EUROGRAPHICS 2016 / J. Jorge and M. Lin
(Guest Editors)

Volume 35 (2016), Number 2

BlendForces: A Dynamic Framework for Facial Animation

Vincent Barrielle[1,2], Nicolas Stoiber[2], Cédric Cagniart

[1] Centrale-Supélec [2] Dynamixyz

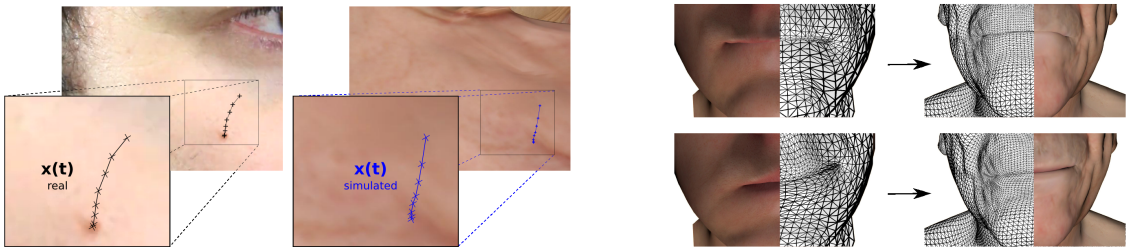


Figure 1: Blendforces merge blendshapes and physically-based animation, allowing for dynamic effects and lip collision handling.

Abstract

In this paper we present a new paradigm for the generation and retargeting of facial animation. Like a vast majority of the approaches that have adressed these topics, our formalism is built on blendshapes. However, where prior works have generally encoded facial geometry using a low dimensional basis of these blendshapes, we propose to encode facial dynamics by looking at blendshapes as a basis of forces *rather than a basis of* shapes*. We develop this idea into a dynamic model that naturally combines the blendshapes paradigm with physics-based techniques for the simulation of deforming meshes. Because it escapes the linear span of the shape basis through time-integration and physics-inspired simulation, this approach has a wider expressive range than previous blendshape-based methods. Its inherent physically-based formulation also enables the simulation of more advanced physical interactions, such as collision responses on lip contacts.*

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Animation—Facial Animation

1. Introduction

Blendshape models have become a standard tool for facial capture and animation retargeting over the past 15 years [LAR*14], and are commonly found at the core of the facial production pipelines of large visual effect studios. In one of its common embodiments, a blendshape model expresses the geometry of a face $\mathbf{x}$ from a neutral shape $\mathbf{x_0}$, a basis of geometric deltas organized as the columns of a matrix $\mathbf{B}$ and a weight vector $\mathbf{w}$.

$$\mathbf{x} = \mathbf{B}\mathbf{w} + \mathbf{x}_0, \tag{1}$$

where $\mathbf{x}$ and $\mathbf{x_0}$ contain the cartesian coordinates of all the vertices from the mesh representing the face. In the context of facial motion capture or facial animation, temporal sequences of facial movements are represented as a time series of blendshape weights $\{\mathbf{w}_t\}$, encoding the successive mesh geometries $\{\mathbf{x}_t\}$ over time.

Encoding facial geometry in this fashion is attractive for a number of reasons: constraining the mesh of a face to lie in a affine subspace limits the occurences of grossly degenerate configurations for reasonable values of $\mathbf{w}$ (typically $[0,1]$). Furthermore, solvers benefit from the *simplicity* of this affine model because it allows to manipulate the mesh geometry without introducing non-linear terms in objective functions. Additionally, the expressive range of blendshape models can be broadened by adding more shapes to $\mathbf{B}$. This offers a predictable *scalability* of the quality of the output and it is not unusual to see animators use thousands of actor-specific shapes for virtual characters in large movie productions [SILN11]. Finally, and possibly most importantly, blending shapes is a sufficiently intuitive paradigm that such models can be *artistically driven*. For example, the commonly used Facial Action Coding System [EF77] defines a basis of shapes that have semantic value and spatial locality, which allows them to be easily interpreted and manipulated by animators [LAR*14].

Despite the simplicity of equation (1), practical experience

of blendshape models reveals that the space of facial expressions of a given character should not be considered affine, as not all values of $\mathbf{w}$ yield plausible facial geometries. As mentioned in [LAR*14], animators routinely use 'intermediate shapes', 'combination shapes' or Pose Space Deformation-based corrections [SSK*12] that introduce non linearities not represented by the affine model. Furthermore, animators tend to only activate a small subset of shapes at any given time, which suggests that blendshapes could be viewed as a dictionary of samples from an hypothesized manifold of facial expressions. Developing a blendshape puppet therefore requires to model enough meshes to sample this manifold and to define the local interpolation functions. It is usually an iterative and very labor intensive process where the expressive range of an initial dictionary of primary shapes (typically FACS) needs to iteratively be widened by modeling new facial expressions.

From a *geometric* perspective, this can be understood as iteratively adding columns to $\mathbf{B}$ so that the model will be able to express new facial configurations that used to lie outside its column space $C(\mathbf{B})$. A number of methods have been proposed to generate plausible new facial geometries that escape this subspace: some authors [TDlTM11] have split the face into segments that are each driven by a local affine model. Others [MWF*11, LYYB13, BWP13, HMYL15] have used captured geometric data regularized by physics-inspired deformation priors to generate new shapes. Rendering more complex physical behaviors such as lip contact and compression effects proves more difficult for such purely data-based methods, as lip interaction data is hard to capture reliably.

From a *temporal* perspective, linearly combining blendshapes also has the effect of constraining the velocities of vertices to lie in $C(\mathbf{B})$, which means that groups of vertices tend to move jointly as "blocks" over time according to the geometric patterns encoded in the activated shapes. This limits the ability of blendshape-based methods to render fine temporal behaviors. Capturing dynamic details such as differences in muscle activation timings within the blendshape paradigm therefore tends to require the introduction of new intermediate shapes and correctives.

In this paper we propose to revisit the relation between physic-inspired mesh deformation methods and blendshapes, which are a data-based paradigm. We notice that most previous works have considered blendshape solving in quasi-static formulations, ignoring the dynamic nature of the facial performance. The key insight behind this paper is that a database of blendshapes can be interpreted as a prior on facial motion rather than facial geometry only. We build on this idea by modeling the face as a physical system and introduce *blendforces*: a set of actuators whose forces lie in the linear span of $\mathbf{B}$. The temporal evolution of the facial mesh is then obtained by integration of the laws of motion under the combined action of these data-based forces and other physically-inspired reactive forces. The work presented here applies this new paradigm to performance-driven animation and to animation retargeting. In section 3, we discuss blendforces and present the other forces at play in the system. We then introduce a dynamic control framework that computes a command for the actuators so that the simulated mesh will fit marker trajectories measured in 3D space. This approach is evaluated experimentally in section 4. The results confirm that blendforces has a wider expressive range than blendshape methods, that it leads to higher fidelity to facial performances, and that its formalism allows to integrate with more advanced physical priors such as collision responses.

2. Related Works

Blendshape-based facial animation Facial animation is often acquired through performance capture, where blendshape weights are optimized to match real-world measurements such as marker trajectories. To prevent degenerate geometries and reduce the risk of mixing incompatible shapes, it is possible to incorporate regularization priors in the fitting process. In that regard, techniques like non-negative weights solving, sequential fitting [SSK*11], or L1-norm regularization [BWP13] are widespread.

To achieve the kind of precision required in high-end motion capture and animation, practical face capture systems tend to rely on models built from performer-specific shapes. The goal is to make sure that the column-space of $\mathbf{B}$ covers all the facial deformation modes that we wish to capture. These shapes are usually acquired by dedicated hardware in a controlled environment [WLVG07, BHB*11], or transfered from an existing basis of another face [SP04, LWP10].

It is often the case however, that blendshape models do not span the complete expression space of the performer. To handle this limitation, [MFD11, KMS11] segment the face into regions and solve for a different set of blendshape weights in each of them. This certainly increases the range of expressions one can capture with a given blendshapes basis, yet it can result in a lack of coherence between segments and raises questions on the treatment of segment boundaries [TDlTM11].

The quality of a performance-driven animation lies as much in the range of expressions recovered from the performance as in the accuracy of reconstructed skin motion. Blendshapes facial animation obtained from sparse marker tracking typically lacks mid- to fine-scale details necessary to produce a realistic result [BBB*14]. These fine-scale deformation details can be added on top of a baseline blendshape animation using physics-inspired [BBA*07] or data-driven techniques [MJC*08, BBB*14].

More recent works have presented a different approach to improve both large- and fine-scale fidelity of performance-based animation. Working off an initial blendshapes basis, [LYYB13] augment it with new shape deformation patterns captured in a real-time face tracking scenario. In a related effort [BWP13] add low-energy eigenvectors of the geometry Laplacian to their shape basis, which improves the fitting of mid- to fine-scale deformations. Recent developments [CWLZ13, CHZ14] continuously adapt an underlying blendshapes representation by refining the identity parameter of a multilinear shape representation.

In the pursuit of escaping the column-space of $\mathbf{B}$ to produce more faithful output geometry, our work differs from recent blendshape-based approaches [LYYB13, BWP13] in that instead of appending physically-plausible shapes to a blendshape basis it integrates blendshapes in a physics-inspired simulation.

Physics-based facial animation The deformation of human skin is the product of complex skull-muscle-derma interactions and

skin-tissue dynamics. As such, a number of previous works have proposed physics-inspired models of the human face, and produced facial animation through physical simulation of those models.

Early attempts at a physics-based representation relied on mass-spring systems, actuated by vector muscles [PB81,Wat87]. [TW93] proposed an animation system in which muscle actuation parameters are computed from video-based tracking of facial features. Later works investigated more anatomically-accurate approaches by using volumetric Finite Element Methods (FEM) for skin tissues [KGPG96, KGC*96]. As with mass-spring systems, studies have proposed methods to recover muscle actuation parameters for those models, either through inverse dynamics approaches [EBDP96, PM01] or with a quasi-static formulation [SNF05].

Physics-based methods work off sound physical foundations and generally produce accurate skin motion, yet they are harder to implement in practice. Building a functional physical model of the face requires medical expertise, and most of the time access to laser and MRI measurements of a face to set physical parameters. Besides, animation production pipelines often require having fine control over results, a straightforward characteristic of data-based frameworks but harder to guarantee with physics-based approaches. In an effort to bring physical simulation to traditional animation pipelines, Rig-space physics [HMT*12,HTC*13] explored the interactions between physical models and artistic rigging to produce plausible secondary motion for body sequences. Recent developments brought secondary motion to arbitrary triangle mesh sequences, with applications to enhancement of facial performance captures [LXB15]. Our approach differs in that the physical simulation is incorporated into the performance capture process.

Our work shares some similarities with [MWF*11], where the rest lengths of a mass-spring system are obtained by interpolating between blendshapes edge lengths. This system provides a shape interpolation framework capable of physical interaction. We push the association of blendshapes and physical simulation further, by considering that the skin is a passive tissue reacting to the action of muscles, and by using blendshapes to model muscle forces.

3. Blendforces

In contrast to many facial animation pipelines, which rely on blendshapes to encode facial performances as a succession of static geometric configurations, our work focuses on the dynamic nature of the process. We represent the face as a physical system, and parameterize its state at time t by two vectors $\mathbf{x}_t$ and $\dot{\mathbf{x}}_t$ containing respectively the positions and velocities of the vertices on the mesh. These vertices are put in motion by an *actuation* force $\mathbf{f}_t^a$ and system forces $\{\mathbf{f}_t^i\}$. Following Newton's second law:

$$\mathbf{M}\ddot{\mathbf{x}}_t = \mathbf{f}_t^a + \sum_i \mathbf{f}_t^i, \tag{2}$$

where $\mathbf{M}$ is a diagonal mass matrix.

3.1. The Case for Blendshapes as Actuators

The innovative part of our work is the introduction of actuators whose forces lie in the low-dimensional linear subspace spanned by a set of blendshapes. Through those actuators we interpret blendshapes not as a basis of the face geometry, but as a basis of forces that steer the evolution of the system. We name $\mathbf{f}_t^a$ the sum of these forces at time t:

$$\mathbf{f}_t^a = \mathbf{B}\mathbf{u}_t, \tag{3}$$

where $\mathbf{B}$ is a matrix of delta blendshapes and $\mathbf{u}_t$ is a weight vector representing the command.

Let us briefly compare this formulation to the delta-blendshapes of equation (1). If we assume a uniform unit mass and ignore system forces, equation (1) and the integrated blendforces of equation (2) can be related by choosing $\mathbf{u}_t = \ddot{\mathbf{w}}_t$ on every time segment where $\mathbf{w}_t$ is twice differentiable. In other words, if $\mathbf{w}$ and $\dot{\mathbf{w}}$ are prescribed at the boundaries of these segment, blendforces can express a blendshape animation. The following paragraphs will discuss how the key differences between the blendshapes and blendforces paradigms only really appear once we consider the problem of synthesizing plausible facial animations.

On one side the blendshapes paradigm starts from a set of sample facial configurations and needs to define a prior distribution for plausible faces on the affine space parameterized by $\mathbf{w}$. This prior is generally accounted for by some loss function on various norms of $\mathbf{w}$. The L1 or L2 norms are often used but it is also possible to use statistics-based (PCA) or physics-inspired metrics in the hope that they will encode a more meaningful geometry prior. From a temporal point of view we find the same problem of defining what is a physically meaningful trajectory in the linear space of blendshape weights. However the problem is generally only addressed with a simple temporal smoothing of the function $\mathbf{w}_t$, which has uncertain physical meaning.

On the other side, the blendforces paradigm puts physical-plausibility in both space and time at the heart of its parametrization by modeling the face as a physical system and deforming it according to the laws of motion. The contribution of blendforces is to introduce a *data-based* prior on how the face moves. This prior builds on the original insight of a FACS basis: facial motions are the result of a small number of individual muscle actions. We deduce that they should therefore be initiated by a low dimensional basis of forces, and that FACS-based blendshapes should provide a good approximation of this basis. Secondary motions can then be produced by reactive forces (see next section).

We finish this section by proposing an interpretation of the blendforces approach that builds on some insights from [LAR*14]. In this survey, one interpretation of the blendshapes method is that $\mathbf{B}$ is the tangent space at point $\mathbf{x}_0$ of some hypothesized face manifold embedded in a $3N_{vertices}$ ambient space. An important question raised by this perspective is that of defining the exponential map that maps $\mathbf{w}$ not to the tangent space but to the manifold itself. From this point of view, second-order "combination" blendshapes can be seen as a way to approximate the curvature of the manifold, and using measurements and physic-inspired priors can be viewed as a projection from the tangent space to the manifold.

We believe that because the blendforces method simulates the mesh deformation, it is reasonable to assume that $\mathbf{x}_t$ stays at all time on some manifold of physically-plausible geometries. Equation (15) will show that contrary to blendshapes, the blendforces

approach does not add shapes to $\mathbf{x}_0$ but adds them to the *current* simulation state $\mathbf{x}_t$. As such, interpreting blendshapes as a basis of motion instead of shape seems to mean that we interpret them as vectors of the tangent space to the manifold at point $\mathbf{x}_t$ and not $\mathbf{x}_0$. This is a key difference if we consider the problem of reprojecting on or at least approaching the face manifold. The blendshape approach needs to project from faces that can be far from the linearization point $\mathbf{x}_0$. Meanwhile in the blendforces framework, this operation happens at every time step and is a local operation performed by the system forces.

3.2. Physical Model of the Face

The system forces are crucial in that they determine the plausibility of the system's dynamic behavior as well as its stability. In our experiments we consider two types of mechanical forces that react to the inputs from the actuators: a first set of conservative forces emulates various damped elastic behaviors that can be observed on the human face. The other forces deal with the response to self-collisions of the facial mesh.

Elastic response Our deformation model uses three elastic forces defined on the facial mesh (ν, ε), where ν is the set of vertices and ε the set of edges. As in many previous works the resistance to stretching and bending is approximated by penalizing the variations of some differential quantities with respect to a rest configuration.

The first force penalizes displacements of vertices away from their rest positions, and derives from the following potential:

$$W_{pos}(\mathbf{x}) = k_p \|\mathbf{x} - \mathbf{x}_0\|^2. \tag{4}$$

These springs mimic the derma's elasticity as it pulls the skin back towards its rest position once a muscle is no longer contracting. In practice they are quite loose and their effect is mostly observed when there is very little input from the actuators. As an implementation detail it should be noted that the rest positions of vertices belonging to the jaw are updated to account for the articulated nature of that part of the skull.

The second force limits the stretching of the skin by penalizing length variations for the edges in ε. These forces derive from a potential that is written here in a simplified form:

$$W_{stretch}(\mathbf{x}) = k_s \sum_{(i,j)\in\varepsilon} \left(\|\mathbf{x}^i - \mathbf{x}^j\| - r_{ij} \right)^2, \tag{5}$$

where the rest lengths $\{r_{ij}\}_{(i,j)\in\varepsilon}$ are computed from the neutral facial expression and $\mathbf{x}^i$ contains the elements of $\mathbf{x}$ that represent the cartesian coordinates of vertex i. In practice we follow the work of [LBOK13] to handle the non-linearity of these residuals that is due to the local rotations of the surface with respect to its rest configuration. Additionally we account for the non-linear stress-strain relationship of the skin [TW93] by activating an additional spring force with greater stiffness when the edge's elongation exceeds 10%.

The third force aims at maintaining the local mean curvature of the skin surface. It derives from a potential energy that penalizes variations of the Laplacian of the position function $\mathbf{x} : \nu \mapsto \mathbb{R}^3$:

$$W_{bend}(\mathbf{x}) = k_b \|\mathbf{L}\mathbf{x} - \delta(\mathbf{x})\|^2, \tag{6}$$

where $\mathbf{L}$ is the cotangent weight discretization of the Laplace-Beltrami operator [WMKG07]), and $\delta(\mathbf{x})$ contains the Laplacians of the surface in its rest configuration, rotated to account for the current state of the surface [SA07].

The stiffnesses $\mathbf{k_p}$, $\mathbf{k_s}$ and $\mathbf{k_b}$ were presented as scalars for notation clarity in the three system forces above, yet in practice they are defined vertex-wise on the facial mesh for the displacement and bending forces in (4) and (6), and edge-wise for the stretch force in (5). This is a noteworthy element, as the role of the system forces is to regularize the face towards physically-plausible configurations, and those configurations are heavily influenced by the distribution of stiffnesses over the mesh. In particular, it is important that the blendforces formulation allows the system to reproduce the blendshapes themselves, that are data-cues strongly representative of the character's expressive range. We formalize this by stating that the system should be able to achieve static equilibrium of all forces on each blendshape expression. That is, for each column of the blendshape matrix $\mathbf{B}$, the activation of the corresponding blend-force $\mathbf{f}^q = \mathbf{B}[:, j]$ should be equal to the internal forces on the mesh configuration $\mathbf{x}_0 + \mathbf{B}[:, j]$ that corresponds to the j-th expression. We therefore determine the stiffnesses distributions by solving the following non-linear least-squares problem:

$$\min_{\mathbf{k_p},\mathbf{k_s},\mathbf{k_b}} \sum_j \|\mathbf{B}[:, j] + \sum_i \mathbf{f}^i(\mathbf{x}_0 + \mathbf{B}[:, j])\|^2. \tag{7}$$

Implementation details on this optimization procedure are presented in appendix B. Note that this optimization needs to be carried out only once per animated character.

Contact response Our physical system is enhanced by a collision handling mechanism designed to prevent self-intersection and handle contact response forces between the upper and lower lips. This happens in two steps: collision detection and response forces computation.

We first detect collision by pruning the collision search space using optimized spatial hashing [THM*03], and detecting vertex-triangle and edge-edge collisions using the method of [BFA02]. Each detected collision yields four vertices $\mathbf{x}^0, \mathbf{x}^1, \mathbf{x}^2, \mathbf{x}^3$ defining the collision primitives (either triangle $\{\mathbf{x}^0, \mathbf{x}^1, \mathbf{x}^2\}$ and point $\mathbf{x}^3$ or edges $\{\mathbf{x}^0, \mathbf{x}^1\}$ and $\{\mathbf{x}^2, \mathbf{x}^3\}$), and barycentric coordinates w_0, w_1, w_2, w_3 specifying the location of the collision point within those primitives.

We then include a collision response to our physical simulation by defining an elastic collision potential. Let

$$\mathbf{s} = w_0\mathbf{x}^0 + w_1\mathbf{x}^1 + \alpha w_2\mathbf{x}^2 - w_3\mathbf{x}^3, \tag{8}$$

with $\alpha = -1$ for an edge-edge collision and $\alpha = 1$ for a point-triangle collision be the vector joining the collision points inside the second and first primitives respectively. With $\mathbf{n}$ the surface normal on the second primitive, we formulate the collision constraint that the primitives are no longer colliding as:

$$\mathbf{n} \cdot \mathbf{s} > 0. \tag{9}$$

Following [HVTG08], we assemble a sparse vector $\mathbf{c}$ such that $\mathbf{c} \cdot \mathbf{x} = \mathbf{n} \cdot \mathbf{s}$, and interpret $\mathbf{c} \cdot \mathbf{x}$ as an approximation of the signed distance between the colliding primitives. To avoid numerical issues,

we want colliding objects to always be separated by a non-zero distance τ (with $\tau = 0.01$mm in our implementation). We finally define the contact response potential as:

$$W_{\text{col}}(\mathbf{x}) = \begin{cases} \frac{k_{\text{col}}}{2}(\mathbf{c}\cdot\mathbf{x}-\tau)^2 & \text{if } \mathbf{c}\cdot\mathbf{x} < \tau \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

In addition, we simulate friction by damping the velocities of the colliding vertices in the plane perpendicular to $\mathbf{n}$ [MHHR07].

3.3. Dynamic Control for Performance-Driven Animation

The previous section introduced a mesh-based physical system that emulates basic physical properties of the skin, and whose deformations are driven by the set of actuators that we named *blendforces*. In this section we present a method that computes these blendforces so that the deforming mesh will best fit the measured trajectory of a sparse set of markers. We do not consider the sequence as a whole, but describe an *online* tracking process that steers the physical system to track marker trajectories. We propose to achieve this with a dynamic position-control framework that fits these measurements in the least-squares sense while maintaining a plausible physical behavior for the face as a whole. In this control framework, $\mathbf{x}_t$ is not manipulated directly but is the result of the temporal simulation of the physical system, steered with the command $\mathbf{u}_t$. Assuming that the state $(\mathbf{x}_{t-1}, \dot{\mathbf{x}}_{t-1})$ of the facial mesh at time $t-1$ is known, the objective is to find a simulation step that minimizes the following cost:

$$\arg\min_{\mathbf{u}_t} \|\mathbf{d}_t - \mathbf{S}\mathbf{x}_t(\mathbf{u}_t)\|^2, \tag{11}$$

where $\mathbf{S}$ is a vertex selection matrix matching the data markers with vertices on the facial mesh, and $\mathbf{d}_t$ contains the coordinates of the markers at time t. Here the notation is kept simple by assuming that all markers have equal weights and are seen at all times in the animation. We also assume that rigid motion has been factored out of the marker trajectories using Procrustes analysis.

We need to express $\mathbf{x}_t$ as a function of the command $\mathbf{u}_t$. The implicit Euler temporal integration of equation (2) yields:

$$\mathbf{x}_t = \mathbf{x}_{t-1} + h\dot{\mathbf{x}}_t, \tag{12a}$$

$$\dot{\mathbf{x}}_t = \dot{\mathbf{x}}_{t-1} + h\mathbf{M}^{-1}\left(\mathbf{f}_t^a + \sum_i \mathbf{f}^i(\mathbf{x}_t)\right), \tag{12b}$$

where h is the time step. Note that because the system forces are conservative, they only depend on the face geometry $\mathbf{x}_t$. Combined with the definition of the actuator forces from equation (3), the previous equations yield:

$$\mathbf{x}_t = \mathbf{x}_{t-1} + h\dot{\mathbf{x}}_{t-1} + h^2\mathbf{M}^{-1}\left(\mathbf{B}\mathbf{u}_t + \sum_i \mathbf{f}^i(\mathbf{x}_t)\right). \tag{13}$$

The difficulty introduced by implicit integration is that the non-linear system forces $\mathbf{f}^i$ are evaluated at $\mathbf{x}_t$, which is usually dealt with by locally linearizing them [BW98, LBOK13, BML*14]. We therefore define the following local affine approximations:

$$\mathbf{f}^i(\mathbf{x}_t) \approx \mathbf{A}_t^i\mathbf{x}_t + \mathbf{b}_t^i. \tag{14}$$

More details on how our implementation relates to some approximations used in previous works are presented in appendix A. Some of these approximations in particular consider constant $\mathbf{A}^i$ matrices, which can lead to efficient prefactorizations. For now, the linearization of equation (13) is written as:

$$\mathbf{x}_t \approx \Phi_t\mathbf{u}_t + \mathbf{y}_t, \text{with} \tag{15}$$

$$\Phi_t = (\mathbf{I} - h^2\mathbf{M}^{-1}\sum_i \mathbf{A}_t^i)^{-1}h^2\mathbf{M}^{-1}\mathbf{B}, \tag{16}$$

$$\mathbf{y}_t = (\mathbf{I} - h^2\mathbf{M}^{-1}\sum_i \mathbf{A}_t^i)^{-1}(\mathbf{x}_{t-1} + h\dot{\mathbf{x}}_{t-1} + h^2\mathbf{M}^{-1}\sum_i \mathbf{b}_t^i). \tag{17}$$

This allows to compute an estimated command $\hat{\mathbf{u}}_t$ that minimizes the local approximation of equation (11):

$$\hat{\mathbf{u}}_t = (\mathbf{S}\Phi_t)^+(\mathbf{d}_t - \mathbf{S}\mathbf{y}_t). \tag{18}$$

It is important to note that $\hat{\mathbf{u}}_t$ and $\mathbf{x}_t$ depend on each other because of our choice of linearization point for the forces. We therefore iteratively recompute the command $\hat{\mathbf{u}}_t$ so that both $\mathbf{x}_t$ and the linearized approximations of the system forces can be refined. A recapitulation of the full procedure for computing the blendforces and generating the corresponding facial animation is presented in algorithm 1.

```
x_1, ẋ_1 ← staticSolve(d_1), 0;
for t ∈ [2, N_frames] do
    x_t ← x_{t-1} + h ẋ_{t-1};
    for numRelinearizations do
        {A_t^i, b_t^i} ← linearizeForces(x_t);
        compute Φ_t and y_t;
        û_t ← leastSquaresSolve(Φ_t, y_t, d_t);
        x_t, ẋ_t ← simulateTimeStep(x_{t-1}, ẋ_{t-1}, û_t);
    end
end
return {x_t}_{t∈[1,N_frames]}
```

Algorithm 1: Marker-based blendforces animation

4. Experimental Validation

In this section we evaluate the method presented in section 3.3, that uses the blendforces paradigm as a performance-driven facial animation system. Blendshapes rigs are the usual representation of realistic face models, and they are commonly used to translate sparse marker data into animation for the face mesh. In the following, we quantitatively and visually compare animation results obtained by frame-wise least-squares blendshapes fitting and blendforces results obtained with the same blendshapes.

Using the same notations as in section 3, we define *static blendshapes solving* the process that fits rigidly stabilized marker data by finding the weights $\mathbf{w}_t$ that minimize the data term $\|\mathbf{d}_t - \mathbf{S}\mathbf{x}_0 - \mathbf{S}\mathbf{B}\mathbf{w}_t\|^2$. Practical implementations constrain the weights to positive values by using non-negative least-squares schemes, and commonly add regularization terms to stay away from grossly degenerate configurations, yielding the minimization problem:

$$\min_{\mathbf{w}_t > 0} \|\mathbf{d}_t - \mathbf{S}\mathbf{x}_0 - \mathbf{S}\mathbf{B}\mathbf{w}_t\|^2 + \alpha_1\|\mathbf{w}_t\|_1 + \alpha_2\|\mathbf{w}_t\|_2^2 \tag{19}$$

The L2-norm term penalizes large values for the components of

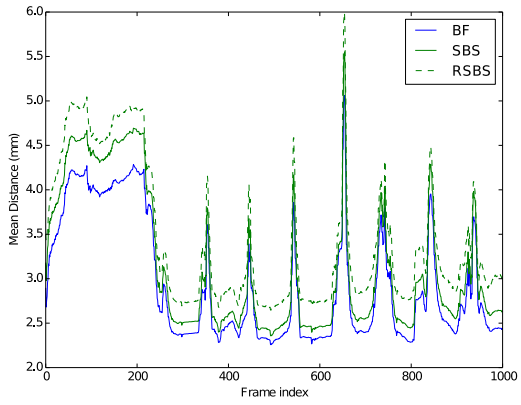


Figure 2: Mean Euclidean distance between motion capture marker positions and corresponding vertices for different solvers on a sequence of speech and varied facial expressions. The same set of artist-crafted blendshapes was used for all solves. As it is limited to facial configurations within the linear span of $\mathbf{B}$, static blendshape fitting (SBS) shows a noticeable fitting error throughout the sequence. Its regularized counterpart (RSBS) even more so, as it must satisfy additional soft constraints (low weights, sparsity). The blendforces solve (BF) consistently reduces fitting error on data marker positions. The mean distances over the whole sequence for SBS, RSBS and BF are 3.17mm, 3.45mm and 2.95mm respectively.

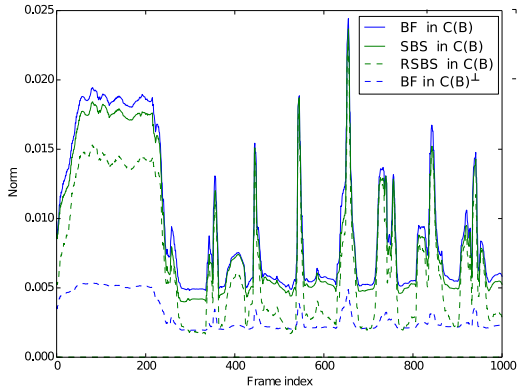


Figure 3: Measure of the projections of $\mathbf{x}$ in the respective subspaces $C(\mathbf{B})$ and $C(\mathbf{B})^{\perp}$ for different solvers. All measures are normalized by the number of vertices. The blendforces meshes shows a higher contribution of components from $C(\mathbf{B})$, as well as a considerable contribution from components outside $C(\mathbf{B})^{\perp}$. The contributions from $C(\mathbf{B})^{\perp}$ to static solves is obviously zero, and thus not displayed in this figure.

$\mathbf{w}_t$ to prevent artifacts due to exaggeration of the geometric pattern of a blendshape. The L1-norm term encourages sparsity, as combining too many shapes often produces artifacts due to conflicting deformation patterns. In the following, we label Regularized Static Blendshapes Solve (RSBS) the solution of the minimization problem in equation (19). SBS will refer to its unregularized version ($\alpha_1 = \alpha_2 = 0$). Results obtained with the blendforces method (algorithm 1) will be labeled BF.

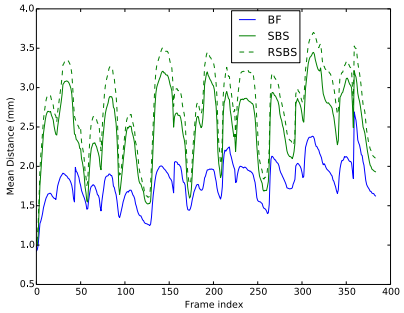
Data constraint fitting error We begin by evaluating the expressive range of blendforces by measuring how well it recovers the large-scale facial configurations spanned by marker data constraints. All solves used artist-crafted blendshapes that match the performer's morphology and expressions. Figure 2 compares marker fitting errors for SBS, RSBS, and BF on a sequence of motion capture marker data. The consistent lower fitting error achieved by BF confirms its ability to reach configurations outside the column space of $\mathbf{B}$. Figure 3 highlights this property by comparing for each frame the magnitudes of the projections of the deformation in $C(\mathbf{B})$ and its orthogonal complement $C(\mathbf{B})^{\perp}$.

Besides the additional geometric components from $C(\mathbf{B})^{\perp}$, we observe in all our experiments that, compared to (R)SBS, BF exhibits significantly higher contributions from geometric components within $C(\mathbf{B})$. This is observable in figure 3 and resonates with the interpretation of blendforces we presented in section 3.1: blendforces use blendshapes in $\mathbf{B}$ as a linear support for system steering, yet the physical simulation at its core maintains the mesh on a manifold of physically-plausible geometries. In that regard, the system behaves as a physical mesh regularizer in $C(\mathbf{B})^{\perp}$, compensating for some geometric artifacts created by blendshapes combination, and allowing them to take larger magnitudes.

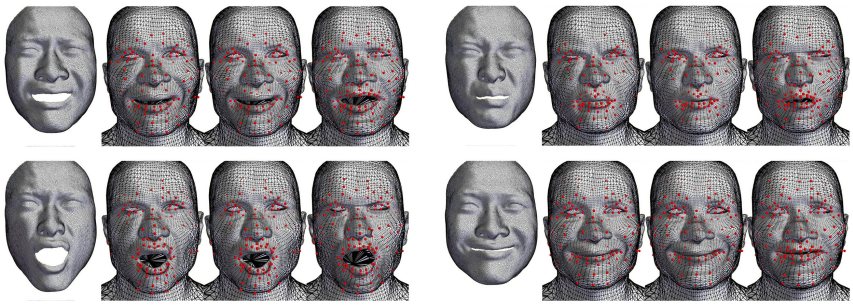
Dense ground-truth error A lower marker fitting error itself does not prove that BF produces valid skin movements outside the sparse set of marker points. We therefore evaluate the accuracy of its skin motion reconstruction on ground-truth data provided by dense facial motion capture data techniques [BHB*11, ZSCS04]. In this case we adapt an existing blendshape model to the morphology of the performer using deformation transfer [SP04]. As with previous experiment, we use a sparse set of points from this dense data as constraints for (R)SBS and BF solving (see figure 4), but this time we also evaluate the error on a *dense* set of vertices all over the face. The results presented in figure 7 suggest that even our relatively modest set of physical forces provides a more accurate and realistic reconstruction of skin deformation than (R)SBS.

Animation retargeting The experiments above were focused on animating a performer-specific facial mesh in motion capture setups. It is however common to want to animate a character who is morphologically, and sometimes even topologically different from the source geometry. Another similar scenario is transferring an existing animation -blendshapes-based or otherwise- to a new character. One common solution is to solve for blendshape weights on the source geometry, and transfer those weights to a target blendshapes rig. This however requires parallel blendshape models -one for the source and one for the target mesh-, costly assets created by specialized artists. Additionally, as the animation transfer occurs in blendshape parameter space the relation of the resulting animation to the captured marker trajectories is lost. An alternative approach consists of warping the facial marker positions to match the proportions of the target character's face, and solve for target blendshapes weights on those adapted marker trajectories. In addition to requiring only one blendshape model, this approach enables solving directly with target-specific priors, that more accurately define the expressive range of the target model [SLS*12].

The blendforces framework can be beneficial in such retarget-



(a) Mean Euclidean distance on the sparse set of vertices acting as marker constraints.



(b) Snapshots of the produced animation. *(left)*: SBS. *(middle)*: RSBS. *(right)*: BF. Red dots indicate the position of data constraints.

Figure 4: Comparison of animation results for (R)SBS and BF solvers. Data courtesy of [ZSCS04]. A sparse set of points was selected from the high-quality dense motion capture stream to act as data constraints. The blendshapes used for solving were adapted from a different character using deformation transfer [SP04]. RSBS produces more pleasing visuals compared to its unregularized counterpart, at the expense of reduced data fitting accuracy. The expressive range of the BF method shows a more accurate fitting of data constraints, while displaying pleasing animation results.

ing scenarios. Animation for a target character can be generated by warping a sparse set of vertices from a source animation, using the method of [SLS*12] to account for differences in facial proportions, and use them as data constraint to run a blendforces solve (algorithm 1). An advantage of using blendforces in such a retargeting scenario is that it defines a *target-specific* physical simulation framework, that is more akin to produce physically plausible skin motion and other physical interactions for this character. Figure 5 shows that the obtained motion for vertices of the target skin during muscle relaxation are different than those obtained from a static blendshapes solve, and look closer to natural skin motion. A dynamic example can be found in the accompanying video, showing varying behaviours upon muscle activation and relaxation [KGPG96]. As an additional example of physically-based animation enhancement, we illustrate in the following paragraph how the blendforces framework allows to correct mesh self-intersection by simulating collision and contact response in the target's physical system.

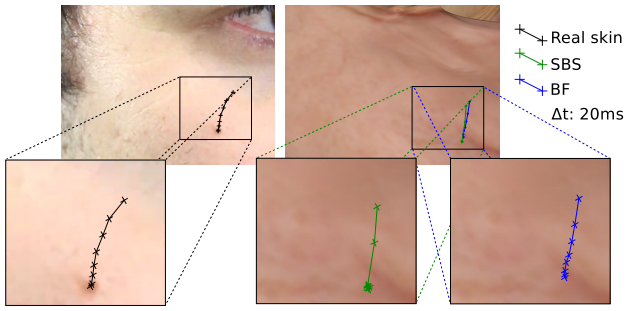


Figure 5: Trajectories of a skin point when relaxing a muscle. While the SBS animation (green) returns to a neutral position in 40ms, BF (blue) produces a slower and smoother trajectory, comparable to trajectories observable on real-world data (left image). This effects is best seen dynamically in the accompanying video.

Mesh self-intersection Self intersections of geometry are one of the noticeable degeneracies that can occur when using blendshapes. For faces, self-intersections are often visible in the lip region, where it is difficult to capture accurate movements on performers; this is particularly true in a retargeting scenario, as an unproblematic animation on one morphology can result in self-intersections on another. While blendshape models contain shapes encoding the geometric behavior of the mouth when the lips are pressed against each other, they offer no solution to handle self-intersections when they occur in solving due to noisy capture data or morphology changes. Conversely, the physical framework of blendforces enables simulating physical phenomena that are not accounted for by blendshapes data alone. The collision and contact response forces presented in 3.2 prevent self-intersections, and emulate a plausible physical reaction of the mesh to lip compression. Illustrations are presented on figures 6 and 8. It is worth noting that in the case of lip collision blendforces produces shapes outside the affine space of blendshapes (see figure 9). This can be interpreted as temporarily enhancing the blendshape set with new shapes specially adapted to the collision scenario. To the best of our knowledge, this is the first work to address lips self intersection in a blendshapes framework.

5. Current Limitations and Future Work

Blendforces solve provides more accuracy in the case of clean marker data, yet it can lead to implausible facial animation in the presence of noise caused by bad camera calibration or motion capture inaccuracies. While static blendshapes solving is sensitive to noise that lies in $C(\mathbf{B})$ but filters out the part that lies outside of it, the effect on a blendforces solve is less straightforward, as blendforces impulses, influenced by noisy data, are propagated to the physical system through time. Experiments show that more unwanted facial deformations due to noise could occur, if they are consistent with the physical system's capabilities.

Even though our approach extends the expressivity of the blendshapes framework, it cannot reconstruct a specific expression if the

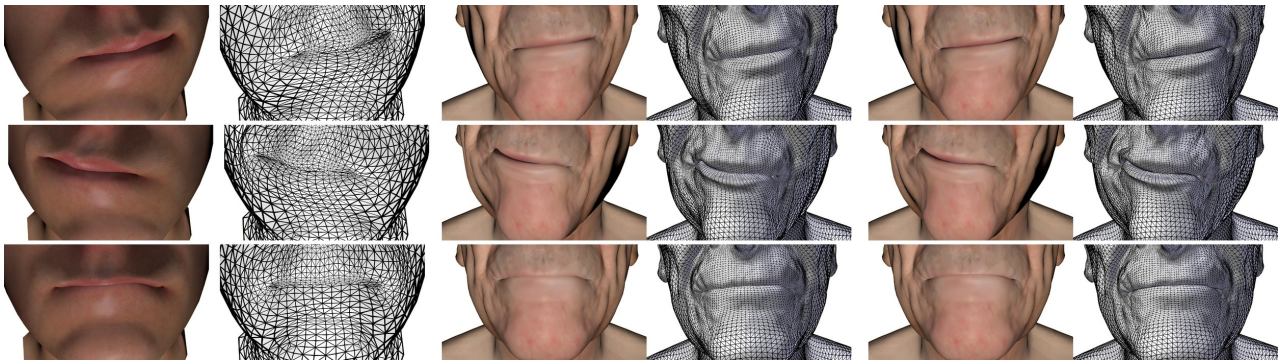


Figure 6: Handling of mesh self-intersections in animation retargeted from another character. Differences in character's facial morphologies result in physical inconsistencies for the target character, such as unnatural lip intersection. Blendforces unrolls a *target-specific* physical simulation, which enables to simulate plausible physical behavior specifically for the vertices of target mesh. This effect is best viewed on the accompanying video. *(left):* source animation *(center):* SBS *(right):* BF.

corresponding combination of blendshapes is lacking. Note that traditional blendshape approaches suffer from this limitation as well.

The physical framework of blendforces requires defining values for the physical forces' stiffnesses. We propose an optimization-based method to determine the vertex and edge stiffnesses, yet this method only takes static equilibrium into account. It could be interesting to enhance our method with additional constraints taking face dynamics into account.

The simplicity of our current physical system limits the ability to produce more complex elements of skin dynamics, such as wrinkles. We note however that its possibilities have not been fully explored. A finer resolution of the mesh, as well as a more elaborate distribution of force stiffnesses in wrinkle-prone areas of the face might generate wrinkles when the skin locally compresses.

From a geometric perspective, exploring the complementary roles of blendshape models and blendforces solving looks promising. Given a set of blendshapes, blendforces produces an animation closer to ground-truth measurements (figure 7) than linear blendshapes combinations, hinting at a better approximation of the face manifold away from the blendshapes. An interesting workflow would be to alternate blendforces solves with a given set of blendshapes, then refining this set with relevant shapes generated by the blendforces physical simulation. Techniques such as [NVW*13] could offer a systematic way to identify relevant blendforces configurations to be added to the blendshape model.

Another interesting enhancement for blendforces would be to complement 3D data constraints with depth- or image-based constraints. Those have been recently used at the core of impressive depth-based and video-based animation systems [LYYB13, BWP13, CWLZ13]. This would provide more dense data to fit the blendforces system to, as well as the ability to use more elaborate constraints such as contours [BGY*13].

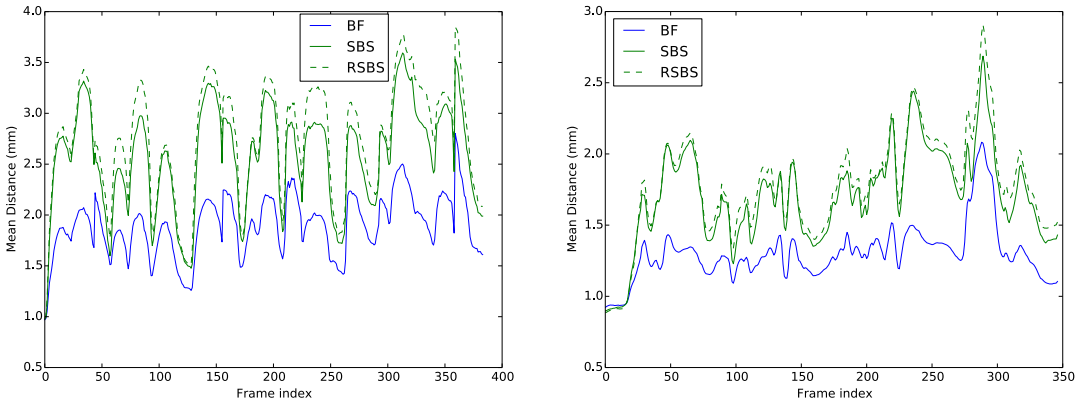
Despite using an efficient implicit time integration framework for physical simulation (appendix A), this study has not focused on computational performance. Our current unoptimized implementation takes about 200ms per frame for a blendforces solve on a 10K vertices mesh, about twice as much as a simple blendshape solve with the same software tools and assets. The Projective Dynamics solver we use (appendix A) is easily parallelizable, which provides a straightforward way to increase our performance. Our current implementation requires recomputing the system's Cholesky factorization upon collision, however recent work [Wan15] has shown that Projective Dynamics could be accelerated with iterative schemes, removing this requirement. This should make a realtime implementation of blendforces feasible in future work.

6. Conclusion

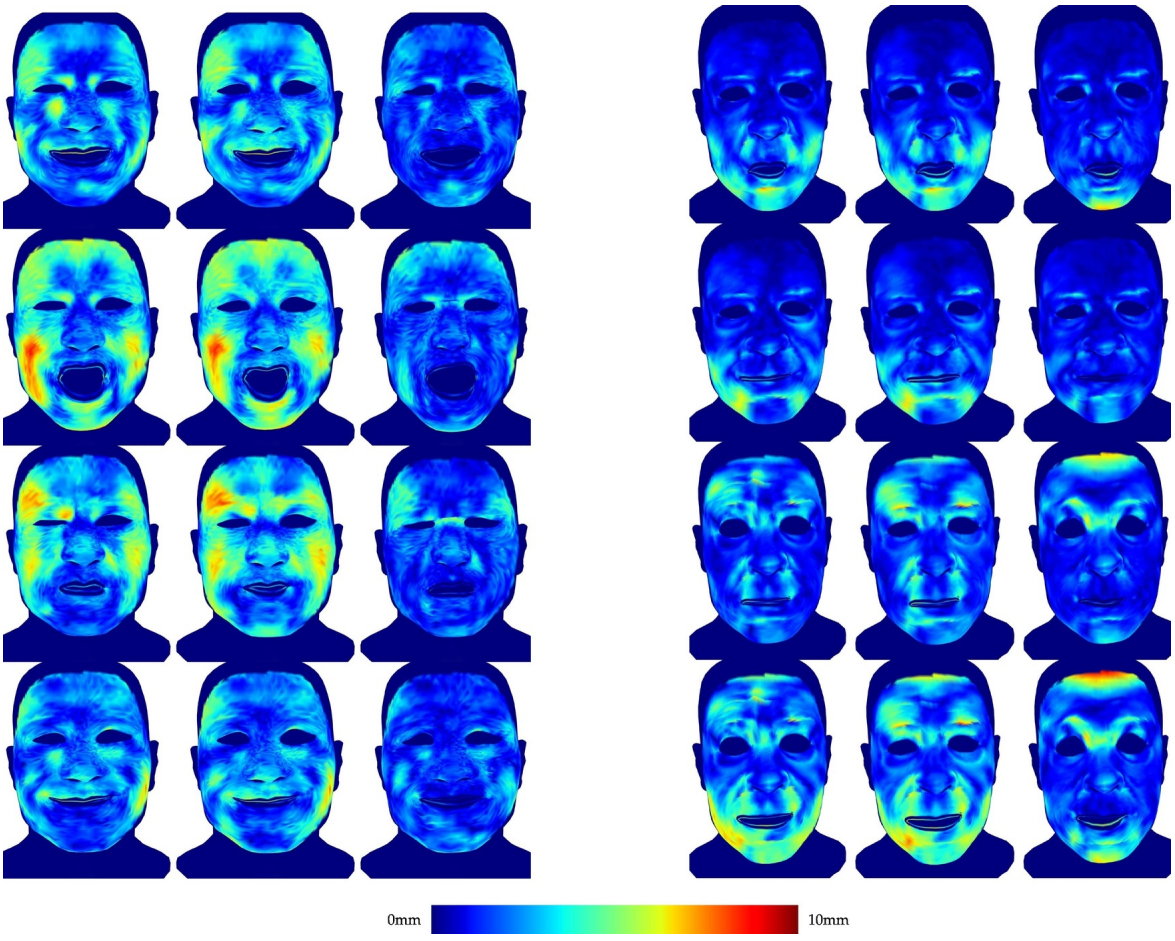
Despite their intuitive formulation, blendshape frameworks require labor intensive shape-editing processes to produce faithful facial animations. Their global and linear nature also limits their ability to model the complex, nonlinear and dynamic nature of facial deformations. In this study we have proposed *blendforces*, a paradigm that elegantly encapsulates the complexity of facial movements within a *physical* simulation framework by using blendshapes as a *data-based* formulation of actuation forces. Blendforces therefore have the ability to escape the affine subspace of the original blendshapes formulation, while remaining physically coherent.

We have applied blendforces to performance-driven facial animation, and produced physically-based animations constrained by motion capture markers. We have demonstrated the effectiveness of our system over blendshapes fitting both in terms of marker fitting accuracy and with respect to dense ground-truth skin motion measurements. The qualities of our framework have also been illustrated for retargeting scenarios, where its built-in ability to deal with lips self-intersection prevents unnatural results. In addition to the quantitative evaluation, our accompanying video shows animation results where the blendforces system exhibits natural-looking skin deformations, contrasting with the tendency of blendshape models to move vertices in noticeable linear patterns.

We believe that our framework bridges a gap between blendshapes-based animation and physically-based animation, and hope it will encourage further research towards artist-oriented physical systems for high-quality facial animation.

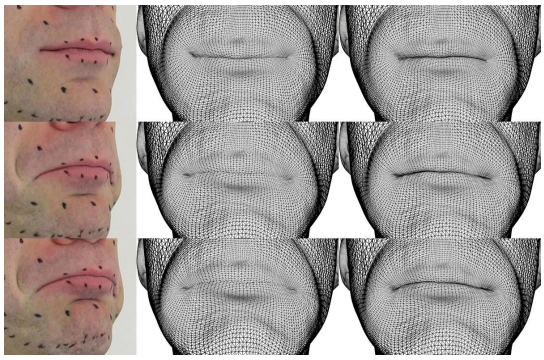


(a) Mean Euclidean distance on a *dense* set of ground-truth face vertices.

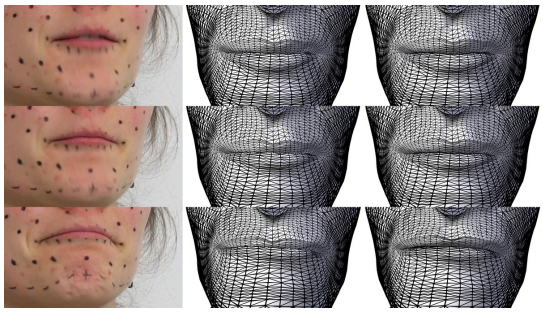


(b) Distribution of the ground-truth error. *(left):* SBS *(middle):* RSBS. *(right):* BF.

Figure 7: Comparison of ground-truth error distribution on the face for static blendshape solving and blendforces. Data courtesy of [ZSCS04] (left) and [BHB*11] (right). A dense set of motion capture points were used as ground-truth, with only a reduced sparse set acting as data constraint Figure 7a displays a consistently reduced mean ground-truth error showing that blendforces reconstruct more accurate skin movements. Figure 7b reveals that a large part of blendforces' error occurs at boundaries, because fixed vertices were imposed (Dirichlet conditions). The inner part of the face shows a significantly lower ground-truth error than (R)SBS solves. Mean distances over the whole [BHB*11] sequence for SBS, RSBS, and BF are 1.71mm, 1.78mm and 1.30mm respectively. Mean distances over the whole [ZSCS04] sequence for SBS, RSBS, and BF are 2.59mm, 2.76mm and 1.90mm respectively.



(a) Handling self-intersections on motion capture-based animation with artist-crafted blendshapes matching the performer's morphology



(b) Handling self-intersections on motion capture-based animation with markers trajectories warped to a different morphology.

Figure 8: Handling of mesh self-intersections. The contact response is necessary for the solve to produce the correct lip shape. (R)SBS are purely data-based, and do not handle contacts and collisions, while blendforces (BF) simulates a plausible physical reaction of the mesh to lip compression. This effect is best viewed on the accompanying video. *(left):* SBS *(right):* BF.

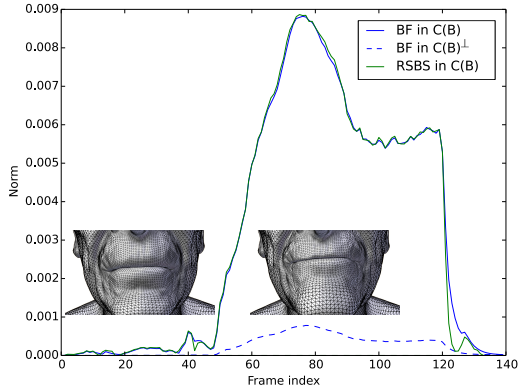


Figure 9: Measure of the projections of $\mathbf{x}$ in the respective subspaces $C(\mathbf{B})$ and $C(\mathbf{B})^{\perp}$ in a collision sequence. The blendforces solver escapes the affine subspace to create collision free shapes.

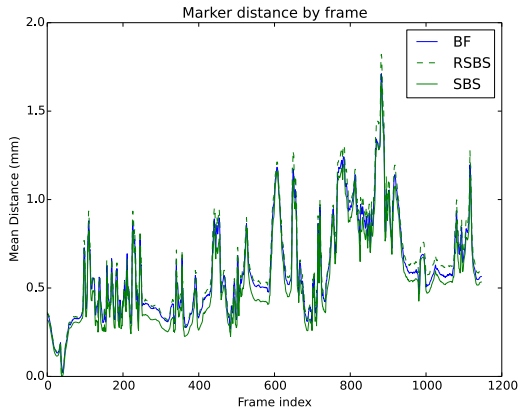


Figure 10: Solving on a transferred animation of speech and varied facial expressions.: mean Euclidean distance on a sparse set of vertices acting as marker constraints. The marker trajectories do not originate from true physical motion, but from a linear blendshapes animation. Quantitative measurements show that Blendforces is not as efficient in this context. The mean distances over the whole sequence for SBS, RSBS and BF are 0.54mm, 0.63mm and 0.60mm respectively.

Acknowledgements

We wish to thank our reviewers for their precious feedback. We also thank the Graphics and Imaging Laboratory of the University of Washington for sharing the data from [ZSCS04], as well as Disney Research for sharing the data from [BHB*11].

References

[BBA*07] BICKEL B., BOTSCH M., ANGST R., MATUSIK W., OTADUY M., PFISTER H., GROSS M.: Multi-scale Capture of Facial Geometry and Motion. In *ACM SIGGRAPH 2007 Papers* (New York, NY, USA, 2007), SIGGRAPH '07, ACM. 2

[BBB*14] BERMANO A. H., BRADLEY D., BEELER T., ZUND F., NOWROUZEZAHRAI D., BARAN I., SORKINE-HORNUNG O., PFISTER H., SUMNER R. W., BICKEL B., GROSS M.: Facial Performance Enhancement Using Dynamic Shape Space Analysis. *ACM Trans. Graph. 33*, 2 (Apr. 2014), 13:1–13:12. 2

[BFA02] BRIDSON R., FEDKIW R., ANDERSON J.: Robust Treatment of Collisions, Contact and Friction for Cloth Animation. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2002), SIGGRAPH '02, ACM, pp. 594–603. 4

[BGY*13] BHAT K. S., GOLDENTHAL R., YE Y., MALLET R., KOPERWAS M.: High fidelity facial animation capture and retargeting with contours. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2013), SCA '13, ACM, pp. 7–14. 8

[BHB*11] BEELER T., HAHN F., BRADLEY D., BICKEL B., BEARDSLEY P., GOTSMAN C., SUMNER R. W., GROSS M.: High-quality passive facial performance capture using anchor frames. *ACM Trans. Graph. 30* (August 2011), 75:1–75:10. 2, 6, 9, 10

[BML*14] BOUAZIZ S., MARTIN S., LIU T., KAVAN L., PAULY M.: Projective dynamics: fusing constraint projections for fast simulation. *ACM Transactions on Graphics (TOG) 33*, 4 (2014), 154. 5, 12

[BW98] BARAFF D., WITKIN A.: Large steps in cloth simulation. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (1998), ACM, pp. 43–54. 5, 12

[BWP13] BOUAZIZ S., WANG Y., PAULY M.: Online modeling for re-

altime facial animation. *ACM Transactions on Graphics 32*, 4 (2013). 2, 8

[CDHR08] CHEN Y., DAVIS T. A., HAGER W. W., RAJAMANICKAM S.: Algorithm 887: CHOLMOD, Supernodal Sparse Cholesky Factorization and Update/Downdate. *ACM Trans. Math. Softw. 35*, 3 (Oct. 2008), 22:1–22:14. 12

[CHZ14] CAO C., HOU Q., ZHOU K.: Displaced Dynamic Expression Regression for Real-time Facial Tracking and Animation. *ACM Trans. Graph. 33*, 4 (July 2014), 43:1–43:10. 2

[CWLZ13] CAO C., WENG Y., LIN S., ZHOU K.: 3d Shape Regression for Real-time Facial Animation. *ACM Trans. Graph. 32*, 4 (2013), 41:1–41:10. 2, 8

[EBDP96] ESSA I., BASU S., DARRELL T., PENTLAND A.: Modeling, Tracking and Interactive Animation of Faces and Heads Using Input from Video. In *Proceedings of the Computer Animation* (Washington, DC, USA, 1996), CA '96, IEEE Computer Society, pp. 68–. 3

[EF77] EKMAN P., FRIESEN W. V.: Facial action coding system. 1

[HMT*12] HAHN F., MARTIN S., THOMASZEWSKI B., SUMNER R., COROS S., GROSS M.: Rig-space Physics. *ACM Trans. Graph. 31*, 4 (2012), 72:1–72:8. 3

[HMYL15] HSIEH P.-L., MA C., YU J., LI H.: Unconstrained realtime facial performance capture. In *Computer Vision and Pattern Recognition (CVPR)* (2015). 2

[HTC*13] HAHN F., THOMASZEWSKI B., COROS S., SUMNER R. W., GROSS M.: Efficient Simulation of Secondary Motion in Rig-space. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2013), SCA '13, ACM, pp. 165–171. 3

[HVTG08] HARMON D., VOUGA E., TAMSTORF R., GRINSPUN E.: Robust treatment of simultaneous collisions. In *ACM Transactions on Graphics (TOG)* (2008), vol. 27, ACM, p. 23. 4

[KGC*96] KOCH R. M., GROSS M. H., CARLS F. R., VON BÃIJREN D. F., FANKHAUSER G., PARISH Y. I. H.: Simulating Facial Surgery Using Finite Element Models. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1996), SIGGRAPH '96, ACM, pp. 421–428. 3

[KGPG96] KEEVE E., GIROD S., PFEIFLE P., GIROD B.: Anatomy-based Facial Tissue Modeling Using the Finite Element Method. In *Proceedings of the 7th Conference on Visualization '96* (Los Alamitos, CA, USA, 1996), VIS '96, IEEE Computer Society Press, pp. 21–ff. 3, 7

[KMS11] KHOLGADE N., MATTHEWS I., SHEIKH Y.: Content retargeting using parameter-parallel facial layers. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2011), SCA '11, ACM, pp. 195–204. 2

[LAR*14] LEWIS J., ANJYO K., RHEE T., ZHANG M., PIGHIN F., DENG Z.: Practice and theory of blendshape facial models. In *Eurographics 2014-State of the Art Reports* (2014), The Eurographics Association, pp. 199–218. 1, 2, 3

[LBOK13] LIU T., BARGTEIL A. W., O'BRIEN J. F., KAVAN L.: Fast simulation of mass-spring systems. *ACM Transactions on Graphics (TOG) 32*, 6 (2013), 214. 4, 5, 12

[LWP10] LI H., WEISE T., PAULY M.: Example-based facial rigging. In *ACM SIGGRAPH 2010 papers* (New York, NY, USA, 2010), SIGGRAPH '10, ACM, pp. 32:1–32:6. 2

[LXB15] LI Y., XU H., BARBIČ J.: *Enriching Triangle Mesh Animations with Physically Based Simulation*. Tech. rep., University of Southern California, 2015. 3

[LYYB13] LI H., YU J., YE Y., BREGLER C.: Realtime facial animation with on-the-fly correctives. *ACM Transactions on Graphics 32*, 4 (2013). 2, 8

[MFD11] MA W.-C., FYFFE G., DEBEVEC P.: Optimized Local Blendshape Mapping for Facial Motion Retargeting. In *ACM SIGGRAPH 2011 Talks* (New York, NY, USA, 2011), SIGGRAPH '11, ACM, pp. 58:1–58:1. 2

[MHHR07] MÜLLER M., HEIDELBERGER B., HENNIX M., RATCLIFF J.: Position Based Dynamics. *J. Vis. Comun. Image Represent. 18*, 2 (Apr. 2007), 109–118. 5

[MJC*08] MA W.-C., JONES A., CHIANG J.-Y., HAWKINS T., FREDERIKSEN S., PEERS P., VUKOVIC M., OUHYOUNG M., DEBEVEC P.: Facial Performance Synthesis Using Deformation-driven Polynomial Displacement Maps. In *ACM SIGGRAPH Asia 2008 Papers* (New York, NY, USA, 2008), SIGGRAPH Asia '08, ACM, pp. 121:1–121:10. 2

[MTGG11] MARTIN S., THOMASZEWSKI B., GRINSPUN E., GROSS M.: Example-based elastic materials. In *ACM Transactions on Graphics (TOG)* (2011), vol. 30, ACM, p. 72. 12

[MWF*11] MA W.-C., WANG Y.-H., FYFFE G., BARBIC J., CHEN B.-Y., DEBEVEC P.: A blendshape model that incorporates physical interaction. In *SIGGRAPH Asia* (Hong Kong, Dec. 2011). 2, 3

[NVW*13] NEUMANN T., VARANASI K., WENGER S., WACKER M., MAGNOR M., THEOBALT C.: Sparse localized deformation components. *ACM Transactions on Graphics (TOG) 32*, 6 (2013), 179. 8

[PB81] PLATT S. M., BADLER N. I.: Animating Facial Expressions. In *Proceedings of the 8th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1981), SIGGRAPH '81, ACM, pp. 245–252. 3

[PM01] PITERMANN M., MUNHALL K. G.: An inverse dynamics approach to face animation. *The Journal of the Acoustical Society of America 110*, 3 (2001), 1570–1580. 3

[SA07] SORKINE O., ALEXA M.: As-rigid-as-possible surface modeling. In *EUROGRAPHICS* (2007). 4

[SILN11] SEO J., IRVING G., LEWIS J., NOH J.: Compression and direct manipulation of complex blendshape models. In *ACM Transactions on Graphics (TOG)* (2011), vol. 30, ACM, p. 164. 1

[SLS*12] SEOL Y., LEWIS J., SEO J., CHOI B., ANJYO K., NOH J.: Spacetime expression cloning for blendshapes. *ACM Trans. Graph. 31*, 2 (2012), 14:1–14:12. 6, 7

[SNF05] SIFAKIS E., NEVEROV I., FEDKIW R.: Automatic determination of facial muscle activations from sparse motion capture marker data. In *ACM Transactions on Graphics (TOG)* (2005), vol. 24, ACM, pp. 417–425. 3

[SP04] SUMNER R. W., POPOVIĆ J.: Deformation transfer for triangle meshes. *ACM Transactions on Graphics (TOG) 23*, 3 (2004), 399–405. 2, 6, 7

[SSK*11] SEOL Y., SEO J., KIM P. H., LEWIS J., NOH J.: Artist friendly facial animation retargeting. *ACM Transactions on Graphics (TOG) 30*, 6 (2011), 162. 2

[SSK*12] SEOL Y., SEO J., KIM P. H., LEWIS J. P., NOH J.: Weighted Pose Space Editing for Facial Animation. *Vis. Comput. 28*, 3 (2012), 319–327. 2

[TDlTM11] TENA J. R., DE LA TORRE F., MATTHEWS I.: Interactive region-based linear 3d face models. *ACM Transactions on Graphics 30*, 4 (Aug. 2011). 2

[THM*03] TESCHNER M., HEIDELBERGER B., MUELLER M., POMERANETS D., GROSS M.: Optimized spatial hashing for collision detection of deformable objects. *Proceedings of Vision, Modeling, Visualization (VMV 2003)* (2003), 47–54. 4

[TW93] TERZOPOULOS D., WATERS K.: Analysis and synthesis of facial image sequences using physical and anatomical models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 15*, 6 (1993), 569–579. 3, 4

[Wan15] WANG H.: A Chebyshev Semi-iterative Approach for Accelerating Projective and Position-based Dynamics. *ACM Trans. Graph. 34*, 6 (Oct. 2015), 246:1–246:9. 8

[Wat87] WATERS K.: A Muscle Model for Animation Three-dimensional Facial Expression. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1987), SIGGRAPH '87, ACM, pp. 17–24. 3

[WLVG07] WEISE T., LEIBE B., VAN GOOL L.: Fast 3d scanning with automatic motion compensation. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on* (2007), IEEE, pp. 1–8. 2

[WMKG07] WARDETZKY M., MATHUR S., KÄLBERER F., GRINSPUN E.: Discrete laplace operators: no free lunch. In *EUROGRAPHICS* (2007). 4

[ZSCS04] ZHANG L., SEITZ S., CURLESS B., SNAVELY N.: Space-time faces: High resolution capture for modeling and animation. *ACM transactions on graphics*, 3 (2004), 546–556. 6, 7, 9, 10

Appendix A: Projective Dynamics Blendforces Solver

As presented in section 3.3, the blendforces solver relies on implicit time integration to simulate the dynamics of facial deformations. For a deformable object, using implicit time integration to predict its state can result in solving a nonlinear optimization problem, as a number of forces relevant to elastic behavior are expressed as a nonlinear function of that state.

Classical approaches solve the problem through iterative algorithms using Newton's method, which amount to successive locally valid linearizations of the force's expression through Taylor expansion [BW98,MTGG11]. Those approaches are slow, as they require reestimating the affine approximation and inverting the Hessian of the system at each iteration.

Recent work on simulation of implicit Euler integration have formalized elegant and efficient solutions to simulate physical systems featuring nonlinear energy potentials [LBOK13, BML*14]. They observe that the potential of an elastic force can be interpreted as a distance of the mesh's current configuration to the closest undeformed configuration, for which the potential is zero. They formulate the potential $W(\mathbf{x})$ of a conservative force as

$$W(\mathbf{x},\mathbf{p}) = \min_{\mathbf{p}} d(\mathbf{x}+\mathbf{p}) + \delta_m(\mathbf{p}) \tag{20}$$

where d is a measure of distance between $\mathbf{x}$ and $\mathbf{p}$, and δ_m is a function that evaluates to zero if $\mathbf{p}$ belongs to the manifold of undeformed configurations and $+\infty$ otherwise. In that perspective, the vector $\mathbf{p}$ where (20) is minimized is interpreted as the projection of $\mathbf{x}$ on the nonlinear rest manifold of undeformed configurations.

Based on this formulation, *Projective Dynamics* solvers are able to simulate the behavior of complex physical systems by alternating two steps [BML*14]: finding the projections $\mathbf{p}_i$ of $\mathbf{x}$ on the rest manifolds for all system forces, and solving the implicit time integration problem with a simpler formulation of the potentials $W_i(\mathbf{x}) = d(\mathbf{x}+\mathbf{p}_i)$ with $\mathbf{p}_i$ known.

In this work we take advantage of the geometric interpretation of nonlinear potentials introduced by Projective Dynamics. Using a simple quadratic distance measure for d we approximate the potential of each system force i as

$$\tilde{W}_i(\mathbf{x},\mathbf{p}_i) = \frac{k}{2}\|\mathbf{F}\mathbf{x} - \mathbf{G}\mathbf{p}_i\|^2 + \delta_{m_i}(\mathbf{p}_i) \tag{21}$$

Following the two-step strategy used in [BML*14], we first determine $\mathbf{p}_i$ through nonlinear projection on the rest manifold of system force i, then we express the force as

$$\tilde{\mathbf{f}}^{int,i}(\mathbf{x},\mathbf{p}_i) = -\nabla\tilde{W}_i(\mathbf{x}) = -k_i(\mathbf{F}_i^T\mathbf{F}_i\mathbf{x} - \mathbf{F}_i^T\mathbf{G}_i\mathbf{p}_i) \tag{22}$$

With $\mathbf{p}_i$ fixed, equation (22) forms an affine approximation of the contribution of system force i, which allows us to solve the blendforces optimization problem of section 3.3. To be consistent with the notations of equation (14), we write $\mathbf{A}^i = -k_i\mathbf{F}_i^T\mathbf{F}_i$ and $\mathbf{b}^i = k_i\mathbf{F}_i^T\mathbf{G}_i\mathbf{p}_i$.

An important advantage of the Projective Dynamics approach over Newton's method is that for the considered system forces (section 3.2) the quantities $\mathbf{F}_i$, $\mathbf{G}_i$ and $\mathbf{k}_i$ involved in the affine approximation of the forces are constant when working with a fixed topology of the facial mesh [BML*14]. This allows for efficient prefactorization of the term $(\mathbf{I} - h^2\mathbf{M}^{-1}\sum_i \mathbf{A}_i^i)$ in equations (16) and (17) using sparse Cholesky factorization [CDHR08]. Thus the terms of equation (18) can be computed efficiently. Assuming the markers are visible through the whole sequence (constant $\mathbf{S}$), the term $(\mathbf{S}\Phi_t)^+$ is constant and can also be precomputed.

Like similar physical systems [LBOK13], we implement damping by filtering velocities, replacing $h\dot{\mathbf{x}}_{t-1}$ by $\alpha h\dot{\mathbf{x}}_{t-1}$ in equation 15, with $\alpha \in [0,1]$.

Appendix B: Stiffnesses Determination

In section 3.2 we derived our physical system stiffnesses by enforcing the equilibrium of forces on each of the expressions specified by the blendshape matrix $\mathbf{B}$. For a column $\mathbf{B}[:,j]$ of the blendshape matrix, the equilibrium forces can be formulated using the Projective Dynamics approximation of the internal conservative forces of equation (22):

$$\min_{\mathbf{k}} \sum_j ||\mathbf{B}[:,j] - \sum_i k_i(\mathbf{F}_i^T\mathbf{F}_i(\mathbf{x}_0 + \mathbf{B}[:,j]) - \mathbf{F}_i\mathbf{G}_i\mathbf{p}_i^j)||^2 \tag{23}$$

where $\mathbf{k}$ stores the stiffnesses distributions for all vertices and edges and $\mathbf{p}_i^j$ is the projection of the configuration $\mathbf{x}_0 + \mathbf{B}[:,j]$ on the rest manifold (see appendix A). Thanks to the Projective Dynamics approximation the optimization term is now linear in $\mathbf{k}$, and we can write it as the following linear system:

$$\min_{\mathbf{k}} ||\mathbf{H}\mathbf{k} + \mathbf{f}||^2 \tag{24}$$

where $\mathbf{f}$ is a vector built by stacking all columns of $\mathbf{B}$ and matrix $\mathbf{H}$ is a sparse matrix mapping the stiffnesses to the linearized elastic forces values at the respective configurations $\mathbf{x}_0 + \mathbf{B}[:,j]$.

In practice we seek for solutions of equation (24) under the constraint $\mathbf{k} > 0$. We also penalize very small stiffness values by adding a prior term to the optimization function:

$$\min_{\mathbf{k}>0} ||\mathbf{H}\mathbf{k} + \mathbf{f}||^2 + \lambda\sum_i \frac{1}{k_i^2} \tag{25}$$

This non-linear optimization problem is solved using the Gauss-Newton algorithm with a projected gradient scheme to keep the stiffnesses positive. We take advantage of the sparsity of the system to precompute a symbolic sparse Cholesky factorization of the Jacobian [CDHR08], and update its numerical values at each Gauss-Newton iteration. We find experimentally that 10 iterations are sufficient, which solves the system in about 3 minutes for a mesh with 60K vertices. Note that this optimization is carried out only once per character.