Example-based Data Optimization for Facial Simulation

Sara C. Schvartzman* Marco Romeo†

MPC

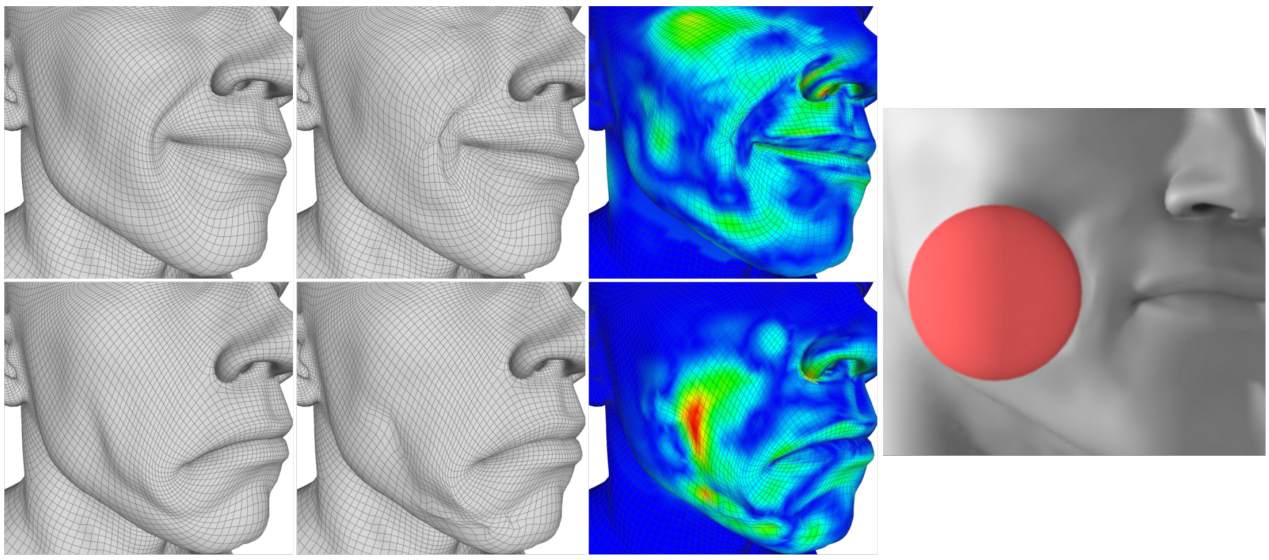

Figure 1: *From left to right columns: two input target poses, their simulated counterparts after optimization, the error maps and a dynamic scene where the resulting deformable interacts with a rigid sphere.*

Keywords: facial animation, physically-based simulation, data optimization

Concepts: •**Computing methodologies** → **Physical simulation; Simulation by animation;**

1 Introduction

Digital characters are common in modern films visual effects and the demand for digital actors has increased during the past few years. The success of digitally created actors is related to their believability and, in particular, the realism of the animation and simulation of their faces. Facial expressions in computer graphics are commonly obtained through linear vertex interpolation techniques such as blend shapes. These enable high artistic control and fast interaction, but cannot properly reproduce collisions or other physical phenomena such as gravity and inertia. These effects can be achieved by applying simulation techniques over the animated facial geometry (e.g. muscle simulation), but could potentially alter the look of the desired facial expression and produce inconsistencies with the work approved in animation. Moreover, animating such muscle rigs can be very cumbersome.

In this poster, we propose an optimization algorithm that uses digitally sculpted blend targets to find the forces and material properties to produce a Finite Element Method (FEM) simulation that is faithful to the facial expressions produced by animators. Our work extends the results of [Bickel et al. 2009] by adding the optimization of forces, similar to what is described in [Skouras et al. 2013], but without relying on a library of predefined materials.

The result of our work will enable a creative workflow that lets artists sculpt the required shapes and create the main animation for the character. The system then replaces the animation with a simulated version that preserves their artistic decisions and enriches the results to convey more realism. Furthermore, this approach removes the need to create and setup complex muscle rigs, can be applied to fictitious creatures and is artistically controllable by nature. Finally, we believe it could be used to generate more target shapes, as a simulated sculpting tool.

2 Optimization process

We first generate a tetrahedral mesh which is defined by the space between the skin and the skull, and will deform an embedded surface mesh. In contrast to [Bickel et al. 2009] and [Skouras et al. 2013], we use an invertible FEM with Neo-Hookean isotropic material for our simulation. For the optimization, we use a quasi-static solver.

The nodes of the tetrahedral mesh are classified into three groups: fixed nodes, control nodes and free nodes. The layer of nodes closest to the skull are set to fixed. The control nodes are the ones to which forces will be applied to. The remaining nodes are classified

*e-mail:sara-sc@moving-picture.com

†e-mail:marco-ro@moving-picture.com

SIGGRAPH '16, July 24-28, 2016, Anaheim, CA,
ISBN: 978-1-4503-4371-8/16/07
DOI: http://dx.doi.org/10.1145/2945078.2945151

as free.

The control nodes are found through an optimization process that tries to reduce the number of forces to apply while still achieving a simulation close to the targets. The final control nodes are the ones that have a maximum force applied greater than a specified threshold.

Once we have defined the control nodes, the optimization process iteratively alternates between force and material optimization, starting from a flesh-like material (Young's modulus=$60000N/m^2$, Poisson's ratio=0.4). For both optimization processes we have used the Levenberg-Marquardt algorithm defined by these function goals:

$$\begin{aligned} E_f(f) &= \frac{1}{2}\left\|y - x(f)\right\|^2 + \frac{1}{2}\gamma_f \left\|f\right\|^2 \\ E_m(p) &= \frac{1}{2}\left\|y - x(p)\right\|^2 + \frac{1}{2}\gamma_m \left\|Lp\right\|^2 \end{aligned} \tag{1}$$

where y are the positions of the target vertices and normals of the target faces stacked together into a single vector, and $x(-)$ are the simulated counterparts. To calculate the $x(-)$, we first setup the new material if necessary. Then, for each target, we reset the simulation, apply the appropriate forces and advance one step of the quasi-static simulation. The updated positions of the vertices and normals of faces of the embedded mesh are stacked into a single vector to compare with y.

To prevent underconstrained systems, we add a regularization parameter for each optimization problem. For the force optimization we prevent the use of large forces by penalizing its norm with γ_f, and for the material optimization we encourage material smoothness between neighboring tetrahedra with γ_m, such as described in [Bickel et al. 2009]. L is a $2T \times 2T$ matrix (being T the number of tetrahedra), where each element w_{ii} is set to the number of vertex-neighbors of tetrahedra i, and the elements w_{ij} are -1 if j is a vertex-neighbor of i.

These regularizations variables γ_m and γ_f are decreased in each outer iteration in order to relax the regularization and reduce the distance error.

3 Jacobian computations

In order to execute the optimizations, we must define the Jacobian functions of each optimization problem (i.e. $\frac{\partial x(f)}{\partial f}$ and $\frac{\partial x(p)}{\partial p}$). From the quasi-static equation $K(x - x_0) = f$ we obtain the Jacobian of the material optimization as $K\frac{\partial x(p)}{\partial p} = \frac{\partial K}{\partial p}(x - x_0)$ where x_0 are the rest positions of the tetrahedra nodes.

The derivative of the stiffness matrix is calculated by simply replacing the derivation of the density function with respect to the invariants $\frac{\partial \Psi}{\partial I_i}$ by its derivative with respect to the parameters $\frac{\partial \Psi / \partial I_i}{\partial p}$ when computing the stiffness matrix as described in [Teran et al. 2005]:

$$\begin{aligned} \Psi &= \frac{\mu}{2}(I_1 - 3) - \mu\log(\sqrt{I_3}) + \frac{\lambda}{2}(\log(\sqrt{I_3}))^2 \\ \frac{\partial \Psi / \partial I_1}{\partial \mu} &= \frac{1}{2}, \quad \frac{\partial \Psi / \partial I_3}{\partial \mu} = \frac{-1}{2I_3} \\ \frac{\partial \Psi / \partial I_3}{\partial \lambda} &= \frac{log(I_3)}{4I_3} \end{aligned} \tag{2}$$

where μ and λ are the Lamé parameters of each tetrahedron, and I_1 and I_3 are the first and third invariants.

The calculation of the force Jacobian is straightforward from the quasi-static equation and assuming we start from a rest pose.

#Verts	#Nodes	#Tetras	#OT	Size	Target	#CN	Error
3288	3591	12762	6455	13	Smile	130	0.34/0.06
					Frown	53	0.26/0.03

Table 1: *Data about the optimization shown in Figure 1. From left to right the columns describe the number of vertices of the surface mesh, the number of nodes of the tetrahedral mesh, the number of tetrahedra, the number of tetrahedra considered for material optimization, the size of the maximum length of the bounding box in cm, the targets, the number of control nodes per target, and the max/mean error in cm.*

4 Results and future work

During the final simulation, when a target i is activated with weight $w_i \in [0, 1]$, we apply the results of the force optimization as displacements of the control nodes scaled by w_i.

Figure 1 shows how the simulated versions of the targets on the characters are very similar to their blend-shape counterparts. Furthermore, the simulated version will respond to dynamics such as collisions and inertia.

The example shown in this abstract was optimized in approximately 1.5 hours. For better performance and as shown in Figure 1, the targets and surface mesh only cover a patch of the face, and the optimization algorithm only changes the material parameters of the tetrahedra that are near that patch. The result show a maximum error of 0.34 cm for the first target, and 0.26 cm for the second target. More details are displayed in Table 1.

There are still other aspects that could improve the optimization even further. Adding a constraint to the optimization process to enforce the goal to be reached when the system is at equilibrium (i.e. the internal forces are balanced with the external forces) would improve the dynamic simulation of the final result.

We have observed that optimizing without taking the normals into account produced a smaller error, but resulted in visible artifacts. We think that adding a regularization parameter that compares the Laplacian operator of the vertices might improve the result allowing the user to prioritize positions over smoothness if necessary.

Finally, an implementation of anisotropic and non-linear materials would provide much wider and richer possibilities for the optimization.

References

BICKEL, B., BÄCHER, M., OTADUY, M. A., MATUSIK, W., PFISTER, H., AND GROSS, M. 2009. Capture and modeling of non-linear heterogeneous soft tissue. In *ACM SIGGRAPH 2009 Papers*, ACM, New York, NY, USA, SIGGRAPH '09, 89:1–89:9.

SKOURAS, M., THOMASZEWSKI, B., COROS, S., BICKEL, B., AND GROSS, M. 2013. Computational design of actuated deformable characters. *ACM Trans. Graph. 32*, 4 (July), 82:1–82:10.

TERAN, J., SIFAKIS, E., IRVING, G., AND FEDKIW, R. 2005. Robust quasistatic finite elements and flesh simulation. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ACM, New York, NY, USA, SCA '05, 181–190.