{\mathcal{H}}$) on the various expressions before (red) and after (green) optimizing $\theta_A \to \hat{\theta}_A$ (top) and $\theta_S \to \hat{\theta}_S$ (bottom) using the full rig while filtering the primary controls on each expression via $\gamma_1^{\mathcal{H}}$ and $\gamma_2^{\mathcal{H}}$.

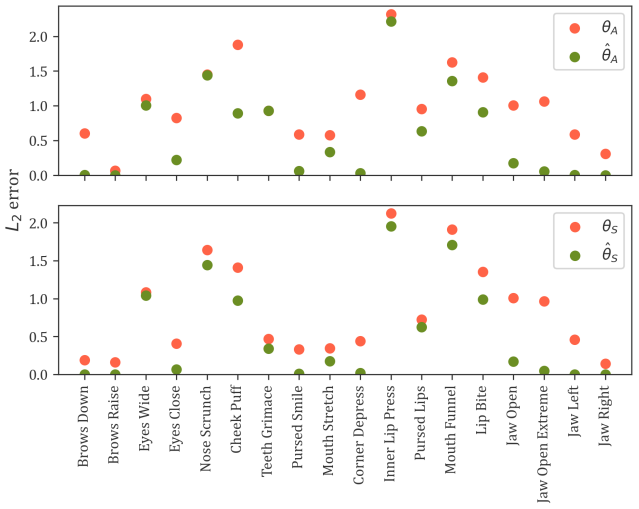

Fig. 13. L_2 errors (according to all $\gamma_1^{\mathcal{H}}$ terms) on the various expressions before (red) and after (green) optimizing $\theta_A \to \hat{\theta}_A$ (top) and $\theta_S \to \hat{\theta}_S$ (bottom) using the full rig while filtering the primary controls on each expression via $\gamma_1^{\mathcal{H}}$ and $\gamma_2^{\mathcal{H}}$ and additionally filtering various spurious controls via an additional $\gamma_1^{\mathcal{H}}$ term. Note that the minimization is still only considering the columns corresponding to the primary controls.

Figure 15 shows a summary of how the full tracker $\mathcal{T}$ improves over all stages. At this point, it is not possible to state the optimal strategy for improving the tracker, for example, it may be beneficial to optimize the columns corresponding to spurious controls first instead of last; however, these results do demonstrate that our approach is both effective and tractable. In particular, an optimal strategy would put more focus on animator preferences, e.g. they tend to prefer the minimization of spurious controls over maximal

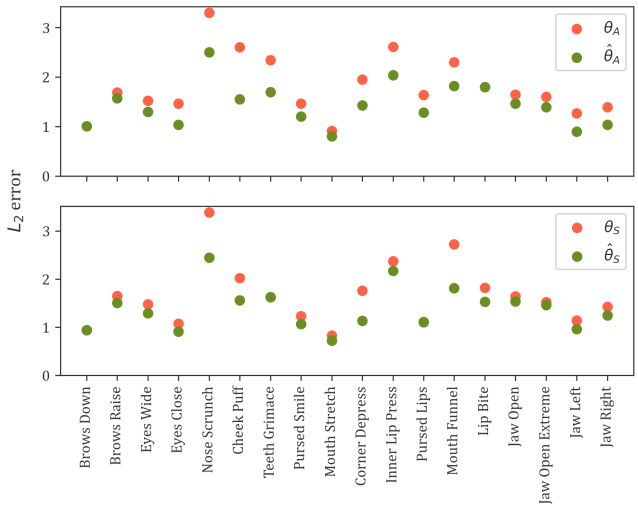

Fig. 14. L_2 errors (according to all γ_1 terms) on the various expressions before (red) and after (green) optimizing $\theta_A \to \hat{\theta}_A$ (top) and $\theta_S \to \hat{\theta}_S$ (bottom) using the full rig while filtering the primary controls on each expression via $\gamma_1^{\mathcal{H}}$ and $\gamma_2^{\mathcal{H}}$ and additionally filtering various spurious controls via an additional $\gamma_1^{\mathcal{H}}$ term. The difference between this figure and Figure 13 is that the minimization is now considering the columns of the spurious controls while freezing the columns of the primary controls.

usage of primary controls; therefore, a proposed so-called "optimal" strategy is best left for future work.

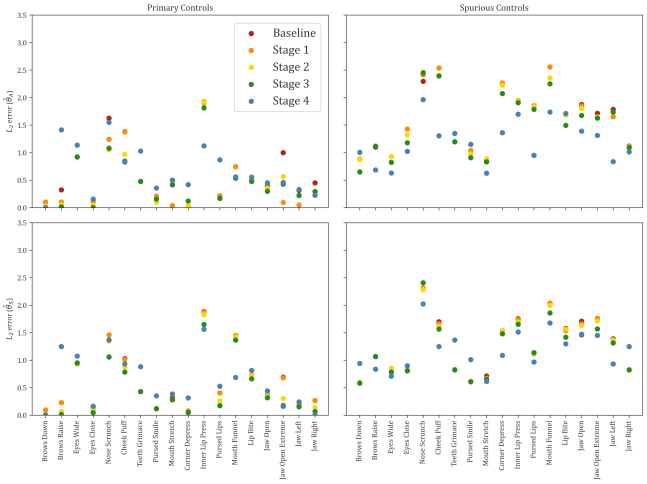

Fig. 15. Summary of the improvement in the L_2 errors (according to γ_1) throughout all four stages of the process (see Figures 11 to 14). The improvement in the primary controls is shown to the left, and the improvement in the spurious controls is shown to the right. Optimizing $\theta_A \to \hat{\theta}_A$ is shown on the top, and optimizing $\theta_S \to \hat{\theta}_S$ is shown on the bottom.

11.1 Validation With Synthetic Data

First, starting with the hand-crafted animation rig parameters θ_{GT} discussed in Section 8.3, we again use the Simon-Says expression set to generate ground-truth geometry for the tracker. As usual, the tracker is not able to properly invert the rig to successfully obtain the controls used to create the geometry. The errors are shown in the first column of Table 10. Using the optimization to improve $\theta_{GT} \rightarrow \hat{\theta}_{GT}$ can improve the results, as shown in the second column of Table 10. Both the morphed rig and the Simon-Says rig can also be improved via optimizing $\theta_M \rightarrow \hat{\theta}_M$ and $\theta_S \rightarrow \hat{\theta}_S$, respectively. Notably, Table 10 shows that optimizing the morphed rig via Simon-Says $\theta_M \rightarrow \theta_S$ and subsequently improving $\theta_S \rightarrow \hat{\theta}_S$ via tracker optimization results in a $\hat{\theta}_S$ that can give lower errors than θ_{GT} on unseen expressions.

Expression	θ_{GT}	$\hat{\theta}_{GT}$	θ_M	$\hat{\theta}_M$	θ_S	$\hat{\theta}_S$
Funnel Jaw Open	0.0260	0.0230	0.0364	0.0351	0.0268	0.0241
Asymmetric Funneler	0.0203	0.0179	0.0304	0.0269	0.0226	0.0189
Happy Speaking	0.0215	0.0191	0.0400	0.0328	0.0256	0.0202
"Dee"	0.0229	0.0206	0.0352	0.0344	0.0252	0.0228

Table 10. Average error in the controls for synthetic expressions that were not included in the Simon-Says expression set when optimizing rig parameters for the tracker.

12 BLACK BOX TRACKER

In this section, we consider a black-box tracker where one cannot make use of $\mathcal{T}_D$. Once again, we aim to improve the semantics of the tracker by modifying $\theta_A \rightarrow \hat{\theta}_A$ and $\theta_S \rightarrow \hat{\theta}_S$; in addition, all the trackers use the same reconstructed geometry from Titan. Figures 16 and 17 show the effect of modifying $\theta_A \rightarrow \hat{\theta}_A$ and $\theta_S \rightarrow \hat{\theta}_S$ on the ability of the tracker to invert the rig and match the geometry. Although the first stage from Section 11 cannot be used to bootstrap the process, since one cannot make use of $\mathcal{T}_D$, the other three stages are straightforward to implement. The results of the optimization are shown in Figures 18, 19, 20, and a summary of how the full tracker $\mathcal{T}$ improves over all stages is shown in Figure 21.

Fig. 16. A 573 frame pangram was used to test the tracker's ability to invert the rig and match the reconstructed geometry. The tracker was mostly able to adequately minimize geometry errors using any of θ_A, $\hat{\theta}_A$, θ_S, $\hat{\theta}_S$, even though the output controls (and thus the semantic interpretation) can vary significantly. See also Figure 17.

When considering a black-box tracker, it is important to note that a θ_S determined via Simon-Says may not work as well as the default θ_A because the internal parameters ψ are often overfit to certain properties assumed to be intrinsic to θ_A. Although the straightforward remedy would be to re-optimize the internal tracker parameters ψ via Equation 2 aiming to increase the efficacy of θ_S, this may not be possible with a black-box tracker. This means that our ability to optimize $\theta_S \rightarrow \hat{\theta}_S$ to obtain a viable $\hat{\theta}_S$ without re-optimizing the internal tracker parameters ψ can be seen as an additional contribution from our approach. That is, our approach facilitates the use of custom rigs and black-box trackers.

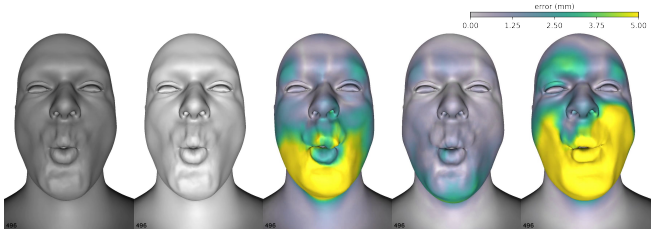

Fig. 17. Ground-truth reconstructed geometry (left) as compared to the geometry output by the tracker using θ_A, $\hat{\theta}_A$, θ_S, $\hat{\theta}_S$, respectively. This corresponds to frame 100, which was chosen because it has relatively large errors, of Figure 16. Colored regions indicate reconstruction error according to the colorbar.

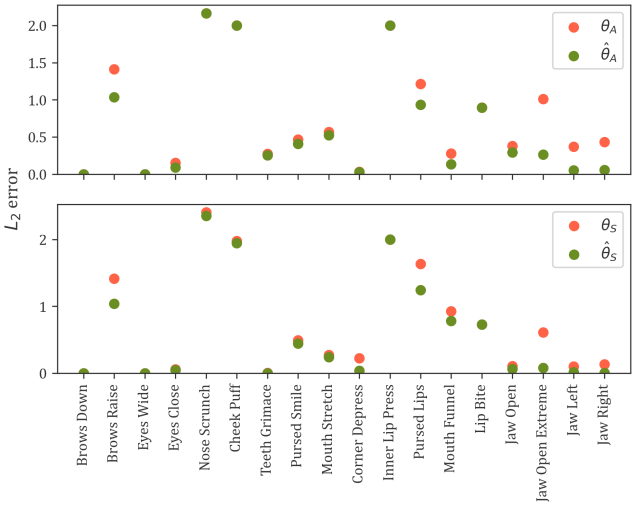

Fig. 18. L_2 errors (according to $\gamma_1^{\mathcal{H}}$) on the various expressions before (red) and after (green) optimizing $\theta_A \rightarrow \hat{\theta}_A$ (top) and $\theta_S \rightarrow \hat{\theta}_S$ (bottom) using the full rig while filtering the primary controls on each expression via $\gamma_1^{\mathcal{H}}$ and $\gamma_2^{\mathcal{H}}$.

Perhaps one of the most important things to keep in mind when perturbing $\theta_A \rightarrow \hat{\theta}_A$ or $\theta_S \rightarrow \hat{\theta}_S$ without the ability to re-optimize ψ via Equation 2 or even to understand how ψ was optimized is that sensitive threshold conditions may have been used. For example, a lip sealing constraint may activate when the lips are close enough together but otherwise do nothing. This means that a small perturbation of the rig could cause lips that were previously sealed via this constraint to instead be noticeably open even though their actual position prior to the constraint activation only changes by a small amount.

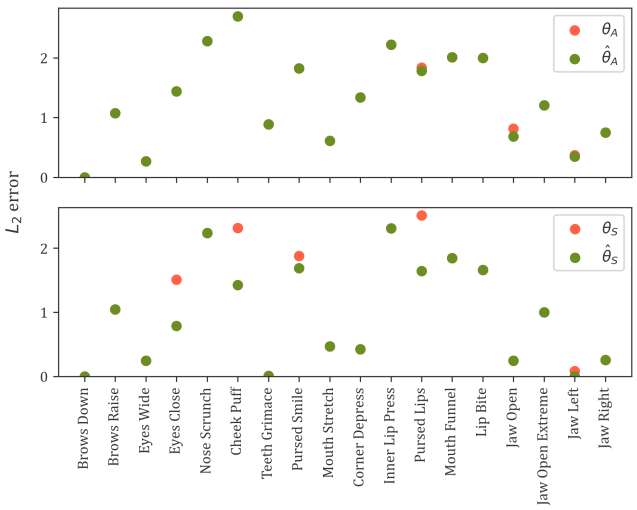

Fig. 19. L_2 errors (according to all $\gamma_1^{\mathcal{H}}$ terms) on the various expressions before (red) and after (green) optimizing $\theta_A \to \hat{\theta}_A$ (top) and $\theta_S \to \hat{\theta}_S$ (bottom) using the full rig while filtering the primary controls on each expression via $\gamma_1^{\mathcal{H}}$ and $\gamma_2^{\mathcal{H}}$ and additionally filtering various spurious controls via an additional $\gamma_1^{\mathcal{H}}$ term. Note that the minimization is still only considering the columns corresponding to the primary controls.

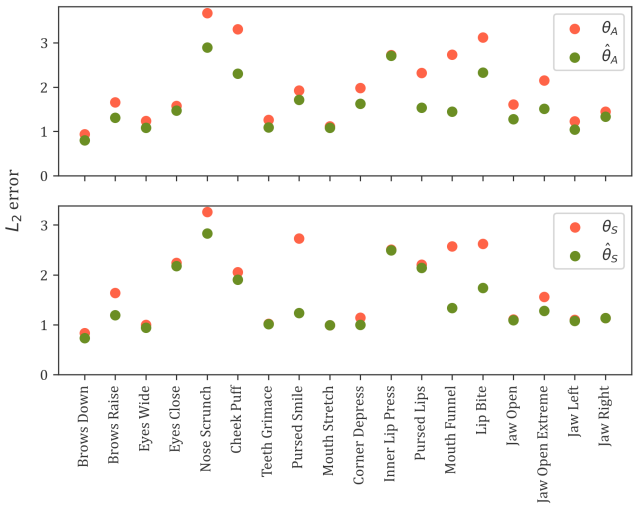

Fig. 20. L_2 errors (according to all γ_1 terms) on the various expressions before (red) and after (green) optimizing $\theta_A \to \hat{\theta}_A$ (top) and $\theta_S \to \hat{\theta}_S$ (bottom) using the full rig while filtering the primary controls on each expression via $\gamma_1^{\mathcal{H}}$ and $\gamma_2^{\mathcal{H}}$ and additionally filtering various spurious controls via an additional $\gamma_1^{\mathcal{H}}$ term. The difference between this figure and Figure 19 is that the minimization is now considering the columns of the spurious controls while freezing the columns of the primary controls.

13 RETARGETING

Having shown that the optimization framework presented in Section 10 can be used to optimize the default MetaHuman rig parameters θ_A or Simon-Says (see Section 8) rig parameters θ_S used by an open-source (see Section 11) or closed-source (see Section 12) tracker and that the optimized results still typically reconstruct reasonable

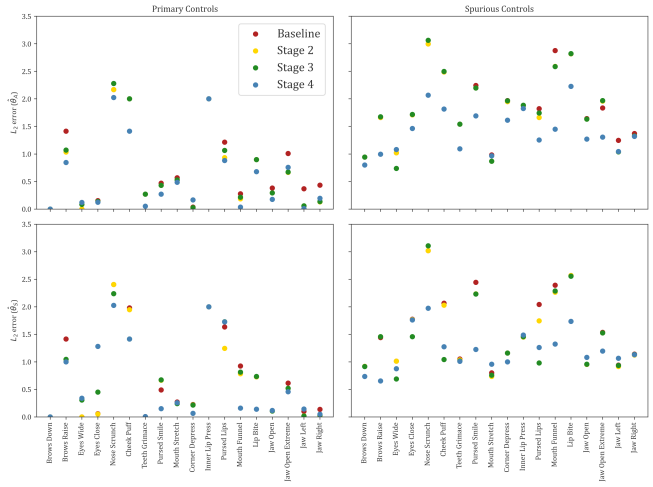

Fig. 21. Summary of the improvement in the L_2 errors (according to γ_1) throughout all three stages (there is no first stage on a closed-source tracker) of the process (see Figures 18 to 20). The improvement in the primary controls is shown to the left, and the improvement in the spurious controls is shown to the right. Optimizing $\theta_A \to \hat{\theta}_A$ is shown on the top, and optimizing $\theta_S \to \hat{\theta}_S$ is shown on the bottom.

geometry when inverting the rig (see Figures 9, 10, 16, 17), this section demonstrates the comparable efficacy of using θ_A, $\hat{\theta}_A$, θ_S, and $\hat{\theta}_S$ in the tracker in the context of retargeting where differing rig controls can produce noticeably different results. In order to do this, we choose a target subject, reconstruct their neutral geometry, fit it with an animation rig, and calibrate that rig via Simon-Says as discussed in Section 8. For the sake of evaluation, we show images and geometric reconstructions of both the performer and the target making semantically equivalent but geometrically different expressions. As a baseline (i.e. before any of our improvements), we show the results obtained using the default MetaHuman rig (i.e. θ_A) in the tracker mapped to the default MetaHuman rig (θ_A instead of θ_S) for the target. The results shown in this section were all chosen from the pangram used in Figures 9, 10, 16, and 17 with data that remained unseen in the Simon-Says calibration (see Section 8) and tracker optimization (see Sections 11 and 12) making it quite useful for predicting generalization.

Figures 23, 24, 25 show some representative results. As expected, the performer geometry (top rows) remains similar regardless of whether θ_A, $\hat{\theta}_A$, θ_S, or $\hat{\theta}_S$ was used. In fact, as compared to the so-called reconstruction (as can be seen by the vertex color shading), the geometry can sometimes get worse when modifying $\theta_A \to \hat{\theta}_A$, $\theta_S \to \hat{\theta}_S$, or even $\theta_A \to \theta_S$; however, these differences are unimportant, since the focus is on obtaining good results on the target. As far as the target is concerned (bottom rows), modifying $\theta_A \to \hat{\theta}_A$, $\theta_S \to \hat{\theta}_S$, and $\theta_A \to \theta_S$ all improve the results, illustrating the efficacy of our approach. In general, key poses in performance or speech are better explained by relevant primary controls and less explained by undesired tweaker controls by using our approach, leading to cleaner and more accurate expressions.

14 PUPPETEERING

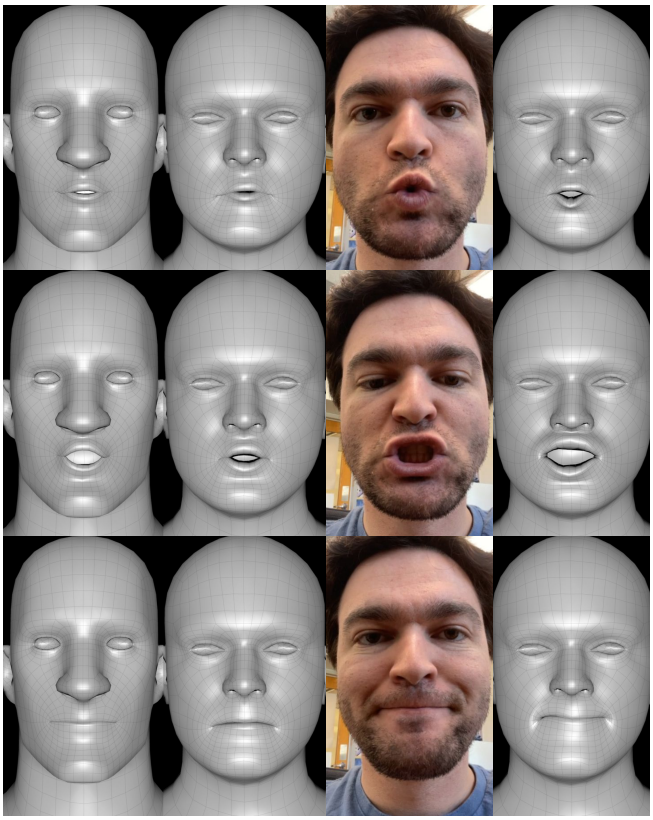

Fig. 22. Simon-Says process on specific phoneme-level controls ("OO", "CH", "M/B/P") unique to the game/VR character rig (one on each row). Left to right: Game/VR character rig, morphed rig, performer's expression, Simon-Says results. Note that the first two columns provide very little usable input for the performer, so they need to rely more heavily on the description of each desired expression.

Whereas Section 13 retargeted to another human, here we consider retargeting to a synthetically generated humanoid character. Various rig frameworks, especially those for game/VR characters, will differ significantly in their semantic control space. Given the difficulties associated with remapping controls between two disparate rigs, it seems prudent to utilize the game/VR character rig on the performer as well. Thus, we create a game/VR character representation for the performer. This is accomplished by first creating a correspondence between the game/VR character's triangle mesh and the performer's triangle mesh in order to assign a game/VR character triangle mesh to the performer's neutral geometry. Note that the teeth were ignored, since they were not used in the tracking of the performer. Then, the game/VR character rig was volumetrically morphed to the performer's new game/VR character consistent neutral geometry. Next, the morphed rig was calibrated via Simon-Says. It is inappropriate to use the same Simon-Says expression set as was used for person-to-person retargeting. Instead, we designed a set of basis expressions that better-covered the rig controls of the game/VR character. In total, fifteen expressions

- neutral, eyes close, eyes wide, brows raise, brows down, teeth grimace, pursed smile, corner depress, jaw open, jaw left, jaw right
- "OO" phoneme, "CH" phoneme, "M/B/P" phoneme, "F/V" phoneme

were used (including the four phoneme-level expressions). As shown in Figure 22, the low expressivity of the game/VR character rig meant that the Simon-Says results did not well match the characteristics of the performer; nevertheless, the vast improvement of the basis shapes led to more accurate semantic interpretation in our retargeting experiments.

Since the closed-source MetaHuman Animator tracker is heavily designed to work specifically with the MetaHuman rig framework, our experiments only use our custom open-source tracker (see Section 11). In addition, unlike the examples shown in Section 13, there is no convenient off-the-shelf solver that can be used as a baseline for comparison. Although some of the prior works discussed in Section 2 could potentially be utilized, it would not be straightforward to implement them on this example. The fact that it is straightforward to apply our methodology to the puppeteering of a highly crafted game/VR character rig emphasizes the efficacy of our approach. For results on a pangram, see Figures 26 through 30. Although we did not expect the puppet rig to be useful without Simon-Says calibration, we show θ_P and $\hat{\theta}_P$ results as a baseline for comparison.

15 CONCLUSION

Unfortunately, too many would-be trackers produce animation curves that require subsequent significant cleanup effort by artists in order to obtain results that can be used effectively for retargeting. Our mathematical analysis illustrated that a tracker can be thought of as geometric reconstruction followed by rig inversion highlighting the fact that those who merely stress the efficacy of their geometric reconstructions are missing a key component of the problem. Moreover, we showed that it can even be beneficial to relax the accuracy of the geometric reconstruction as a tradeoff for obtaining better and more usable animation curves via the rig inversion.

Although we did build an open source tracker in order to test our hypotheses, our real goal is to improve off-the-shelf closed source trackers specialized for use in industry. In the latter case, one does not necessarily have access to the internal tracker parameters or have the ability to optimize them or to rewrite the tracker to be differentiable (even if one could modify such a tracker, it may no longer work as desired). However, all of these trackers do accommodate variations in rig parameters, since rig parameters (and geometry) vary across individuals and characters. Thus, we set out to improve the animation curve output of the tracker by modifying the accessible rig parameters. In order to do this without access to the internal workings of the (closed source) tracker itself, we leveraged the implicit function theorem and a reformulation of Broyden's method in order to implicitly differentiate the closed source tracker with respect to the parameters of the animation rig. Our results showed that the derivative estimates were valid enough to minimize the

various objective functions one might consider when aiming to improve the animation curve output of the tracker.

Compared to the baseline results obtained without using our approach, we showed that the rig parameters could be modified (either $\theta_A \rightarrow \hat{\theta}_A$ in Section 13 or $\theta_P \rightarrow \hat{\theta}_P$ in Section 14) in order to obtain improved results when retargeting from one individual to another (in Section 13) or when utilizing performance capture for puppeteering (in Section 14). It is important to stress that both of these results are also just baselines for the sake of comparison, since we strongly advocate using personalized animation rigs in order to capture the varying motion signatures particular to each individual. In order to do this, we utilize a Simon-Says calibration process in order to personalize the performer's animation rig (obtaining θ_S) before modifying it (i.e. $\theta_S \rightarrow \hat{\theta}_S$) via our implicit differentiation approach. The results obtained using $\hat{\theta}_S$ for the rig parameters seen by the tracker represent our proposed method. Note that a personalized animation rig is also preferable for the target in person-to-person retargeting and that the animation rig for a character (i.e. θ_P) is already highly personalized by design.

We obviously did not exhaust all possible approaches for improving rig parameters, nor did we incorporate an exhaustive list of all possible hand-tuned constraints on rig parameters and rig motion. This can be seen as either limitations or future work. That is, although we show the efficacy of our approach, we did not build the ultimate rig or the ultimate tracker. Given the high level of expertise of those who work in this field, we felt that providing an additional tool was more important than providing some sort of claim about an ultimate solution. One interesting avenue for future work consists of generalizing the concepts presented in this paper to a wider set of approaches. For example, consider decomposing the latent code obtained via GAN inversion of a specific individual into two components, where a parallel can be drawn between one component and a triangulated surface representation of the neutral pose and a second parallel can be drawn between the other component and animation rig parameters.

ACKNOWLEDGMENTS

Research supported in part by ONR N00014-24-1-2644, ONR N00014-21-1-2771, and ONR N00014-19-1-2285. We would like to acknowledge both SONY and Epic Games for additional support.

REFERENCES

Deepali Aneja, Bindita Chaudhuri, Alex Colburn, Gary Faigin, Linda Shapiro, and Barbara Mones. 2018. Learning to Generate 3D Stylized Character Expressions from Humans. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. 160–169. https://doi.org/10.1109/WACV.2018.00024

ShahRukh Athar, Shunsuke Saito, Zhengyu Yang, Stanislav Pidhorskyi, and Chen Cao. 2024. Bridging the Gap: Studio-Like Avatar Creation from a Monocular Phone Capture. In *Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29–October 4, 2024, Proceedings, Part XII* (Milan, Italy). Springer-Verlag, Berlin, Heidelberg, 72–88. https://doi.org/10.1007/978-3-031-73254-6_5

Autodesk, Inc. 2025. *Maya.* https://www.autodesk.com/products/maya

Kelian Baert, Shrisha Bharadwaj, Fabien Castan, Benoit Maujean, Marc Christie, Victoria Abrevaya, and Adnane Boukhayma. 2024. SPARK: Self-supervised Personalized Real-time Monocular Face Capture. (2024). https://doi.org/10.1145/3680528.3687704

Shaojie Bai, Te-Li Wang, Chenghui Li, Akshay Venkatesh, Tomas Simon, Chen Cao, Gabriel Schwartz, Jason Saragih, Yaser Sheikh, and Shih-En Wei. 2024. Universal Facial Encoding of Codec Avatars from VR Headsets. *ACM Trans. Graph.* 43, 4, Article 93 (jul 2024), 22 pages.

Stephen W. Bailey, Dalton Omens, Paul Dilorenzo, and James F. O'Brien. 2020. Fast and Deep Facial Deformations. *ACM Transactions on Graphics* 39, 4 (Aug. 2020), 94:1–15. https://doi.org/10.1145/3386569.3392397 Presented at SIGGRAPH 2020, Washington D.C..

Michael Bao, Matthew Cong, Stéphane Grabli, and Ronald Fedkiw. 2019. High-Quality Face Capture Using Anatomical Muscles. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 10794–10803. https://doi.org/10.1109/CVPR.2019.01106

Thabo Beeler and Derek Bradley. 2014. Rigid stabilization of facial expressions. *ACM Trans. Graph.* 33, 4, Article 44 (jul 2014), 9 pages.

Shrisha Bharadwaj, Yufeng Zheng, Otmar Hilliges, Michael J. Black, and Victoria Fernandez Abrevaya. 2023. FLARE: Fast learning of Animatable and Relightable Mesh Avatars. *ACM Transactions on Graphics* 42 (Dec. 2023), 15. https://doi.org/10.1145/3618401

Volker Blanz and Thomas Vetter. 1999. A Morphable Model For The Synthesis Of 3D Faces. In *SIGGRAPH*.

Blender Online Community. 2025. *Blender.* https://www.blender.org/

C. G. Broyden. 1965. A Class of Methods for Solving Nonlinear Simultaneous Equations. *Math. Comp.* 19 (1965), 577–593.

Marcel C. Buehler, Gengyan Li, Erroll Wood, Leonhard Helminger, Xu Chen, Tanmay Shah, Daoye Wang, Stephan Garbin, Sergio Orts-Escolano, Otmar Hilliges, Dmitry Lagun, Jérémy Riviere, Paulo Gotardo, Thabo Beeler, Abhimitra Meka, and Kripasindhu Sarkar. 2024. Cafca: High-quality Novel View Synthesis of Expressive Faces from Casual Few-shot Captures. In *ACM SIGGRAPH Asia 2024 Conference Paper*. https://doi.org/10.1145/3680528.3687580

Marcel C. Buehler, Kripasindhu Sarkar, Tanmay Shah, Gengyan Li, Daoye Wang, Leonhard Helminger, Sergio Orts-Escolano, Dmitry Lagun, Otmar Hilliges, Thabo Beeler, and Abhimitra Meka. 2023. Preface: A Data-driven Volumetric Prior for Few-shot Ultra High-resolution Face Synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*.

Chen Cao, Tomas Simon, Jin Kyu Kim, Gabe Schwartz, Michael Zollhoefer, Shun-Suke Saito, Stephen Lombardi, Shih-En Wei, Danielle Belko, Shoou-I Yu, Yaser Sheikh, and Jason Saragih. 2022. Authentic volumetric avatars from a phone scan. *ACM Trans. Graph.* 41, 4, Article 163 (jul 2022), 19 pages.

Sihun Cha, Serin Yoon, Kwanggyoon Seo, and Junyong Noh. 2025. Neural Face Skinning for Mesh-agnostic Facial Expression Cloning. *Computer Graphics Forum* n/a, n/a (2025), e70009. https://doi.org/10.1111/cgf.70009

Prashanth Chandran, Loïc Ciccone, Markus Gross, and Derek Bradley. 2022. Local anatomically-constrained facial performance retargeting. *ACM Trans. Graph.* 41, 4, Article 168 (jul 2022), 14 pages.

Byungkuk Choi, Haekwang Eom, Benjamin Mouscadet, Stephen Cullingford, Kurt Ma, Stefanie Gassel, Suzi Kim, Andrew Moffat, Millicent Maier, Marco Revelant, Joe Letteri, and Karan Singh. 2022. Animatomy: an Animator-centric, Anatomically Inspired System for 3D Facial Modeling, Animation and Transfer. In *SIGGRAPH Asia 2022 Conference Papers* (Daegu, Republic of Korea) *(SA '22)*. Association for Computing Machinery, New York, NY, USA, Article 16, 9 pages.

Yeonsoo Choi, Inyup Lee, Sihun Cha, Seonghyeon Kim, Sunjin Jung, and Junyong Noh. 2025. Deep-Learning-Based Facial Retargeting Using Local Patches. *Computer Graphics Forum* 44, 1 (2025), e15263. https://doi.org/10.1111/cgf.15263

Matthew Cong, Michael Bao, Jane L. E, Kiran S. Bhat, and Ronald Fedkiw. 2015. Fully automatic generation of anatomical face simulation models. In *Proceedings of the 14th ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (Los Angeles, California) *(SCA '15)*. Association for Computing Machinery, New York, NY, USA, 175–183.

Radek Danecek, Michael J. Black, and Timo Bolkart. 2022. EMOCA: Emotion Driven Monocular Face Capture and Animation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 20311–20322.

William C. Davidon. 1991. Variable Metric Method for Minimization. *SIAM Journal on Optimization* 1, 1 (1991), 1–17. https://doi.org/10.1137/0801001

Paul Debevec, Tim Hawkins, Chris Tchou, Haarm-Pieter Duiker, Westley Sarokin, and Mark Sagar. 2000. Acquiring the reflectance field of a human face. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00)*. ACM Press/Addison-Wesley Publishing Co., USA, 145–156.

Yu Deng, Jiaolong Yang, Sicheng Xu, Dong Chen, Yunde Jia, and Xin Tong. 2019. Accurate 3D Face Reconstruction With Weakly-Supervised Learning: From Single Image to Image Set . In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE Computer Society, Los Alamitos, CA, USA, 285–295. https://doi.org/10.1109/CVPRW.2019.00038

Zhigang Deng, Pei-Ying Chiang, Pamela Fox, and Ulrich Neumann. 2006. Animating blendshape faces by cross-mapping motion capture data. In *Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games* (Redwood City, California) *(I3D '06)*. Association for Computing Machinery, New York, NY, USA, 43–48. https://doi.org/10.1145/1111411.1111419

Abdallah Dib, Junghyun Ahn, Cédric Thébault, Philippe-Henri Gosselin, and Louis Chevallier. 2023. S2F2: Self-Supervised High Fidelity Face Reconstruction from Monocular Image. In *2023 IEEE 17th International Conference on Automatic Face and Gesture Recognition (FG)* (Waikoloa Beach, HI, USA). IEEE Press, 1–8. https://doi.org/10.1109/FG57933.2023.10042713

A. Dib, G. Bharaj, J. Ahn, C. Thébault, P. Gosselin, M. Romeo, and L. Chevallier. 2021a. Practical Face Reconstruction via Differentiable Ray Tracing. *Computer Graphics Forum* 40, 2 (2021), 153–164. https://doi.org/10.1111/cgf.142622

Abdallah Dib, Luiz Gustavo Hafemann, Emeline Got, Trevor Anderson, Amin Fadaeinejad, Rafael M. O. Cruz, and Marc-André Carbonneau. 2024. MoSAR: Monocular Semi-Supervised Model for Avatar Reconstruction using Differentiable Shading. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 1770–1780. https://doi.org/10.1109/CVPR52733.2024.00174

Abdallah Dib, Cedric Thebault, Junghyun Ahn, Philippe-Henri Gosselin, Christian Theobalt, and Louis Chevallier. 2021b. Towards High Fidelity Monocular Face Reconstruction with Rich Reflectance using Self-supervised Learning and Ray Tracing . In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE Computer Society, Los Alamitos, CA, USA, 12799–12809. https://doi.org/10.1109/ICCV48922.2021.01258

Carl Doersch, Yi Yang, Mel Vecerik, Dilara Gokay, Ankush Gupta, Yusuf Aytar, Joao Carreira, and Andrew Zisserman. 2023. TAPIR: Tracking any point with per-frame initialization and temporal refinement. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 10061–10072.

Hao-Bin Duan, Miao Wang, Jin-Chuan Shi, Xu-Chuan Chen, and Yan-Pei Cao. 2023. BakedAvatar: Baking Neural Fields for Real-Time Head Avatar Synthesis. *ACM Trans. Graph.* 42, 6, Article 225 (sep 2023), 14 pages. https://doi.org/10.1145/3618399

Bernhard Egger, William A. P. Smith, Ayush Tewari, Stefanie Wuhrer, Michael Zollhoefer, Thabo Beeler, Florian Bernard, Timo Bolkart, Adam Kortylewski, Sami Romdhani, Christian Theobalt, Volker Blanz, and Thomas Vetter. 2020. 3D Morphable Face Models—Past, Present, and Future. *ACM Trans. Graph.* 39, 5, Article 157 (jun 2020), 38 pages.

Epic Games. 2025. *MetaHuman Animator*. https://www.unrealengine.com/en-US/digital-humans

R. Fletcher. 1987. *Practical Methods of Optimization*. John Wiley and Sons.

R. Fletcher and M. J. D. Powell. 1963. A Rapidly Convergent Descent Method for Minimization. *Comput. J.* 6, 2 (08 1963), 163–168. https://doi.org/10.1093/comjnl/6.2.163

Guy Gafni, Justus Thies, Michael Zollhöfer, and Matthias Nießner. 2021. Dynamic Neural Radiance Fields for Monocular 4D Facial Avatar Reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 8649–8658.

Pablo Garrido, Levi Valgaert, Chenglei Wu, and Christian Theobalt. 2013. Reconstructing detailed dynamic face geometry from monocular video. *ACM Trans. Graph.* 32, 6, Article 158 (nov 2013), 10 pages.

Charles William Gear. 1971. *Numerical initial value problems in ordinary differential equations*. Longman Higher Education, Harlow, England.

Abhijeet Ghosh, Graham Fyffe, Borom Tunwattanapong, Jay Busch, Xueming Yu, and Paul Debevec. 2011. Multiview face capture using polarized spherical gradient illumination. *ACM Trans. Graph.* 30, 6 (Dec. 2011), 1–10. https://doi.org/10.1145/2070781.2024163

Philip-William Grassal, Malte Prinzler, Titus Leistner, Carsten Rother, Matthias Nießner, and Justus Thies. 2021. Neural Head Avatars from Monocular RGB Videos. *arXiv preprint arXiv:2112.01554* (2021).

Yuxuan Han, Junfeng Lyu, and Feng Xu. 2024. High-Quality Facial Geometry and Appearance Capture at Home. *CVPR*.

Charlie Hewitt, Fatemeh Saleh, Sadegh Aliakbarian, Lohit Petikam, Shideh Rezaeifar, Louis Florentin, Zafiirah Hosenie, Thomas J. Cashman, Julien Valentin, Darren Cosker, and Tadas Baltrusaitis. 2024. Look Ma, no markers: holistic performance capture without the hassle. *ACM Trans. Graph.* 43, 6, Article 235 (Nov. 2024), 12 pages. https://doi.org/10.1145/3687772

Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. 2023. CoTracker: It is Better to Track Together.

Ladislav Kavan, John Doublestein, Martin Prazak, Matthew Cioffi, and Doug Roble. 2024. Compressed Skinning for Facial Blendshapes. In *ACM SIGGRAPH 2024 Conference Papers* (Denver, CO, USA) *(SIGGRAPH '24)*. Association for Computing Machinery, New York, NY, USA, Article 47, 9 pages. https://doi.org/10.1145/3641519.3657477

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Trans. Graph.* 42, 4, Article 139 (July 2023), 14 pages. https://doi.org/10.1145/3592433

Seonghyeon Kim, Sunjin Jung, Kwanggyoon Seo, Roger Blanco i Ribera, and Junyong Noh. 2021. Deep Learning-Based Unsupervised Human Facial Retargeting. *Computer Graphics Forum* 40, 7 (2021), 45–55.

Tobias Kirschstein, Shenhan Qian, Simon Giebenhain, Tim Walter, and Matthias Nießner. 2023. NeRSemble: Multi-View Radiance Field Reconstruction of Human Heads. *ACM Trans. Graph.* 42, 4, Article 161 (jul 2023), 14 pages. https://doi.org/10.1145/3592455

Samuli Laine, Tero Karras, Timo Aila, Antti Herva, Shunsuke Saito, Ronald Yu, Hao Li, and Jaakko Lehtinen. 2017. Production-level facial performance capture using deep convolutional neural networks. In *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (Los Angeles, California) *(SCA '17)*. Association for Computing Machinery, New York, NY, USA, Article 10, 10 pages.

Biwen Lei, Jianqiang Ren, Mengyang Feng, Miaomiao Cui, and Xuansong Xie. 2023. A Hierarchical Representation Network for Accurate and Detailed Face Reconstruction from In-The-Wild Images . In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, Los Alamitos, CA, USA, 394–403. https://doi.org/10.1109/CVPR52729.2023.00046

Hao Li, Bart Adams, Leonidas J. Guibas, and Mark Pauly. 2009. Robust single-view geometry and motion reconstruction. *ACM Trans. Graph.* 28, 5 (dec 2009), 1–10.

Hao Li, Thibaut Weise, and Mark Pauly. 2010. Example-based facial rigging. *ACM Trans. Graph.* 29, 4, Article 32 (jul 2010), 6 pages.

Tianye Li, Timo Bolkart, Michael. J. Black, Hao Li, and Javier Romero. 2017. Learning a model of facial shape and expression from 4D scans. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)* 36, 6 (2017), 194:1–194:17. https://doi.org/10.1145/3130800.3130813

Winnie Lin, Yilin Zhu, Demi Guo, and Ron Fedkiw. 2022. Leveraging Deepfakes to Close the Domain Gap between Real and Synthetic Images in Facial Capture Pipelines. arXiv:2204.10746 [cs.CV] https://arxiv.org/abs/2204.10746

Dong C. Liu and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical Programming* 45, 1 (08 1989), 503–528. https://doi.org/10.1007/BF01589116

Stephen Lombardi, Jason Saragih, Tomas Simon, and Yaser Sheikh. 2018. Deep appearance models for face rendering. *ACM Trans. Graph.* 37, 4, Article 68 (jul 2018), 13 pages.

S. Ma, T. Simon, J. Saragih, D. Wang, Y. Li, F. La Torre, and Y. Sheikh. 2021. Pixel Codec Avatars. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, Los Alamitos, CA, USA, 64–73.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2021. NeRF: representing scenes as neural radiance fields for view synthesis. *Commun. ACM* 65, 1 (Dec. 2021), 99–106. https://doi.org/10.1145/3503250

Xin Ming, Jiawei Li, Jingwang Ling, Libo Zhang, and Feng Xu. 2024. High-Quality Mesh Blendshape Generation from Face Videos via Neural Inverse Rendering. In *Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29–October 4, 2024, Proceedings, Part LXX* (Milan, Italy). Springer-Verlag, Berlin, Heidelberg, 106–125. https://doi.org/10.1007/978-3-031-72897-6_7

Lucio Moser, Chinyu Chien, Mark Williams, Jose Serra, Darren Hendler, and Doug Roble. 2021. Semi-supervised video-driven facial animation transfer for production. *ACM Trans. Graph.* 40, 6, Article 222 (dec 2021), 18 pages.

Jorge Nocedal and Stephen J. Wright. 2006. *Numerical optimization*. Springer.

Jun-yong Noh and Ulrich Neumann. 2001. Expression cloning. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*. Association for Computing Machinery, New York, NY, USA, 277–288.

Verónica Orvalho, Pedro Bastos, Frederic Parke, Bruno Oliveira, and Xenxo Alvarez. 2012. A Facial Rigging Survey. In *Eurographics 2012 - State of the Art Reports*, Marie-Paule Cani and Fabio Ganovelli (Eds.). The Eurographics Association. https://doi.org//10.2312/conf/EG2012/stars/183-204

Ye Pan, Chang Liu, Sicheng Xu, Shuai Tan, and Jiaolong Yang. 2025. VASA-Rig: Audio-Driven 3D Facial Animation with 'Live' Mood Dynamics in Virtual Reality. *IEEE Transactions on Visualization and Computer Graphics* 31, 5 (2025), 2416–2425. https://doi.org/10.1109/TVCG.2025.3549168

Dafei Qin, Jun Saito, Noam Aigerman, Groueix Thibault, and Taku Komura. 2023. Neural Face Rigging for Animating and Retargeting Facial Meshes in the Wild. In *SIGGRAPH 2023 Conference Papers*.

Feng Qiu, Wei Zhang, Chen Liu, Rudong An, Lincheng Li, Yu Ding, Changjie Fan, Zhipeng Hu, and Xin Yu. 2024. FreeAvatar: Robust 3D Facial Animation Transfer by Learning an Expression Foundation Model. In *SIGGRAPH Asia 2024 Conference Papers* (Tokyo, Japan) *(SA '24)*. Association for Computing Machinery, New York, NY, USA, Article 42, 11 pages. https://doi.org/10.1145/3680528.3687669

Stevo Racković, Dušan Jakovetić, and Cláudia Soares. 2024. Refined Inverse Rigging: A Balanced Approach to High-fidelity Blendshape Animation. In *SIGGRAPH Asia 2024 Conference Papers* (Tokyo, Japan) *(SA '24)*. Association for Computing Machinery, New York, NY, USA, Article 45, 9 pages. https://doi.org/10.1145/3680528.3687670

Ravi Ramamoorthi and Pat Hanrahan. 2001. An efficient representation for irradiance environment maps. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*. Association for Computing Machinery, New York, NY, USA, 497–500.

Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. 2020. Accelerating 3D Deep Learning with PyTorch3D. *arXiv:2007.08501* (2020).

George Retsinas, Panagiotis P. Filntisis, Radek Danecek, Victoria F. Abrevaya, Anastasios Roussos, Timo Bolkart, and Petros Maragos. 2024. 3D Facial Expressions through Analysis-by-Neural-Synthesis. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Roger Blanco i Ribera, Eduard Zell, J. P. Lewis, Junyong Noh, and Mario Botsch. 2017. Facial retargeting with automatic range of motion alignment. *ACM Trans. Graph.* 36, 4, Article 154 (jul 2017), 12 pages.

Shunsuke Saito, Gabriel Schwartz, Tomas Simon, Junxuan Li, and Giljoo Nam. 2024. Relightable Gaussian Codec Avatars. In *CVPR*.

Shunsuke Saito, Lingyu Wei, Liwen Hu, Koki Nagano, and Hao Li. 2017. Photorealistic Facial Texture Inference Using Deep Neural Networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2326–2335. https://doi.org/10.1109/CVPR.2017.250

Kripasindhu Sarkar, Marcel C. Bühler, Gengyan Li, Daoye Wang, Delio Vicini, Jérémy Riviere, Yinda Zhang, Sergio Orts-Escolano, Paulo Gotardo, Thabo Beeler, and Abhimitra Meka. 2023. LitNeRF: Intrinsic Radiance Decomposition for High-Quality View Synthesis and Relighting of Faces. In *SIGGRAPH Asia 2023 Conference Papers* (Sydney, NSW, Australia) *(SA '23)*. Association for Computing Machinery, New York, NY, USA, Article 42, 11 pages. https://doi.org/10.1145/3610548.3618210

Yeongho Seol, J.P. Lewis, Jaewoo Seo, Byungkuk Choi, Ken Anjyo, and Junyong Noh. 2012. Spacetime expression cloning for blendshapes. *ACM Trans. Graph.* 31, 2, Article 14 (apr 2012), 12 pages.

Jaewon Song, Byungkuk Choi, Yeongho Seol, and Junyong Noh. 2011. Characteristic facial retargeting. *Computer Animation and Virtual Worlds* 22, 2-3 (2011), 187–194.

Robert W. Sumner and Jovan Popović. 2004. Deformation transfer for triangle meshes. *ACM Trans. Graph.* 23, 3 (aug 2004), 399–405.

Felix Taubner, Prashant Raina, Mathieu Tuli, Eu Wern Teh, Chul Lee, and Jinmiao Huang. 2024. 3D Face Tracking from 2D Video through Iterative Dense UV to Image Flow. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 1227–1237.

Kartik Teotia, Mallikarjun B R, Xingang Pan, Hyeongwoo Kim, Pablo Garrido, Mohamed Elgharib, and Christian Theobalt. 2024. HQ3DAvatar: High-quality Implicit 3D Head Avatar. *ACM Trans. Graph.* 43, 3, Article 27 (apr 2024), 24 pages.

Ayush Tewari, Michael Zollhöfer, Hyeongwoo Kim, Pablo Garrido, Florian Bernard, Patrick Pérez, and Christian Theobalt. 2017. MoFA: Model-Based Deep Convolutional Face Autoencoder for Unsupervised Monocular Reconstruction. In *2017 IEEE International Conference on Computer Vision (ICCV)*. 3735–3744. https://doi.org/10.1109/ICCV.2017.401

Phong Tran, Egor Zakharov, Long-Nhat Ho, Adilbek Karmanov, Ariana Bermudez Venegas, McLean Goldwhite, Aviral Agarwal, Liwen Hu, Anh Tran, and Hao Li. 2024. VOODOO XP: Expressive One-Shot Head Reenactment for VR Telepresence. *ACM Trans. Graph.* 43, 6, Article 253 (Nov. 2024), 26 pages. https://doi.org/10.1145/3687974

Alex Trevithick, Matthew Chan, Michael Stengel, Eric Chan, Chao Liu, Zhiding Yu, Sameh Khamis, Manmohan Chandraker, Ravi Ramamoorthi, and Koki Nagano. 2023. Real-Time Radiance Fields for Single-Image Portrait View Synthesis. *ACM Trans. Graph.* 42, 4, Article 135 (July 2023), 15 pages. https://doi.org/10.1145/3592460

Yang Wang, Xiaolei Huang, Chan-Su Lee, Song Zhang, Zhiguo Li, Dimitris Samaras, Dimitris Metaxas, Ahmed Elgammal, and Peisen Huang. 2004. High Resolution Acquisition, Learning and Transfer of Dynamic 3-D Facial Expressions. *Computer Graphics Forum* (2004). https://doi.org/10.1111/j.1467-8659.2004.00800.x

Zidu Wang, Xiangyu Zhu, Tianshuo Zhang, Baiqin Wang, and Zhen Lei. 2024. 3D Face Reconstruction with the Geometric Guidance of Facial Part Segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1672–1682.

Erroll Wood, Tadas Baltrušaitis, Charlie Hewitt, Matthew Johnson, Jingjing Shen, Nikola Milosavljević, Daniel Wilde, Stephan Garbin, Toby Sharp, Ivan Stojiljković, et al. 2022. 3d face reconstruction with dense landmarks. In *European Conference on Computer Vision*. Springer, 160–177.

Chenglei Wu, Derek Bradley, Markus Gross, and Thabo Beeler. 2016. An anatomically-constrained local deformation model for monocular face capture. *ACM Trans. Graph.* 35, 4, Article 115 (jul 2016), 12 pages.

Sijing Wu, Yichao Yan, Yunhao Li, Yuhao Cheng, Wenhan Zhu, Ke Gao, Xiaobo Li, and Guangtao Zhai. 2023. GANHead: Towards Generative Animatable Neural Head Avatars. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 437–447.

Jun Xiang, Xuan Gao, Yudong Guo, and Juyong Zhang. 2024. FlashAvatar: High-fidelity Head Avatar with Efficient Gaussian Embedding. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Yuelang Xu, Benwang Chen, Zhe Li, Hongwen Zhang, Lizhen Wang, Zerong Zheng, and Yebin Liu. 2024. Gaussian Head Avatar: Ultra High-fidelity Head Avatar via Dynamic Gaussians. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Haotian Yang, Mingwu Zheng, Wanquan Feng, Haibin Huang, Yu-Kun Lai, Pengfei Wan, Zhongyuan Wang, and Chongyang Ma. 2023a. Towards Practical Capture of High-Fidelity Relightable Avatars. In *SIGGRAPH Asia 2023 Conference Proceedings*.

Haotian Yang, Mingwu Zheng, Chongyang Ma, Yu-Kun Lai, Pengfei Wan, and Haibin Huang. 2024. VRMM: A Volumetric Relightable Morphable Head Model. In *ACM SIGGRAPH 2024 Conference Papers* (Denver, CO, USA) *(SIGGRAPH '24)*. Association for Computing Machinery, New York, NY, USA, Article 46, 11 pages. https://doi.org/10.1145/3641519.3657406

Lingchen Yang, Gaspard Zoss, Prashanth Chandran, Paulo Gotardo, Markus Gross, Barbara Solenthaler, Eftychios Sifakis, and Derek Bradley. 2023b. An Implicit Physical Face Model Driven by Expression and Style. In *SIGGRAPH Asia 2023 Conference Papers* (Sydney, NSW, Australia) *(SA '23)*. Association for Computing Machinery, New York, NY, USA, Article 106, 12 pages.

Juyong Zhang, Keyu Chen, and Jianmin Zheng. 2022. Facial Expression Retargeting From Human to Avatar Made Easy. *IEEE Transactions on Visualization and Computer Graphics* 28, 2 (2022), 1274–1287. https://doi.org/10.1109/TVCG.2020.3013876

Li Zhang, Noah Snavely, Brian Curless, and Steven M. Seitz. 2004. Spacetime faces: high resolution capture for modeling and animation. *ACM Trans. Graph.* 23, 3 (aug 2004), 548–558.

T. Zhang, X. Chu, Y. Liu, L. Lin, Z. Yang, Z. Xu, C. Cao, F. Yu, C. Zhou, C. Yuan, and Y. Li. 2023. Accurate 3D Face Reconstruction with Facial Component Tokens. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE Computer Society, Los Alamitos, CA, USA, 8999–9008.

Yufeng Zheng, Victoria Fernández Abrevaya, Marcel C. Bühler, Xu Chen, Michael J. Black, and Otmar Hilliges. 2022. I M Avatar: Implicit Morphable Head Avatars from Videos. In *Computer Vision and Pattern Recognition (CVPR)*.

Yufeng Zheng, Wang Yifan, Gordon Wetzstein, Michael J. Black, and Otmar Hilliges. 2023. PointAvatar: Deformable Point-based Head Avatars from Videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Mingyuan Zhou, Rakib Hyder, Ziwei Xuan, and Guojun Qi. 2024. UltrAvatar: A Realistic Animatable 3D Avatar Diffusion Model with Authenticity Guided Textures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 1238–1248.

ChangAn Zhu and Chris Joslin. 2024a. A Facial Motion Retargeting Pipeline for Appearance Agnostic 3D Characters. In *Proceedings of Computer Graphics International 2024* (Campus Biotech, Geneva, Switzerland) *(CGI 2024)*. Association for Computing Machinery, New York, NY, USA, 1–15.

ChangAn Zhu and Chris Joslin. 2024b. A review of motion retargeting techniques for 3D character facial animation. *Computers & Graphics* 123 (2024), 104037. https://doi.org/10.1016/j.cag.2024.104037

Yilin Zhu, Dalton Omens, Haodi He, and Ron Fedkiw. 2024. Democratizing the Creation of Animatable Facial Avatars. arXiv:2401.16534 [cs.GR] https://arxiv.org/abs/2401.16534

Wojciech Zielonka, Timo Bolkart, and Justus Thies. 2022a. Instant Volumetric Head Avatars. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2022), 4574–4584. https://api.semanticscholar.org/CorpusID:253761096

Wojciech Zielonka, Timo Bolkart, and Justus Thies. 2022b. Towards Metrical Reconstruction of Human Faces. In *European Conference on Computer Vision*.

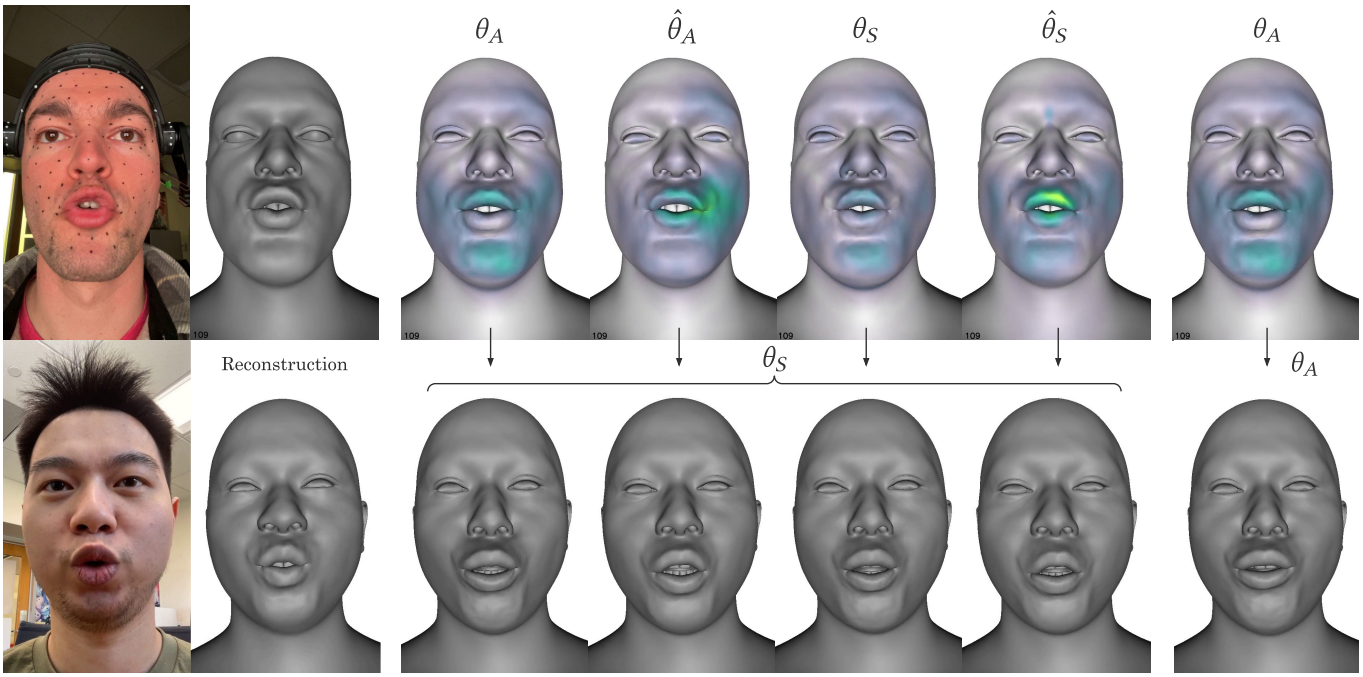

Fig. 23. Open-source tracker: retargeting to another identity. The top row shows geometry errors for the different trackers, and the bottom row shows the retargeted results. Both subjects are speaking the same word but have a different motion signature. Modifying $\theta_A \rightarrow \hat{\theta}_A$ and $\theta_S \rightarrow \hat{\theta}_S$ both improve the retargeted results. The last column shows a state-of-the-art off-the-shelf baseline, not using any of our approaches, for the sake of comparison.

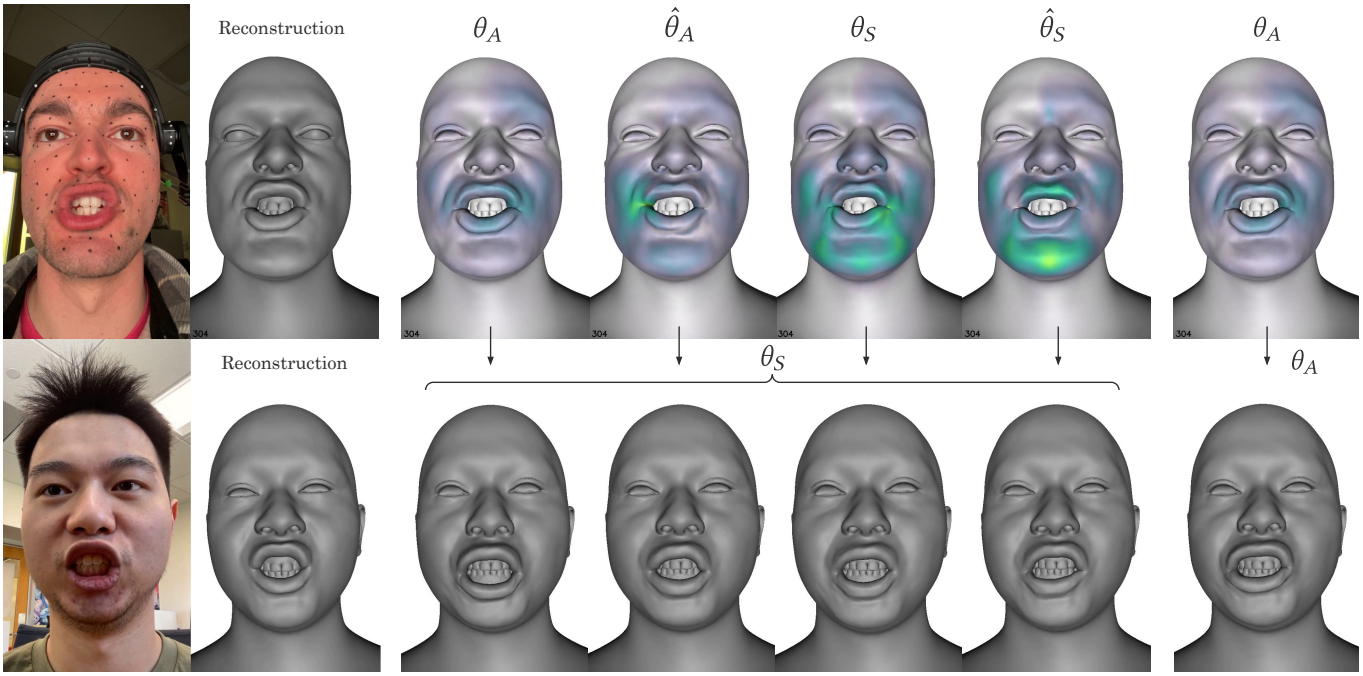

Fig. 24. Open-source tracker: retargeting to another identity. The top row shows geometry errors for the different trackers, and the bottom row shows the retargeted results. On this funnel-type expression, modifying $\theta_A \rightarrow \theta_S$ improves the severity of the expression due to the difference in motion signatures, while modifying $\theta_A \rightarrow \hat{\theta}_A$ and $\theta_S \rightarrow \hat{\theta}_S$ improves lip shape as compared to the reference. The last column shows a state-of-the-art off-the-shelf baseline, not using any of our approaches, for the sake of comparison.

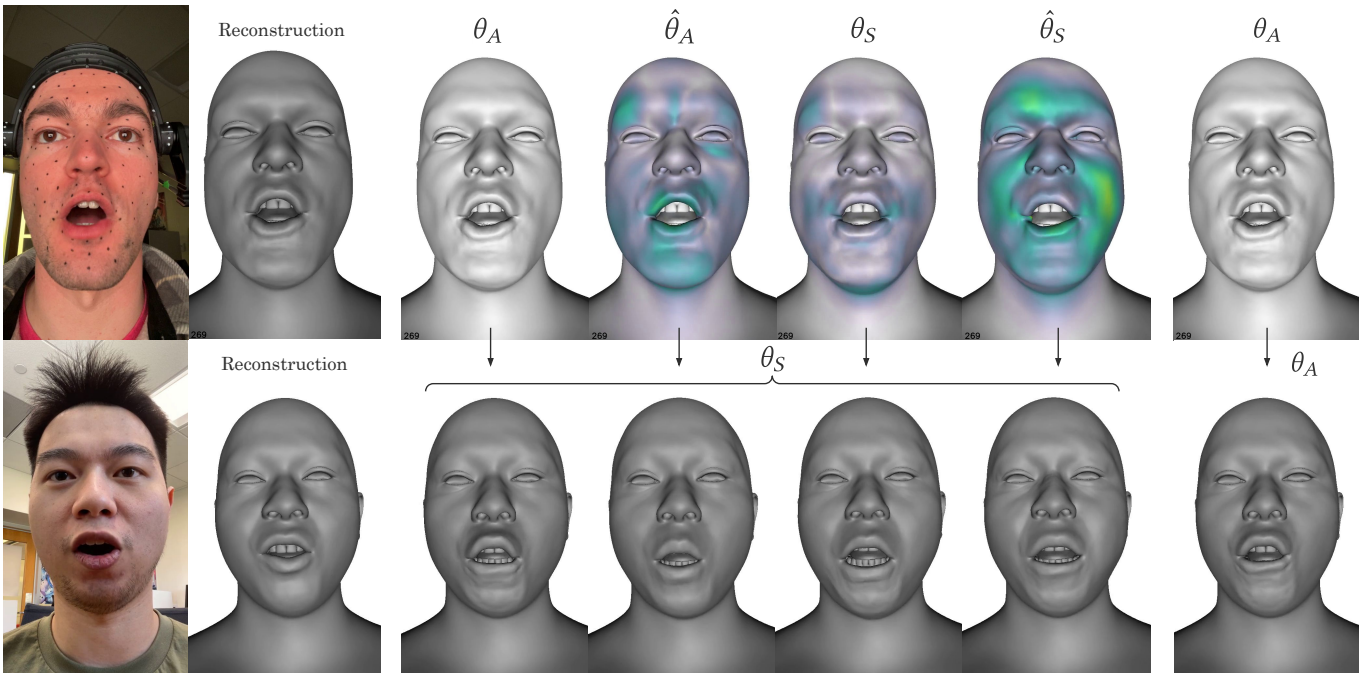

Fig. 25. Closed-source tracker: retargeting to another identity. The top row shows geometry errors for the different trackers, and the bottom row shows the retargeted results. This example was chosen to highlight the plausible behavior of our approach under asymmetries, even though both the Simon-Says calibration and the tracker optimization only considered expressions that were aspirationally symmetric. The last column shows a state-of-the-art off-the-shelf baseline, not using any of our approaches, for the sake of comparison.

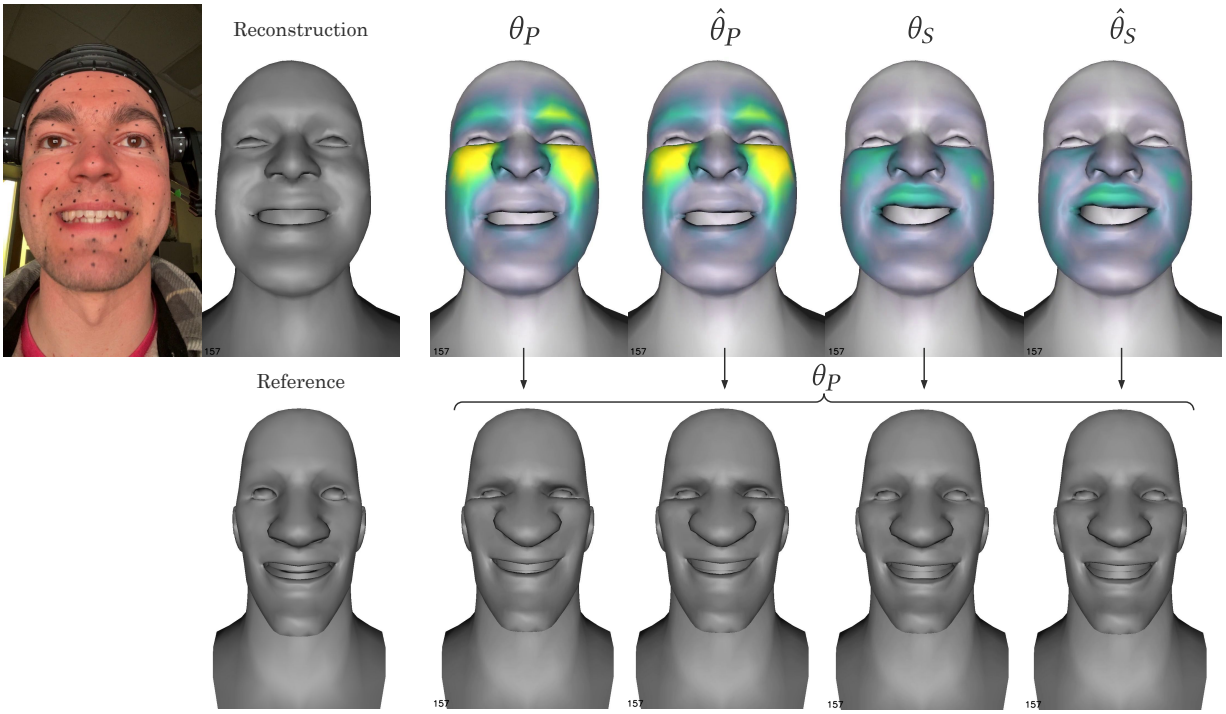

Fig. 26. Open-source tracker: puppeteering. The top row shows geometry errors for the different trackers, and the bottom row shows the retargeted results. In this expression, extraneous controls in the eye region activated by the uncalibrated rig parameters θ_P are remedied by θ_S. In addition, $\hat{\theta}_S$ retargets to a mouth shape that better matches the reference for this phoneme, despite additional reconstruction errors being introduced around the mouth.

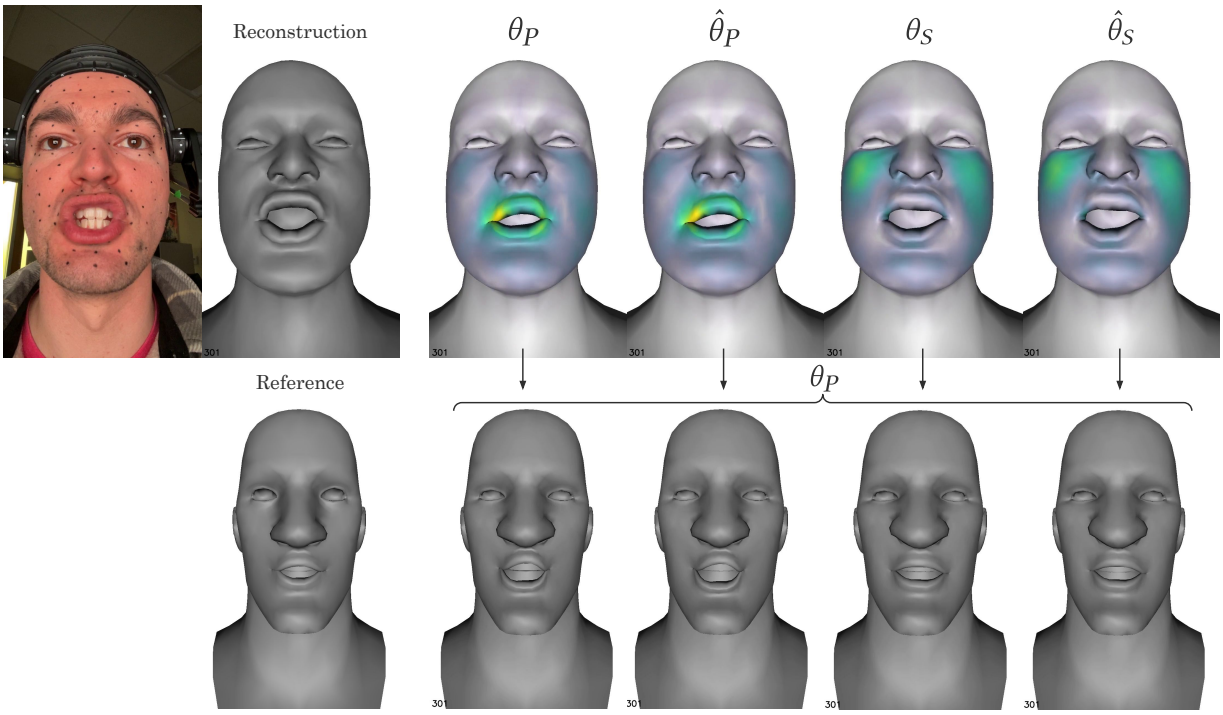

Fig. 27. Open-source tracker: puppeteering. The top row shows geometry errors for the different trackers, and the bottom row shows the retargeted results. The "CH" phoneme control is activated correctly using θ_S and $\hat{\theta}_S$, resulting in the retargeted expression more closely matching the reference. The θ_P and $\hat{\theta}_P$ results over-emphasize the lip movement in the retarget since the their motion signatures are significantly different.

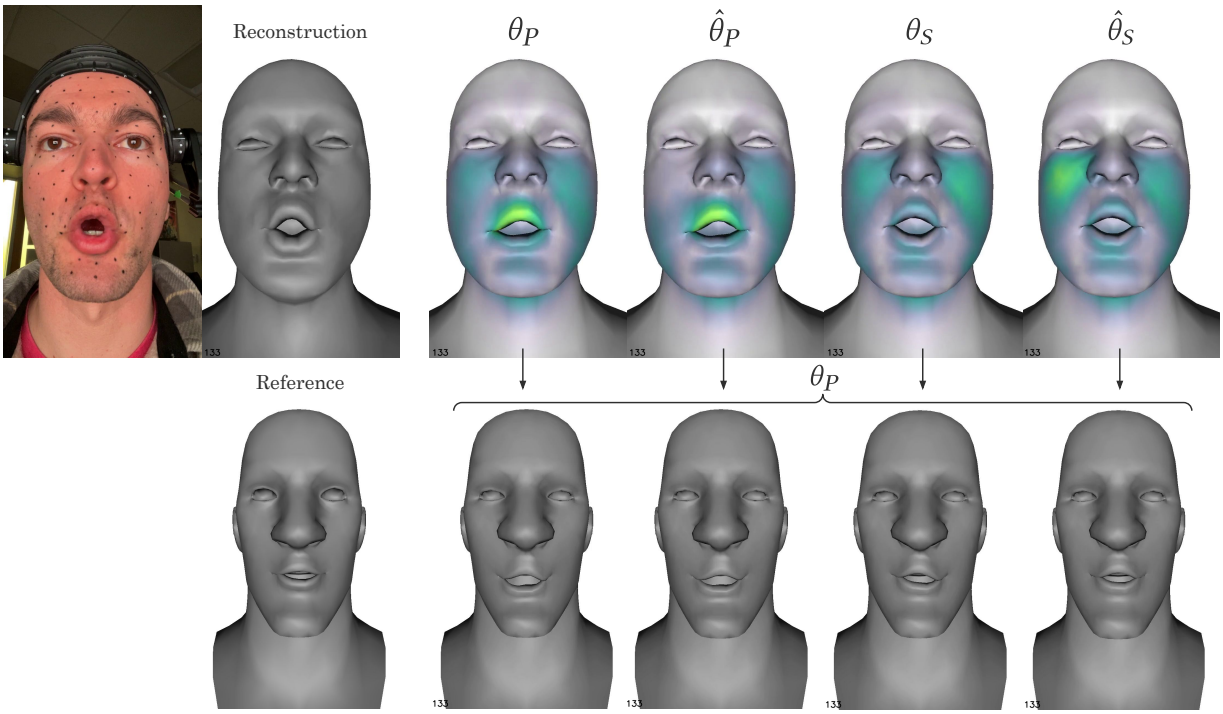

Fig. 28. Open-source tracker: puppeteering. The top row shows geometry errors for the different trackers, and the bottom row shows the retargeted results. The "OO" phoneme control is activated more correctly when using θ_S, and even more correctly when using $\hat{\theta}_S$.

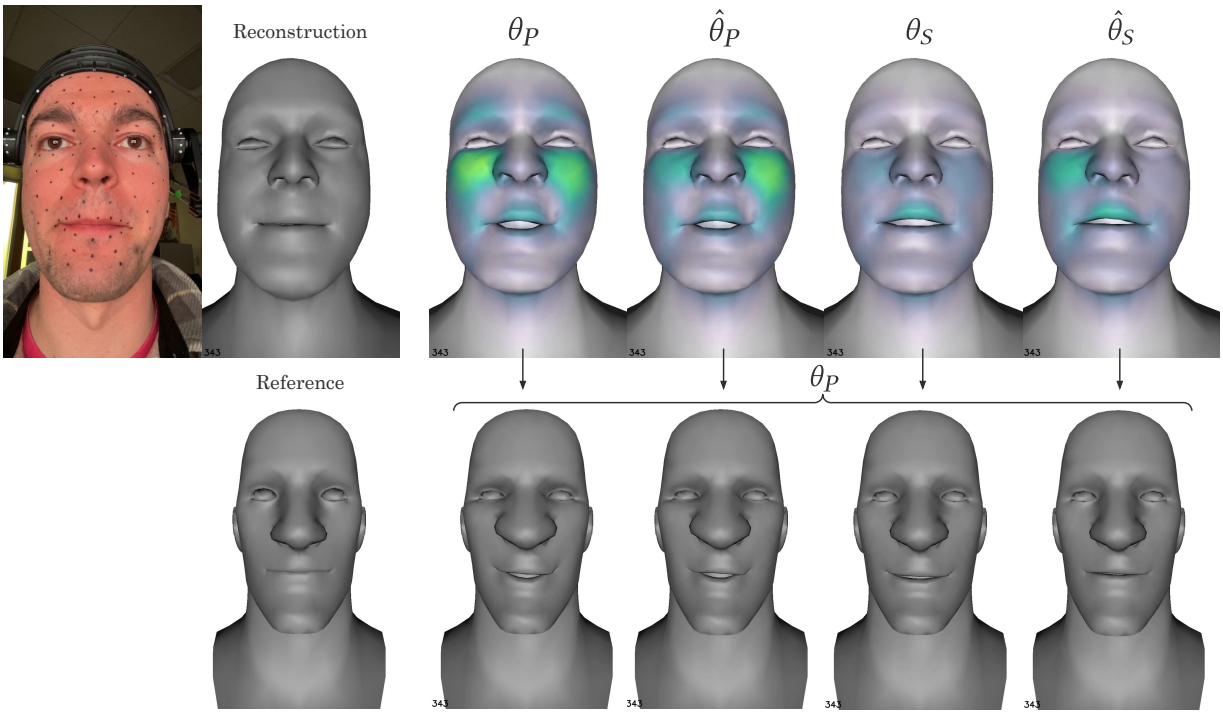

Fig. 29. Open-source tracker: puppeteering. The top row shows geometry errors for the different trackers, and the bottom row shows the retargeted results. The "M/B/P" phoneme control improves when using θ_S and improves even more when using $\hat{\theta}_S$, resulting in the better lip sealing. This is especially important for avoiding the uncanny valley.

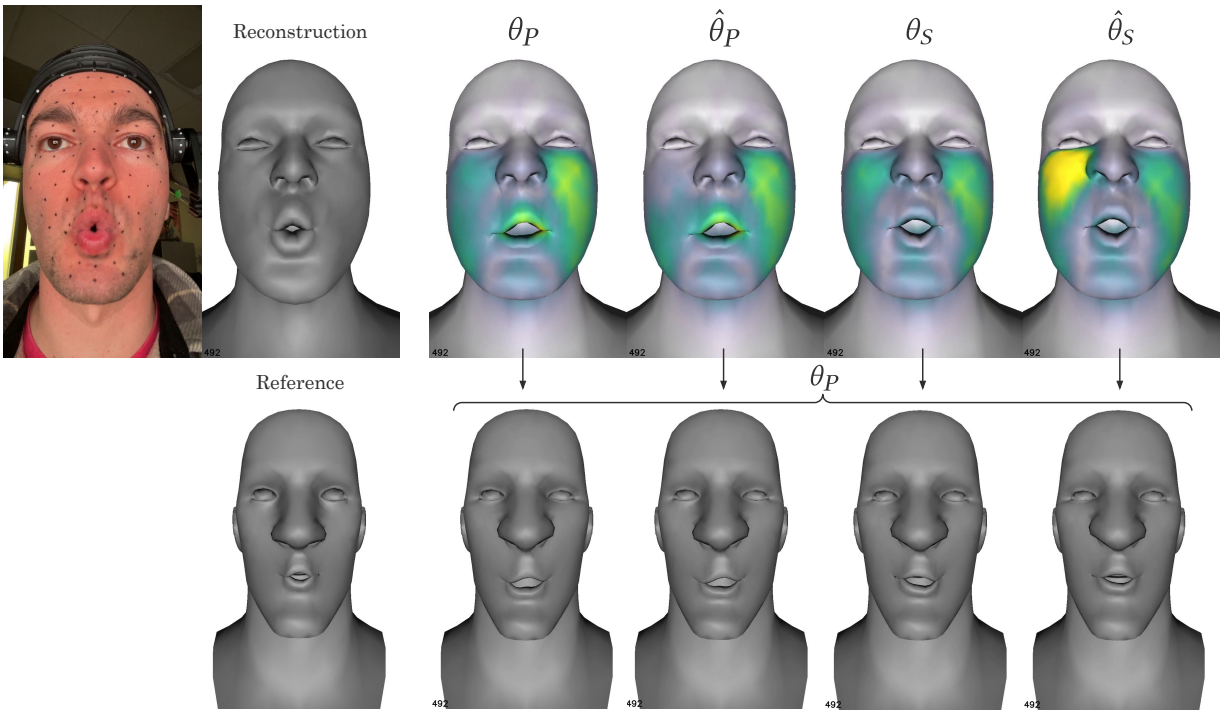

Fig. 30. Open-source tracker: puppeteering. The top row shows geometry errors for the different trackers, and the bottom row shows the retargeted results. Lip tightness and shape are improved during the word "quacking" when using θ_S and even much more so when using $\hat{\theta}_S$.