ANNÉE 2017

CentraleSupélec

UNIVERSITE BRETAGNE LOIRE

N° d'ordre : 2017-12-TH

THÈSE / CENTRALESUPÉLEC
sous le sceau de l'Université Bretagne Loire

pour le grade de

DOCTEUR DE CENTRALESUPÉLEC

Mention : Signal, Image, Vision

École doctorale 601 « Mathématiques et Sciences et Technologies de l'Information et de la Communication (MathSTIC) »

présentée par

Vincent Barrielle

Préparée à l'UMR 6164 - IETR (Équipe FAST)
Institut d'Électronique et de Télécommunications de Rennes

Leveraging Blendshapes for Realtime Physics-Based Facial Animation

Thèse soutenue à CentraleSupélec le 24/11/2017

devant le jury composé de :

Gérard BAILLY
Directeur de recherche au CNRS / *rapporteur*

Bruno LÉVY
Directeur de recherche à l'INRIA / *rapporteur*

Luce MORIN
Professeur à l'INSA Rennes / *présidente du jury*

Florence BERTAILS-DESCOUBES
Chargée de recherche à l'INRIA / *examinatrice*

Renaud SÉGUIER
Professeur à CentraleSupélec / *directeur de thèse*

Nicolas STOIBER
Directeur Technique à Dynamixyz / *encadrant*

Contents

Remerciements

Je tiens à remercier tous ceux qui m'ont permis de réaliser cette thèse. Merci à mon directeur de thèse, Renaud Séguier, d'avoir encadré cette thèse et permis son déroulement serein. Cette thèse n'aurait pas pu avoir lieu sans le soutien de mon employeur, Gaspard Breton, et son appréciation de la recherche, qui m'ont permis d'effectuer mes recherches sans sombrer sous les impératifs de court-terme auxquels Dynamixyz peut être soumise. J'ai eu le grand plaisir d'être encadré par et de travailler en étroite collaboration avec Nicolas Stoiber, et je le remercie pour la grande qualité de nos discussions, qu'elles soient scientifiques ou autres. Cédric Cagnart est à l'origine de l'idée derrière blendforces, et je le remercie d'être venu parler de cette idée à Nicolas, ça a été un plaisir de travailler ensemble pour Eurographics.

Merci aux rapporteurs de cette thèse, Gérard Bailly et Bruno Lévy, ainsi qu'aux examinatrices, Luce Morin et Florence Bertails-Descoubes, pour toutes les questions et les discussions très enrichissantes qui en ont découlé lors de la soutenance.

Travailler au sein de Dynamixyz est un plaisir quotidien, merci à tous les collègues pour la bonne ambiance. Merci à Carole Garnier avec qui j'ai beaucoup apprécié de travailler lorsqu'elle était dans l'entreprise. C'est toujours un plaisir de discuter avec Olivier Aubault qui s'intéresse toujours à nombre de sujets. Merci à Maxime Thomas le Déoré dont l'aide m'aura été précieuse pour bien cerner les différentes techniques de rigging. Sans ordre particulier, merci à Sandrine Lhermitte, Artemisa Ahmeti, Louis-Paul Cordier, Flora Jullien, Guillaume Nicolas, Florian Dussable, c'est toujours un plaisir de vous retrouver chaque jour pour travailler, rire, discuter. L'équipe R&D de Dynamixyz s'est agrandie, bienvenue à Marc Tournadre et Éloise Berson, il semble qu'une belle période de recherches intéressantes s'ouvre.

Merci à tout les membres du personnel de CentraleSupélec à qui j'ai pu avoir affaire, j'ai toujours pu compter sur votre compétence. Un merci tout particulier à Karine Bernard sans qui je n'aurais pas sû affronter les nombreuses démarches administratives de la thèse. Ça a été un plaisir de parler avec tous les membres de l'équipe FAST, en particulier Catherine Soladié, Amine Kacete ou Raphaël Weber, que je remercie également pour la collaboration très intéressante sur l'analyse d'émotions.

Merci également à tous mes amis (notamment Alex, Simon, Vasco, Eska, Pablo, les triplés de Belleville, Geromino, mais aussi beaucoup d'autres), c'est un plaisir de vous connaître. Pour partie, ce sont les discussions passionées avec vous qui m'ont donné l'envie de la recherche.

Merci à ma famille de m'avoir toujours soutenu pour en arriver là. Mes parents ont patiemment répondu aux questions d'un enfant très curieux, et m'ont ainsi donné le goût des sciences. Ma sœur Laureline a toujours été d'un grand soutien en toute occasion, et c'est un plaisir de savoir qu'elle assistera à ma soutenance de thèse trois ans après que j'aie assisté à la sienne. Merci aussi à ma belle famille, Georges-Henri, Laurence et Silouane, qui m'ont aussi soutenu durant cette thèse. Ma femme Eugénie est mon alliée au quotidien, toujours source de réconfort lorsque j'en ressens le besoin. Pour toute son aide et le plaisir de vivre avec elle chaque jour, je la remercie énormément. Notre fils Baptiste n'est pas

encore en âge de lire ces lignes, mais j'espère qu'il aura lui aussi l'occasion de s'épanouir et d'aller au bout de ses passions. Eugénie, Baptiste, je pense fort à vous en terminant cette thèse, merci pour l'énergie que vous me transmettez chaque jour.

Résumé en français

0.1 Contexte et motivation

Au cours de la dernière décennie, la production de médias basés sur les images de synthèse a décuplé, avec la montée en puissance des effets spéciaux cinématographiques, des jeux vidéos et plus récemment de la réalité virtuelle. Ces médias nécessitent la production de nombreuses heures d'animation faciale de synthèse pour donner vie aux personnages qui peuplent les mondes virtuels. La qualité de ces animations faciales est cruciale pour le succès de ces médias. Cependant, produire des animations faciales de synthèse convaincantes n'est pas une tâche aisée, et malgré de nombreuses années de recherche sur le sujet, il demeure ardu de créer des animations faciales de synthèse suffisamment réalistes pour duper un observateur et lui faire croire qu'il observe une animation naturelle. En effet, les êtres humains communiquent au quotiden par le biais de leurs expressions faciales, et sont ainsi naturellement doués pour interpréter le moindre détail. En conséquence, chaque erreur dans la synthèse de l'animation faciale est immédiatement répérée.

Avec le perfectionnement continu des techniques de capture de mouvement, il devient possible de capturer l'expression d'acteurs dans ses moindres détails, suivant image par image le mouvement de chaque pore de la peau, de chaque ride. Toutefois, une fois la performance capturée transférée à un personnage de synthèse, la subtilité de ces mouvements est généralement perdue. En effet, le standard industriel pour encoder l'animation des visages de synthèse, les *blendshapes*, ne peut représenter pleinement la complexité des expressions faciales [98]. Le formalisme des blendshapes encode les déformations faciales comme la combinaison linéaire d'un certain nombre d'expressions faciales basiques représentant les activations des différents muscles faciaux (figure 1). Les blendshapes forment donc un modèle affine des déformations faciales, et ne peuvent de ce fait représenter précisémment les déformations de la peau, qui sont non-linéaires.

L'utilisation de la simulation physique permet de générer des animations faciales de synthèse de grande qualité [158], mais la construction du système physique représentant le visage est une tâche ardue nécessitant l'utilisation de matériel médical pour obtenir les paramètres physiques de la peau, et de nombreuses heures de travail manuel pour convertir ces données en un modèle physique utilisable dans une simulation numérique.

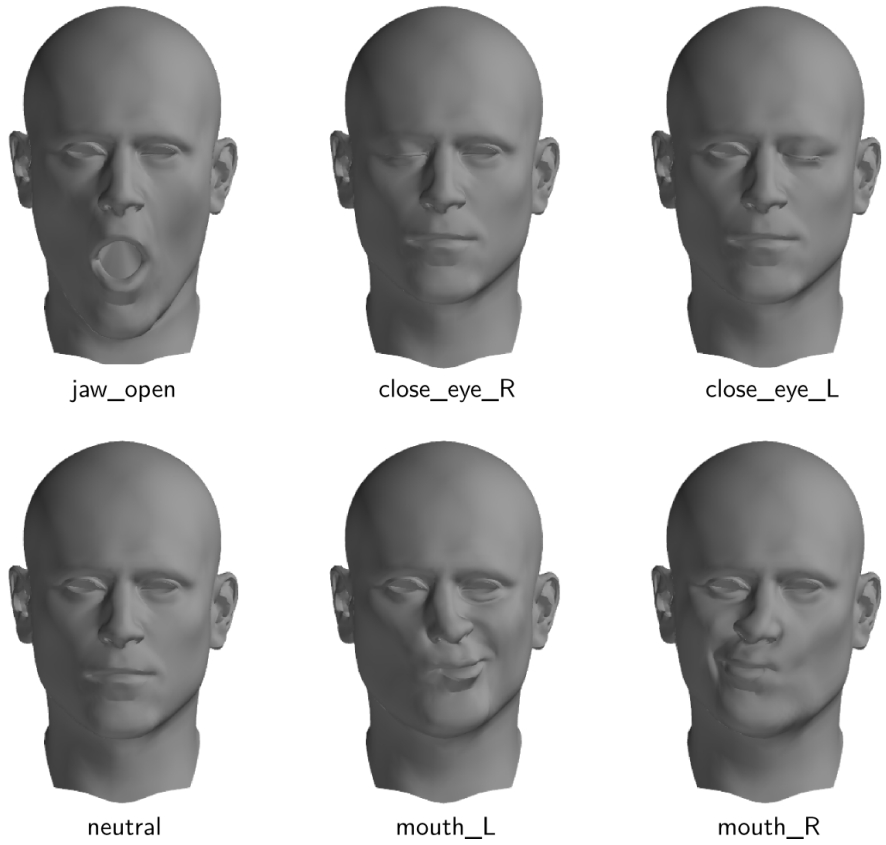


FIGURE 1 – Éléments d'un jeu de blendshapes. L'expression neutre, ainsi que l'activation unitaire de 5 blendshapes sont montrées.

Avec une demande d'animations faciales de synthèse de haute qualité toujours plus grande, ces méthodes de simulation physiques, qui peuvent se justifier pour des films à budget élevé, ne peuvent être adaptées pour des domaines aux moyens moins abondants. *A contrario*, la création de jeux de blendshapes est récemment devenue extrèmement commode, avec l'apparition de méthodes nécessitant uniquement un téléphone portable pour créer des blendshapes correspondant au visage d'une personne [75, 35]. Il n'a donc jamais été aussi simple de produire de l'animation faciale de synthèse en utilisant des méthodes basées sur les blendshapes, mais la qualité des animations ainsi produites est réduite du fait des limitations inhérentes à ce formalisme.

Nous proposons de tirer parti de l'ubiquité des personnages de synthèse dôtés de jeux de blendshapes faciales pour construire des modèles de simulation physique à partir de ces blendshapes, et de contrôler ce modèle de simulation physique à partir de données de capture de mouvements. Étant basée sur des jeux de blendshapes pré-existants, notre méthode est facilement applicable à de nombreux visages de synthèse. L'utilisation de

simulation physique permet par ailleurs d'augmenter le réalisme des animations faciales réalisées. En particulier, la simulation physique permet d'obtenir des mouvements de peau aux dynamiques réalistes, de gérer le contact entre les lèvres, et de simuler le phénomène des lèvres collantes.

Grâce au développement récent de méthodes de capture de mouvements faciaux en temps réel à l'aide de blendshapes [26, 105, 32], l'animation de visages synthétiques a trouvé de nouvelles applications dans la téléprésence ou la conférence vidéo par avatars interposés. Nous nous proposons donc de développer notre système d'animation faciale avec les objectifs de permettre une simulation physique temps-réel, ainsi que de pouvoir contrôler notre système par le biais d'une capture de mouvements utilisant une seule caméra.

L'objectif de cette thèse est donc de concevoir un système d'animation faciale temps-réel, dirigé par la capture de mouvements, qui combine les avantages du formalisme blendshapes avec ceux de la simulation physique. Plus précisémment, nos objectifs sont :

- **Basé sur les blendshapes.** Nous souhaitons construire un système physique d'animation faciale en utilisant des personnages virtuels déjà dôtés de jeux de blendshapes, profitant ainsi de leur ubiquité.
- **Basé sur la physique.** Nous voulons produire des animations faciales de synthèse présentant des caractéristiques qui ne peuvent être atteintes sans simulation physique.
- **Pratique.** Notre système se doit de présenter une mise en oeuvre facile, ne nécessitant qu'un minimum de matériel et de calibration.
- **Temps-réel.** Notre système doit être suffisamment efficace pour permettre une utilisation en temps réel. En particulier, des animations dirigées en temps réel par de la capture de mouvement doivent être possibles.

Dans la suite de ce résumé, nous commençons par présenter notre formalisme *blendforces*, qui permet d'interpréter les blendshapes comme des forces mettant en mouvement le système physique formé par le visage (section 0.2). La section 0.3 présente ensuite la construction du système physique représentant le visage. Enfin, nous verrons en section 0.4 comment nous parvenons à effectuer la simulation physique de mouvements faciaux en temps réel. Les principales contributions de cette thèse sont :

- la réinterprétation des blendshapes comme une base de forces permettant de réaliser une simulation physique des mouvements faciaux ;
- la conception d'un système physique de visage permettant une simulation réaliste des tissus faciaux ainsi que la prise en contact des forces impliquées dans les contacts des lèvres ;
- et l'amélioration des performances d'une méthode de simulation physique de l'état de l'art permettant la simulation physique des mouvements du visage en temps réel.

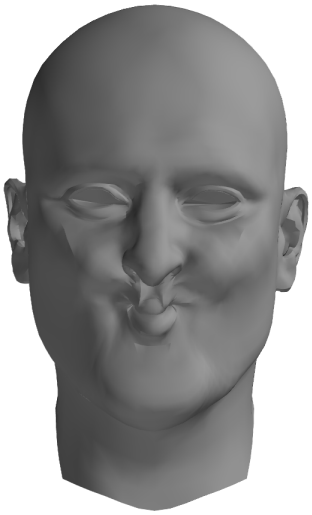


FIGURE 2 – Activation conflictuelle de deux blendshapes. Activer simultanément les blendshapes mouth_L et mouth_R crée une déformation non naturelle.

0.2 Unification des blendshapes et de la simulation physique

Nous proposons dans cette thèse d'étendre l'approche des blendshapes en la replaçant dans un contexte de simulation physique. En effet, la simplicité de mise en oeuvre du formalisme blendshapes a conduit à une grande popularité de la méthode. Ainsi, il est fréquent qu'un visage de synthèse soit représenté sous la forme de blendshapes. En basant notre méthode sur l'existence de blendshapes, nous profitons ainsi indirectement de cette popularité.

Dans cette section, nous présentons le formalisme *blendforces*, dans lequel nous réinterprétons les blendshapes comme les directions dans lesquelles l'action des muscles faciaux se produit, ce qui nous permet de réaliser des animations faciales de synthèse en se basant sur la simulation physique. Nous commençons par analyser les défauts du paradigme blendshapes (section 0.2.1), puis présentons le fonctionnement de la méthode blendforces (section 0.2.2). Enfin, en section 0.2.3, nous présentons une interprétation du formalisme blendforces permettant d'expliquer comment les animations générées dans ce formalisme sortent du sous-espace affine dans lequel sont confinées les blendshapes.

0.2.1 Limitations des blendshapes

Le formalisme blendshapes encode les mouvements faciaux comme l'interpolation linéaire de déformations unitaires correspondant aux résultats sur la peau de l'activation de chaque muscle facial pris en isolation, selon les principes du Facial Action Coding System [57]. Formellement, le visage est représenté par un maillage triangulaire, dont les coordonnées des vertex sont représentées par le vecteur $\mathbf{x} \in \mathbb{R}^{3N_v}$, où N_v est le nombre de vertex. Pour

une expression encodée par des blendshapes, on aura

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

où $\mathbf{x}_0$ contient les coordonnées des vertex pour l'expression neutre, $\mathbf{B}$ est une matrice dont les colonnes contiennent les déplacements des vertex correspondant à une expression unitaire, et où $\mathbf{x} \in [0,1]^{N_v}$ est un vecteur contenant les poids d'activation de chacune des blendshapes.

En pratique, en restreignant les blendshapes au seul ensemble des actions faciales unitaires, il est impossible d'obtenir des expressions faciales précises. En effet, certains mouvements faciaux, tels l'ouverture de la machoire, contiennent des mouvements de rotation, qui ne peuvent être représentés par l'activation linéaire d'une blendshape. Ce problème se manifeste également pour la représentation d'interactions complexes, telles que la compression des lèvres lorsqu'elles sont pressées l'une contre l'autre. La solution, dans le paradigme blendshapes, consiste à ajouter de nouvelles déformations à la matrice $\mathbf{B}$, et de considérer le modèle blendshapes comme une approximation affine par morceaux de la variété des expressions faciales. Ce paliatif présente cependant l'inconvénient de déporter la non linéarité sur la variété des poids $\mathbf{w}$ pour lesquels l'expression produite est valide. Comme illustré en figure 2, l'activation simultanée de certaines blendshapes crée des déformations non naturelles, et ce problème est d'autant plus présent que le nombre de blendshapes est grand.

D'un point de vue temporel, prendre la dérivée de l'équation 1 montre que la méthode blendshapes contraint les vitesses des vertex à appartenir à l'espace image $C(\mathbf{B})$ de la matrice des blendshapes. En conséquence, on peut observer que les vertex proches tendent à se déplacer simultanément, par blocs, selon les motifs encodés dans les blendshapes activés. Ainsi, les dynamiques causées par les activations des différents muscles faciaux sont généralement gommées par la transcription en blendshapes, et seuls les mouvements correspondant à des fréquences spatiales encodées dans les blendshapes sont préservés.

0.2.2 Animation faciale par blendforces

Nous proposons un nouveau paradigme étendant le formalisme des blendshapes, joignant le monde des approches orientées données auquel appartiennent les blendshapes avec celui de la simulation physique. Nous considérons que les mouvements faciaux sont un phénomène intrinsèquement dynamique, qui ne peut dès lors pas bien être représenté avec une approche statique comme celle des blendshapes. Par exemple, le phénomène des lèvres collantes ne peut être expliqué sans considérer l'historique des précédents mouvements : les lèvres ne collent entre elles que parce qu'elles sont rentrées précedemment en contact.

Dans notre paradigme, les coordonnées $\mathbf{x}$ des vertex du maillage facial sont soumises à la deuxième loi de Newton :

$$\mathbf{M}\ddot{\mathbf{x}} = \mathbf{f}_{\text{ext}} + \mathbf{f}_{\text{int}}, \tag{2}$$

où $\mathbf{M}$ est une matrice diagonale contenant les masses associées à chaque vertex, $\mathbf{f}_{\text{int}}$ représente la contribution des forces d'élasticité qui s'opposent au déformations du visage, et $\mathbf{f}_{\text{ext}}$ représente les forces externes qui s'appliquent au système, parmi lesquelles l'activation des

muscles faciaux. Nous introduisons une nouvelle formulation pour les forces représentant l'activation des muscles faciaux. Nous supposons en effet que ces forces appartiennent au sous-espace linéaire généré par la matrice $\mathbf{B}$:

$$\mathbf{f}_{\text{muscles}} = \mathbf{B}\mathbf{u}, \tag{3}$$

où le terme $\mathbf{B}\mathbf{u}$ représente les *blendforces*, l'encodage par les blendshapes des forces issues des muscles faciaux.

Ce formalisme blendforces peut-être vu comme une extension des blendshapes. En effet, en supposant un masse égale à 1 pour chaque vertex, et l'absence de forces autres que les blendforces, il est possible de simuler n'importe quelle animation blendshapes en choisissant $\mathbf{u} = \ddot{\mathbf{w}}$ et des conditions aux limites appropriées. Ainsi, nous voyons qu'un composant essentiel pour différencier les animations produites par le formalisme blendforces réside dans les forces internes qui s'opposent aux blendforces. Nous nous intéressons plus en détail à ces forces internes en section 0.3.

Nous nous intéressons à présent à la génération d'une animation blendforces à partir de données issues de capture de mouvements. Nous cherchons une séquence d'activations de blendforces $\mathbf{u}_t$ permettant de suivre au mieux les trajectoires d'un ensemble de marqueurs faciaux représentés par leur vecteur coordonnées $\mathbf{d}_t$. Ce problème peut s'écrire comme

$$\arg\min_{\mathbf{u}_t} \|\mathbf{d}_t - \mathbf{S}\mathbf{x}_t(\mathbf{u}_t)\|^2 , \tag{4}$$

où $\mathbf{S}$ est une matrice sélectionnant les vertex du maillage facial correspondant aux marqueurs. Pour résoudre ce problème d'optimisation, nous remarquons qu'en linéarisant les forces internes s'appliquant au visage, il est possible d'exprimer $\mathbf{x}_t$ comme une fonction affine de la commande $\mathbf{u}_t$:

$$\mathbf{x}_t = \mathbf{\Phi}_t\mathbf{u}_t + \mathbf{q}_t, \tag{5}$$

où la matrice $\mathbf{\Phi}_t$ ne dépend que de $\mathbf{B}$ et des forces internes, et le vecteur $\mathbf{q}_t$ varie en fonction de la linéarisation des forces internes.

Nous pouvons donc formuler un algorithme de contrôle simple, consistant en l'aternance de phases de linéarisation des forces internes et calcul de la commande optimale $\hat{\mathbf{u}}_t$, obtenue via

$$\hat{\mathbf{u}}_t = \arg\min_{\mathbf{u}_t \geq 0} \|\mathbf{S}\mathbf{\Phi}_t\mathbf{u}_t - (\mathbf{d}_t - \mathbf{S}\mathbf{q}_t)\|^2 . \tag{6}$$

Nous ne permettons pas d'activations négatives des blendforces pour simuler le fait que les muscles faciaux ne peuvent que tirer sur la peau, et jamais la pousser.

0.2.3 Variété des expressions faciales

Nous avons vu dans la section précédente que le principal facteur différenciant le formalisme blendforces des blendshapes était la présence de forces internes. Dans cette section, nous analysons plus précisémment comment les forces internes affectent l'ensemble des configurations géométriques atteignables, et comment ces configurations se rapprochent des configurations faciales naturelles.

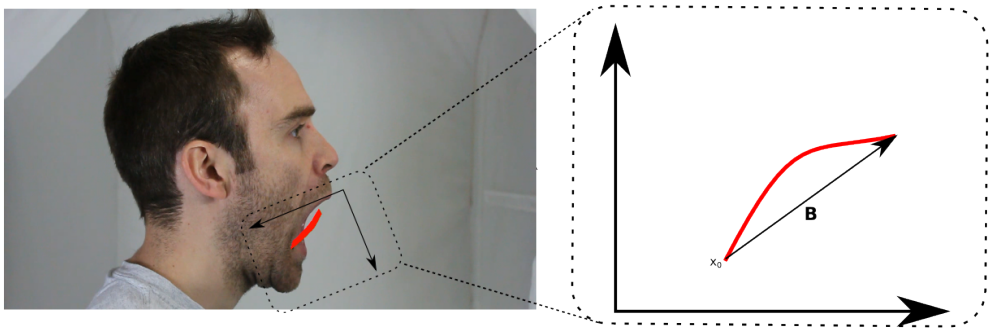


FIGURE 3 – Projection de la variété des expressions faciales $\mathcal{M}_{\text{true}}$ (en rouge) sur le plan caméra pour un point de la joue, limitée à l'expression d'ouverture de la machoire. L'approximation de cette variété par les blendshapes est représentée par la flèche noire.

L'ensemble des configurations faciales, formulé ici comme l'ensemble des coordonnées $\mathbf{x}$ d'un maillage de visage correspondant à des géométries du visages réalistes, forme une variété non linéaire

$$\mathcal{M}_{\text{true}} = \{\mathbf{x} \mid \mathbf{x} \text{ est un visage valide}\}. \tag{7}$$

Nous pouvons représenter une projection de cette variété en étudiant la projection caméra d'un seul point du visage au cours d'une expression donnée, comme représenté en figure 3. Les configurations atteignables par les blendshapes peuvent alors être vues comme une variété affine approximant la variété $\mathcal{M}_{\text{true}}$.

En raison de la présence de forces internes, le formalisme blendforces génère des configurations géométriques proches de la variété d'équilibre quasi statique $\mathcal{M}_{\text{eq}}$, qui regroupe les configurations pour lesquelles les forces internes et les blendforces sont en équilibre :

$$\mathcal{M}_{\text{eq}} = \{\mathbf{x} \mid \exists \mathbf{u} \text{ such that } \mathbf{B}\mathbf{u} + \mathbf{f}_{\text{int}}(\mathbf{x}) = 0\}. \tag{8}$$

Au cours d'une animation, les blendforces donnent une impulsion appartenant au sous-espace linéaire des blendshapes, tandis que les forces internes attirent en direction de la variété $\mathcal{M}_{\text{eq}}$. La résultante de ces forces permet d'obtenir une animation faciale présentant des configurations au voisinage de la variété $\mathcal{M}_{\text{eq}}$ (figure 4). Si la variété $\mathcal{M}_{\text{eq}}$ n'a *a priori* pas de raison d'être proche de la variété des expressions naturelles, nous remarquons qu'il est possible de paramétrer les forces internes pour permettre que les expressions encodées par les blendshapes appartiennent à $\mathcal{M}_{\text{eq}}$, rapprochant de ce fait les deux variétés.

0.3 Modélisation d'un système physique facial

Dans la section précédente nous avons présenté le formalisme *blendforces*, dans lequel nous produisons des animations faciales de synthèse au sein d'un système physique où les tissus faciaux sont mis en mouvements par l'action de muscles faciaux modélisés comme des forces vectorielles appartenant à l'image de la matrice de blendshapes $\mathbf{B}$, et où les

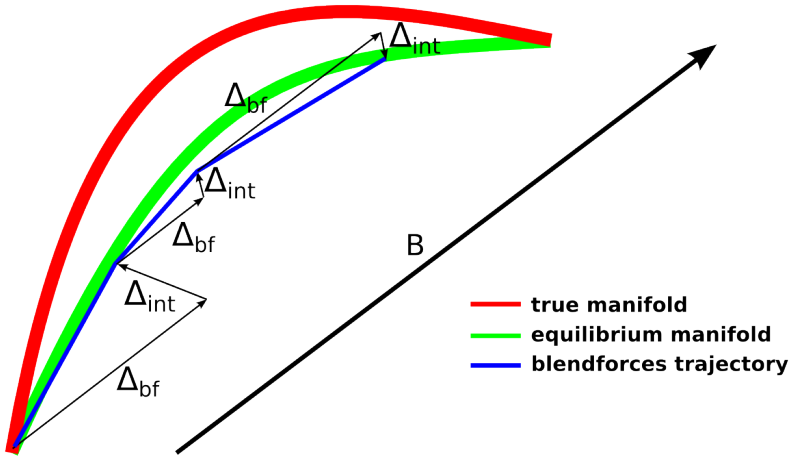


FIGURE 4 – Trajectoire des coordonnées du maillage facial pendant une animation blendforces. Sous l'influence combinée des blendforces et des forces internes, la trajectoire se situe au voisinage de la variété $\mathcal{M}_{eq}$ (en vert).

tissus faciaux résistent à ces forces musculaires sous la forme de forces internes. Nous avons vu que la définition de ces forces internes était cruciale pour permettere d'obtenir des animations réalistes. Dans cette section, nous nous intéressons à la définition de ces forces internes, et montrons comment nous pouvons créer des forces réalistes à partir des données surfaciques procurées par un jeu de blendshapes.

0.3.1 Influence du crâne

La peau est un tissu élastique ne présentant aucune rigidité. Sans la présence du crâne, les tissus faciaux ne prendraient pas la forme d'un visage. Afin de simuler la présence du crâne, nous utilisons l'information présente dans l'expression neutre. En effet, à l'expression neutre, la peau du visage adopte la forme du crâne aux points où elle le touche. Nous modélisons l'attachement de la peau à la surface du crâne avec des ressorts de longueur au repos nulle connectant chaque vertex à sa position sur l'expression neutre. Comme certains vertex ne sont pas connectés au crâne, nous laissons la possibilité de spécifier les régions du visage pour lesquelles ces ressorts ne doivent pas être créés.

0.3.2 Élasticité de la peau

0.3.2.1 Modélisation surfacique

La peau du visage est un tissu viscoélastique non homogène et non isotrope, avec une moindre élasticité le long des lignes de Borges [179] (voir la figure 5 pour les lignes de Borges). Nous pouvons modéliser ces comportements sur un maillage surfacique en plaçant des ressorts sur chaque arête du maillage. En faisant varier spatialement la raideur de ces ressorts, nous pouvons reproduire la non homogénéité de l'élasticité. Par ailleurs, les

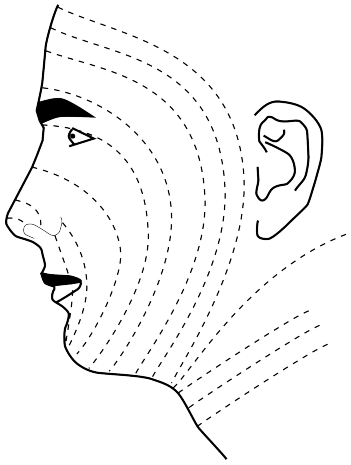


FIGURE 5 – Représentation schématique des lignes de Borges de relaxation de la tension dermique [22].

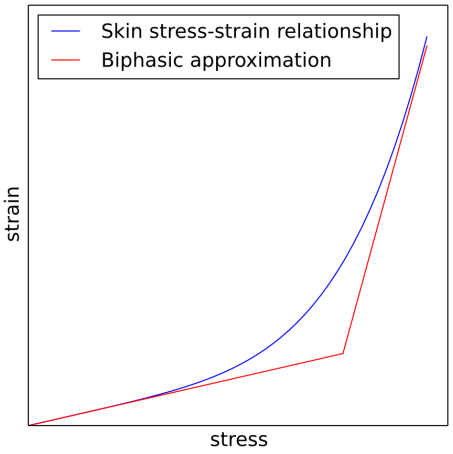


FIGURE 6 – Rapport entre la force élastique de la peau (stress) et l'élongation des ressorts utilisés (strain). Cette figure est purement illustrative, les courbes n'ont ici pas de valeurs exactes.

maillages de visages dédiés à l'animation ont généralement des arêtes orientées le long des lignes de Borges [137], ce qui permet également de modéliser les caractéristiques non isotropiques de l'élasticité de la peau.

Une autre caractéristique importante des tissus faciaux est le comportement non linéaire de l'élasticité [179]. Pour modéliser ce comportement, nous adaptons un modèle biphasique de raideur, où la raideur du ressort augmente à partir d'un palier d'élongation (figure 6).

L'utilisation des arêtes du maillage facial pour former un réseau de ressorts ne peux cependant servir pour simuler l'incompressibilité des tissus faciaux. Nous compensons ce problème en ajoutant des forces internes préservant la courbure du maillage facial telle qu'observée sur le visage neutre. Nous pourrions préserver cette courbure en ajoutant des ressorts reliant les vertex situés à une distance topologique de 2 sur le graphe des arêtes [119], mais nous préférons pénaliser explicitement les changements de courbure via une force préservant le Laplacien du graphe d'arêtes. La préservation de la courbure simule non seulement l'incompressibilité de la peau, mais simule également la présence du crâne en dessous de la peau, améliorant la technique présentée dans la section précédente.

0.3.2.2 Résultats

Nous comparons les animations produites dans le formalisme blendforces à des animations produites dans les mêmes conditions de capture de mouvement avec l'approche blendshapes. Les animations produites avec des blendshapes peuvent être obtenues par la résolution d'un problème aux moindres carrés avec paramètres non-négatifs, avec optionnelement une pénalisation en norme L1 du vecteur de poids de blendshapes $\mathbf{w}$. Dans la suite, nous

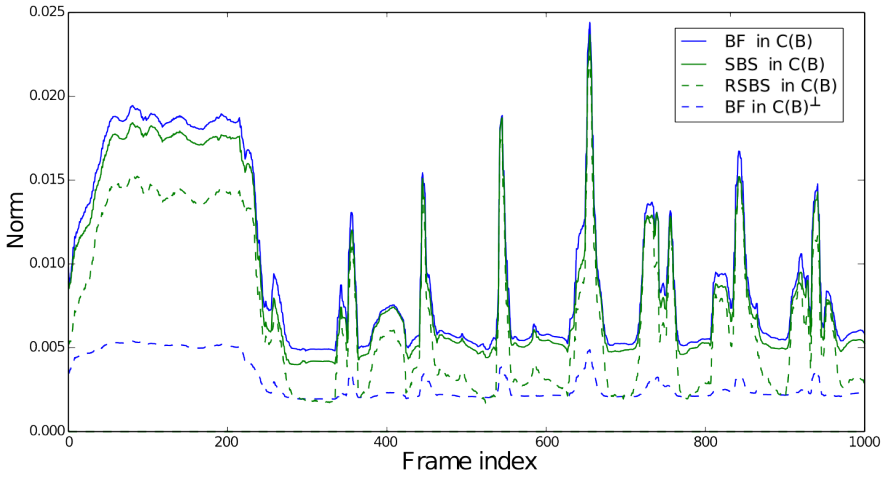


FIGURE 7 – Mesures des projections de $\mathbf{x}$ dans le sous espace $C(\mathbf{B})$ des blendshapes et dans son orthogonal $C(\mathbf{B})^{\perp}$, pour les méthodes BF, SBS et RSBS. Les résultats de blendforces ont une contribution plus importante dans $C(\mathbf{B})$, ainsi qu'une contribution significative dans $C(\mathbf{B})^{\perp}$. Les contributions dans $C(\mathbf{B})^{\perp}$ des méthodes SBS et RSBS sont trivialement identiquement nulles et ne sont donc pas affichées.

nommons SBS (Static Blendshapes Solve) les résultats obtenus par approche blendshapes sans régularisation, RSBS (Regularized SBS) les résultats blendshapes avec pénalisation L1, et BF les résultats obtenus avec le formalisme blendforces.

Nous observons tout d'abord que les animations produites avec les blendforces n'appartiennent pas au sous espace affine des blendshapes (figure 7) : elles présentent une composante non nulle dans le complémentaire $C(\mathbf{B})^{\perp}$. Nous observons que cette capacité de la méthode blendforces à sortir du sous-espace affine dans lequel évoluent les animations blendshapes permet de réduire l'erreur de suivi des marqueurs de capture de mouvement, en conservant une animation réaliste. Pour confirmer la précision apportée par la méthode blendforces, nous utilisons des données de capture de mouvement dense, et vérifions que non seulement les animations blendforces suivent mieux les mouvements des marqueurs, mais elles sont aussi plus précises pour prédire la position des points sur lesquels aucun marqueur n'est fourni, ce qui montre que les mouvements faciaux issus du formalisme blendforces sont en effet plus réalistes.

Nous étudions enfin la capacité du formalisme blendforces à introduire des dynamiques réalistes dans les mouvements de la peau. Comme montré en figure 8, les animations produites par les blendforces produisent des dynamiques de mouvement de peau plus proches de celles observables sur des visages naturels. En effet, les blendforces ne pouvant qu'agir dans une seule direction, le retour au neutre lors des relaxations des muscles est alors dicté uniquement par les caractéristiques des forces internes, là où les animations blendshapes peuvent produire n'importe quelle trajectoire temporelle.

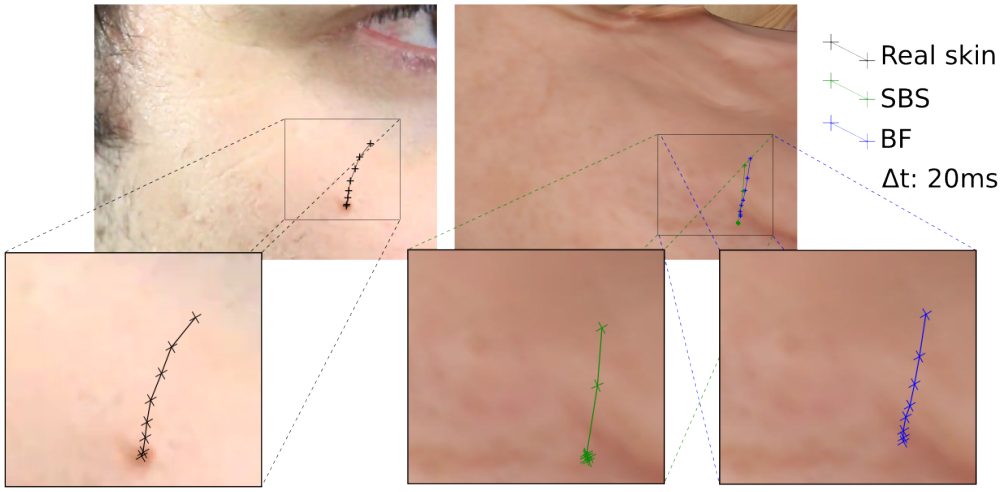


FIGURE 8 – Évolution temporelle d'un point sur la peau pendant la relaxation d'un muscle. Là où l'animation blendshapes (en vert) revient à la position neutre en 40ms, l'animation blendforces produit une trajectoire plus fluide, comparable aux trajectoires observables sur de vrais visages (image de gauche). Cet effet est mieux visualisé en vidéo.

0.3.3 Forces volumétriques

Bien que le modèle surfacique présenté dans la section précédente obtienne des résultats intéressants, il n'est pas satisfaisant de modéliser la peau comme une simple surface. Un problème de l'approche précédente est la nécessité d'utiliser une force de préservation de la courbure pour simuler le volume de la peau. Cependant, la présence d'une force préservant la courbure est gênante pour la simulation d'autres phénomènes physiques. En effet, la formation de rides nécessite une courbure de la peau, et les lèvres changent leur courbure lorsqu'elles sont pressées l'une contre l'autre. Il apparaît ainsi nécessaire de modéliser l'élasticité de la peau à l'aide de forces volumétriques. Cependant, nos données d'entrée ne sont que des maillages faciaux surfaciques, sans données de volume. Nous pourrions contourner ce problème en adaptant un modèle volumétrique de crâne et de tissus faciaux sur notre maillage [77], mais celà empêcherait notre méthode de fonctionner sur des maillages de visages non humains.

Nous prenons le parti de ne simuler que le volume de tissus faciaux situé entre la peau et le crâne. Pour créer ce volume, nous extrudons une surface sous la peau à partir du visage neutre en ajoutant un nouveau vertex le long de chaque vecteur normal, à une distance calculée comme un compromis entre une cible pré-définie et un terme visant à produire une surface régulière. Ainsi, chaque vertex de la surface *épidermique* d'origine se voit attribuer un vertex correspondant dans la nouvelle surface *hypodermique*. Nous construisons ensuite un maillage volumétrique entre ces deux surfaces en construisant trois tétrahèdres pour chaque triangle du maillage surfacique originel. Comme il y a plus d'une possibilité pour construire ces trois tétrahèdres, nous maintenons une structure cohérente en créant ces tétrahèdres lors d'un parcours du graphe d'adjacence des triangles.

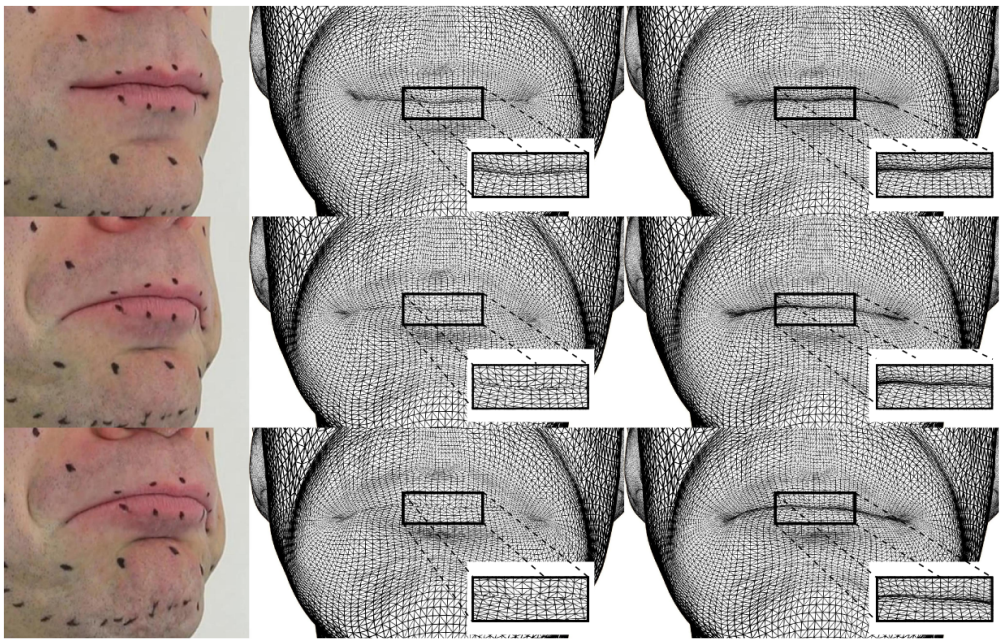


FIGURE 9 – Gestion des intersections de lèvres. La gestion des contacts est nécessaire pour produire la bonne forme de lèvres. L'animation blendshapes (au centre) produit des lèvres qui s'interpénetrent, et une forme de lèvres qui diffère de celle de l'acteur capturé. Au contraire, le formalisme blendforces dôté de forces internes gérant les collisions produit un maillage sans interpénétrations, avec la bonne forme de lèvres. Cet effet est mieux visualisé en vidéo.

Avec la donnée de ce maillage volumétrique, nous pouvons redéfinir les forces internes. Les forces d'attachement au crâne de la section 0.3.1 sont désormais utilisées pour les vertex de la surface hypodermique. Nous ajoutons une force d'élasticité volumétrique pénalisant les déformations de chaque tétrahèdre s'éloignant d'une transformation rigide, ainsi qu'une force de préservation du volume de la peau préservant le volume de chaque tétrahèdre.

0.3.4 Gestion des contacts des lèvres

0.3.4.1 Gestion des collisions

Les lèvres sont la partie la plus fexible d'un visage, et peuvent ainsi prendre une grande variété de formes. Les collisions entre les lèvres sont fréquentes, particulièrement pendant la parole. Il est cependant extrèmement difficile de capturer précisémment les mouvements de cette région. En effet, il est difficile de placer des marqueurs sur les lèvres, et les méthodes de capture sans marqueurs sont génées par les fréquentes occlusions. Les mouvements de lèvres capturés sont ainsi généralement peu précis, ce qui peut produire des intersections entre les surfaces des deux lèvres.

Nous détectons efficacement les collisions potentielles entre les lèvres avec du hashage

spatial [167], puis obtenons les collisions vertex-triangle et arête-arête en détectant la coplanarité des quatre points ainsi considérés [27]. Cette méthode permet de garantir que toutes les collisions sont détectées. Nous instantions alors une force interne pénalisant les collisions, proportionnelle à la profondeur de pénétration. Pour assurer une bonne séparation, nous maintenons une force non nulle tant que les éléments en collision ne sont pas séparés par une distance minimale (0.01mm dans nos expériences).

0.3.4.2 Lèvres collantes

Nous profitons de la détection des collisions entre les lèvres pour simuler le phénomène des lèvres collantes. Dans ce phénomène, l'humidité des lèvres les pousse à rester collées après contact, et ce collage reste actif jusqu'à ce que la force exercée soit suffisante pour le rompre, ou que l'assèchement des lèvres ne mette fin à l'effet.

Pour chaque collision entre les lèvres, nous rajoutons une nouvelle force interne sous la forme de ressorts de longueur au repos nulle connectant les points rentrés en contact. Ces ressorts sont dôtés d'une probabilité de se casser à chaque fin de pas de simulation physique. La probabilité de se casser augmente avec la longueur du ressort.

0.3.4.3 Résultats

Nous vérifions à présent la capacité de ces nouvelles forces à produire des effets réalistes. La figure 9 montre que l'addition de forces gérant les contacts permet effectivement d'empêcher les interpénétrations, et d'ainsi produire des formes de lèvres plus réalistes. Par ailleurs, la figure 10 montre que les lèvres collantes simulées par notre système sont proches de celles observées sur des visages réels. De plus, nous vérifions également que la dynamique de formation et de rupture des lèvres collante correspond également à celle observée sur des visages réels.

0.4 Simulation physique d'animation faciale temps-réel

Les sections précédentes ont présenté le formalisme blendforces, avec lequel nous produisons des animations faciales de synthèse réalistes via une simulation physique basée sur un jeu de blendshapes. Ce formalisme était décrit dans le contexte d'une animation basée sur de la capture de performance à l'aide de marqueurs. Cependant, nécessiter des marqueurs ne permet pas d'utiliser facilement notre systèmes dans de nombreux contextes. Dans cette section, nous montrons que notre méthode peut-être adaptée pour fonctionner à partir de capture de performance effectuée depuis une simple caméra RGB, sans nécessiter de marqueurs. De plus, nous montrons qu'il est possible d'effectuer la simulation physique en temps-réel, augmentant ainsi la gamme des situations dans lesquelles notre méthode peut être utilisée.

FIGURE 10 – Comparaison des formes des lèvres collantes naturelles avec celles simulées par notre méthode (images de droite). Nous observons des formes similaires, montrant le réalisme de notre méthode.

0.4.1 Contrôle temps-réel

Nous proposons un système pour produire en temps-réel des animations faciales via le formalisme blendforces à partir d'images provenant d'une caméra RGB (figure 11). Nous commençons par obtenir la position des contours caractéristiques du visage (yeux, sourcils, lèvres, ...) en utilisant une méthode d'apprentissage supervisé basée sur une cascade d'arbres de régression [82]. Ensuite, nous utilisons un modèle paramétrique tridimensionnel de déformations du visage en fonction de l'identité et de l'expression formé avec la base de données FaceWarehouse [34]. En minimisant l'erreur de reprojection de ce modèle dans l'image, nous pouvons obtenir les paramètres d'identité et d'expression correspondant à l'image capturée par la caméra RGB. En ré-appliquant les paramètres d'expression sur l'identité moyenne, nous obtenons alors des marqueurs virtuels que nous pouvons utiliser pour contrôler une animation blendforces comme en section 0.2.2. Nous utilisons des fonctions à base radiale pour apprendre une fonction compensant les différences de morphologie entre l'identité moyenne et le visage cible [153].

En retrouvant les positions tridimensionnelles des points du visage, nous obtenons également la pose du visage au cours du temps. Nous utilisons cette information pour prendre en compte dans notre simulation les forces inertielles, ainsi que la gravité dont nous pouvons ainsi connaître l'orientation relativement au visage, en supposant que la caméra soit posée dans une orientation connue. Ces forces produisent généralement des effets négligeables sur les visages, mais sont intéressantes pour simuler des personnages non humains, par exemple un personnage possédant des tentacules sur son visage.

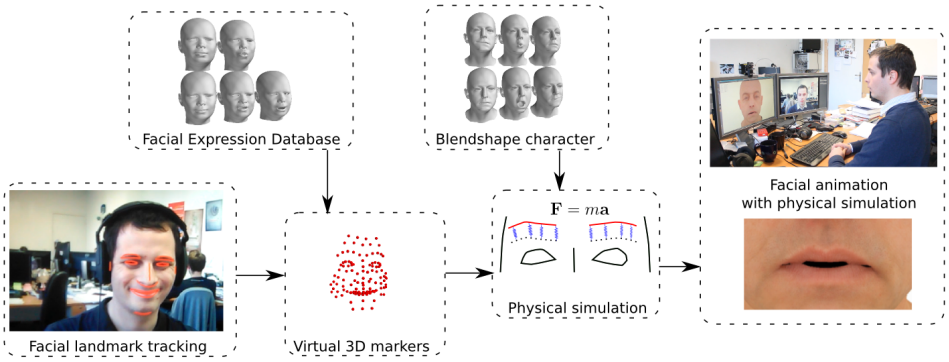


FIGURE 11 – Procédé pour produire des animations faciales avec simulation physique en temps-réel depuis une caméra RGB.

0.4.2 Solveur physique temps-réel

Pour la simulation physique, nous utilisons le formalisme Projective Dynamics [24]. Dans ce formalisme, la simulation physique est réalisée par la succession d'étapes de projection permettant de calculer une approximation linéaire des forces internes, et une étape de résolution globale des contraintes dans laquelle la position du système physique est raffinée. Cette résolution globale fait intervenir la résolution d'un système linéaire avec une matrice constante si la configuration des forces internes ne change pas. Sous cette condition, avec une factorisation de Cholesky pré-calculée, la simulation physique peut-être effectuée en temps réel. Cependant, avec la présence de forces internes pour simuler les collisions et les lèvres collantes, cette matrice n'est plus constante, et il n'est alors plus possible d'utiliser la factorisation de Cholesky, trop coûteuse si réalisée à chaque pas de temps. Récemment, Wang [174] a proposé de remplacer l'utilisation de Cholesky par une itération de Jacobi, ce qui permet de ne pas nécessiter de pré-factorisation. Cependant, nous avons observé que, pour des forces volumétriques, les itérations de Jacobi ne convergeaient pas. De plus, Wang requiert un GPU afin d'effectuer un grand nombre d'itérations, mais nous souhaitons que notre méthode fonctionne sur CPU.

Liu et al. [112] ont proposé une méthode pour accélérer la convergence de Projective Dynamics, en intercalant la méthode L-BFGS pour apprendre de meilleures actualisations du système physique. Leur méthode requiert cependant la pré-factorisation de Cholesky de la matrice de résolution globale. Nous proposons d'étendre leur méthode pour y incorporer une méthode itérative de résolution de système linéaire. Nous choisissons d'incorporer la méthode itérative Gauss-Seidel. En effet, cette méthode est garantie de converger sur des matrices symmétriques définies positives, une caractéristique des matrices de résolution globale dans Projective Dynamics.

Nous incorporons Gauss-Seidel dans l'approche L-BFGS de Liu et al. en remplaçant les résolutions par Cholesky pré-factorisées par un nombre variable d'itérations de Gauss-Seidel. Partant de 2 itérations de Gauss-Seidel, nous doublons ce nombre à chaque fois

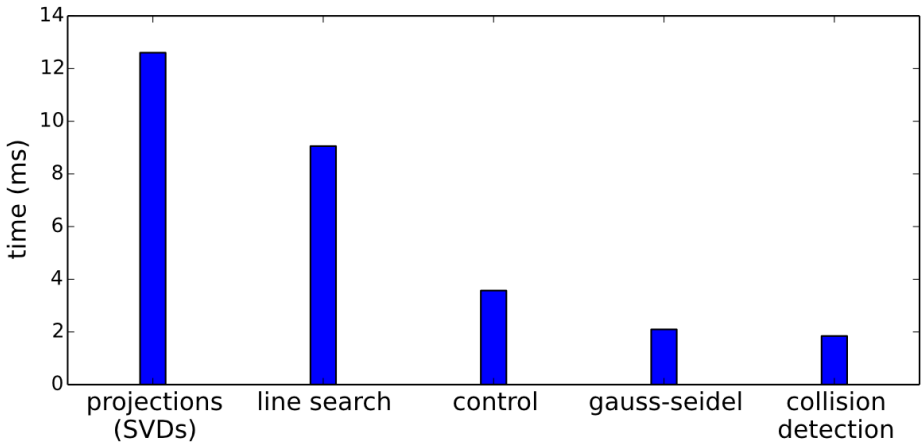


FIGURE 12 – Répartition des temps de calcul sur un pas de temps impliquant 50 contraintes de collision et 110 contraintes de lèvres collantes.

que la direction calculée par cette méthode est considérée comme ne produisant pas une réduction d'erreur suffisante. Ainsi, notre méthode ne nécessite qu'un minimum de calcul pour atteindre un niveau de précision donné.

0.4.3 Approximation rapide de la SVD

La simulation de forces volumétriques dans le formalisme Projective Dynamics requiert le calcul de nombreuses décompositions en valeurs singulières (SVD). Pour chaque tétrahèdre, la SVD $\mathbf{F} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ de son gradient de déformation $\mathbf{F}$ doit-être calculée. Cette décomposition est une décomposition itérative, dont le calcul est coûteux. Même avec une simulation physique effectuée sur un GPU, où les calculs massivement parallèles sont favorisés, le calcul d'une SVD par tétrahèdre est tellement coûteux qu'il domine le temps de calcul total [174].

Nous proposons de tirer parti du côté itératif de la simulation physique pour réduire drastiquement le nombre de SVD à calculer. En effet, entre chaque itération de Projective Dynamics, les coordonnées du système physique n'évoluent que peu. Nous pouvons donc approximer les SVD à calculer en utilisant les Jacobiens des précédentes SVD calculées. Cependant, un calcul exhaustif des Jacobiens de SVD [138] est encore plus coûteux qu'une décomposition. Nous contournons ce problème en remarquant que, dans le cas d'un faible mouvement, nous pouvons nous ramener au cas particulier de la SVD d'une matrice diagonale. Dans ce cas particulier, les Jacobiens de SVD peuvent être calculés avec une solution de forme fermée, et sont de plus des matrices très creuses. Nous remarquons de plus qu'en calculant les Jacobiens des matrices $\mathbf{U}$ et $\mathbf{V}$ en termes d'angles d'Euler, ces matrices sont encore plus creuses, ne nécessitant plus que le calcul de six scalaires. Ainsi, le calcul des Jacobiens d'une SVD ne prend que 9.13ns, contrairement au calcul exhaustif qui prendrait 456ns.

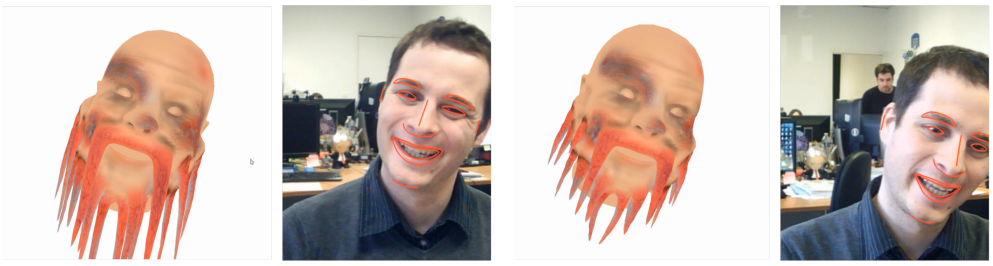


FIGURE 13 – Action de la gravité sur un personnage non humain. Les tentacules du personnages s'inclinent et s'allongent selon la gravité (à gauche). En comparaison, une animation blendshapes (à droite) donne l'impression d'un personnage fait de plastique. Cet effet est mieux visualisé en vidéo.

Avec la donnée des Jacobiens de SVD, nous pouvons de plus utiliser les normes matricielles pour déterminer en avance si la nouvelle SVD sera significativement différente de la précédente. Nous fixons ainsi deux paliers, le premier déterminant s'il suffit de réutiliser la SVD précédente, et le second déterminant s'il vaut mieux utiliser une approximation de Taylor basée sur les Jacobiens de SVD ou recalculer la SVD. Ainsi, nous pouvons éviter de calculer la grande majorité des SVD, avec en moyenne 44% des SVD ne nécessitant aucun recalcul, et 40% nécessitant une approximation de Taylor. Notre méthode permet d'éviter toute dérive, puisque toute déformation significative entraine un calcul exact de la SVD et des Jacobiens associés.

0.4.4 Résultats

Nous testons la performance de notre système sur un maillage de visage composé de 14000 vertex et de 45000 tétrahèdres. Sur un CPU Intel Xeon E5-1650 dôté de 6 coeurs cadencés à 3.2 GHz, la simulation d'un pas de temps sans collisions ou contraintes de lèvres collantes prend 28ms, tandis qu'un pas de temps avec ces contraintes prend jusqu'à 40ms. Ainsi, notre système peut soutenir les 25 images par seconde de la caméra RGB et être utilisé en temps réel. Nous montrons en figure 12 la répartition des différents coûts dans le calcul d'un pas de temps. Nos principaux coûts proviennent de la résolution globale de Projective Dynamics et du calcul des SVD. Nous remarquons que pour les calculs de SVD exacts, nous utilisons l'implémentation de la bibliothèque C++ Eigen, qui n'est pas la plus performante. Il est probable que passer à une méthode plus optimisée [124] permettrait d'atteindre de meilleurs temps de calcul.

Qualitativement, notre système capture correctement l'expression de l'utilisateur en temps réel, et produit une animation réaliste augmentée d'effets physiques impossibles à obtenir sans simulation physique. Ainsi, les lèvres collantes de la figure 10 présentée dans la section précédente ont été simulées en temps réel avec le système présenté dans cette section. Pour une meilleure appréciation des résultats de notre système, il est recommandé de les regarder en vidéo.

Avec la prise en compte de la pose du visage dans notre simulation, nous obtenons

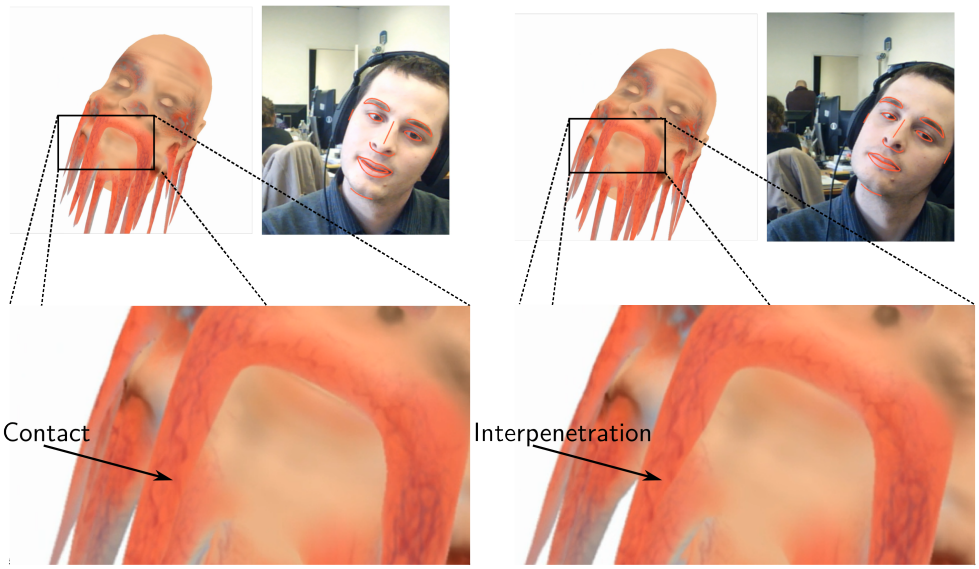


FIGURE 14 – Notre système peut être utilisé pour empêcher les collisions entre les tentacules d'un visage non humain. À gauche : sans gestion des collisions. À droite : avec gestion des collisions. Sans la gestion des collisions, les tentacules pénètreraient le visage.

des effets visuels intéressants sur les personnages non-humains. Ainsi, en figure 13, nous montrons que la prise en compte de la gravité permet de rendre réaliste l'animation d'un visage avec des tentacules. En effet, sans cette force, les tentacules restent figées, donnant l'impression d'animer une marionnette en plastique plutôt qu'un personnage de chair. Nous pouvons également utiliser nos forces internes dédiées aux contacts entre les lèvres pour gérer les collisions des tentacules (figure 14), et même faire coller les tentacules grâce aux forces de lèvres collantes.

0.5 Conclusion

Nous passons à présent en revue les contributions et résultats développés au cours de cette thèse. Dans la section 0.2 nous avons introduit le concept des blendforces, qui nous permet de créer un système de simulation physique à partir d'un jeu de blendshapes faciaux. Nous avons doté ce formalisme d'un algorithme de contrôle de ces blendforces, permettant de créer des animations de visage à partir de données de capture de mouvements.

Nous avons continué notre exploration des blendforces dans la section 0.3, où nous avons montré que des forces internes réalistes pouvaient être déduites de la donnée d'un jeu de blendshapes constitué de données uniquement surfaciques. Ainsi, nous avons proposé une technique pour simuler des forces volumétriques à partir des maillages surfaciques initiaux, et avons également proposé des forces internes déstinées à gérer les contacts et collages des lèvres. Nous avons montré que l'utilisation de ces forces internes dans le

formalisme blendforces permettait de produire des animations faciales réalistes, fidèles aux mouvements capturés, et améliorées par la présence d'effets physiques difficiles à capturer.

Dans la section 0.4 nous avons repoussé les limites de notre système pour produire nos animations faciales avec simulation physique en temps réel. Nous avons ainsi proposé un système pratique, reposant sur la capture de mouvements faciaux sans marqueurs en temps réel à l'aide d'une simple caméra RGB. Pour parvenir à nos fins, nous avons proposé des améliorations significatives au formalisme de simulation physique Projective Dynamics, avec une méthode de résolution globale adaptée à la simulation de systèmes physiques présentant des changements dans le nombre et la nature des forces internes. Nous avons également proposé une méthode efficace pour approximer la SVD, réduisant grandement le coût de calcul des forces volumétriques. Grâce à ces contributions, notre système est, à notre connaissance, le premier exemple d'un système de simulation physique temps-réel appliqué à l'animation faciale.

Avec ces contributions, nous avons pleinement atteints les objectifs fixés en section 0.1. Nous avons créé un système de simulation physique mettant à profit la popularité des blendshapes. Ce système est aisé à mettre en oeuvre puisqu'il ne requiert qu'un jeu de blendshapes et une caméra RGB pour produire des animations basées sur la capture de mouvements. Enfin, ces animations peuvent être produites en temps réel, donnant ainsi un retour immédiat à l'utilisateur.

Ce chapitre constitue uniquement un résumé du contenu en anglais présenté dans les chapitres suivants. Pour plus de détails, prière de se référer à la suite de cette thèse.

Chapter 1

Introduction

Contents

1.1 Motivation and Problem Statement

The last decade has seen a constant rise of the production of synthetic media, from movie special effects to video games, and recently virtual reality. These media require the production of several hours of synthetic facial animation, whose quality may be essential in the acceptance of the produced media. However, producing convincing facial animation is a difficult task, which despite years of research effort from the graphics community, still cannot produce results that would fool an observer. Human beings are so used to interpreting subtle variations of facial expressions that any missing detail can drastically reduce the perceived quality of the animation.

The now widespread usage of ever more advanced performance capture systems makes it possible to capture every subtle detail of an actor's performance, but transferring that performance to a virtual character's animation can lead to losing the subtlety of the movements. Indeed, the industry standard for facial rigs, the *blendshapes*, cannot represent the full complexity of facial expressions [98]. As an affine model, blendshapes are ill-suited for representing the inherently non-linear deformation of the skin.

Simulating the true physical behavior of facial elements does lead to high quality facial animations [158], but building an adequate physical model is a daunting task that requires costly data acquisition, as well as tedious manual work.

With a growing demand for high-quality synthetic facial animation, expensive physical simulation methods that could be justified for movies can not be adapted to a wider audience.

Conversely, the creation of blendshapes rigs has recently been made extremely practical, with simple methods requiring only a smartphone to work [75, 35]. It has therefore never been simpler to produce blendshapes-based synthetic facial animations, but the quality of these animations suffers from the limitations inherent to the blendshapes framework.

We propose to leverage the wide availability of blendshapes sets to build face physical systems that can be animated with practical performance capture. Building upon already existing blendshapes sets will allow our system to be widely deployed, while physical simulation will improve the realism of the resulting synthetic facial animation by allowing to escape the affine space blendshapes-based animations are confined into. Among the enhancements that physical simulation can bring, we are particularly interested in getting lifelike skin dynamics, handling lips contacts, and simulating the sticky lips phenomenon.

The recent development of several blendshapes-based realtime performance capture techniques [26, 105, 32] has extended the application area of synthetic facial animation to new domains such as telepresence and avatar-based video conferencing, live theater, and interactive video games. We therefore propose to design our facial animation system with the goal of realtime performance for the physical simulation, and require the ability to work with perfomance captured by widespread devices such as RGB cameras.

The objective of this thesis is thus to design a practical realtime facial animation system, driven by performance capture, that presents the advantages of both the blendshapes framework and physical simulation. More precisely, our objectives are as follows:

- **Blendshapes-based.** We wish to build a face physical system by leveraging existing blendshapes characters, taking advantage of their wide availability and their ease of creation.
- **Physically-based.** We want to produce facial animation presenting characteristics that can only be achieved with physical simulation.
- **Practical.** The resulting system must be easy to deploy, requiring a minimal setup.
- **Realtime.** The system must be performant enough for realtime applications, namely the creation of live facial animation from realtime performance capture.

1.2 Organization of this Thesis

We organized this thesis as a progressive construction of our target system. It is intended to be read linearly.

In chapter 2 we start by exploring the problem domain and analyzing the relevant related works. We organized our analysis of the state of the art by segmenting prior art with respect to their degree of proximity to the actual physical process underlying the formation of facial expressions. We identify connections between some of these domains that plead for a combination of blendshapes-based approaches and physical simulation.

In chapter 3 we then further explore the relevant background notions upon which this work is built. We thus analyze the formalism behind the blendshapes approach, exploring its

advantages and shortcomings. We subsequently present the physical simulation framework used in this work, as well as the principles behind the simulation of elastic materials.

Our contributions are exposed in chapters 4, 5, and 6. We begin by introducing the *blendforces* framework (chapter 4), wherein blendshapes are interpreted as a basis of forces that can be interpreted as an approximation of muscular forces. We show how, used in a dynamic physical simulation framework, these forces can be understood as a generalization of the blendshapes framework. In this framework, the adjunction of internal forces keeping the face physical system together enable the face mesh to reach configurations outside the affine subspace of blendshapes while maintaining physical plausibility. We also describe how the blendforces framework can be used to produce animations from performance capture data.

In chapter 5 we describe the construction of a face physical system from a surfacic face mesh. We show how using surfacic internal forces to simulate elasticity and preserve the curvature of the face, in conjunction with blendforces, can be used to accurately fit motion capture data, escaping the affine subspace of blendshapes, and generates lifelike skin dynamics. We subsequently propose a technique to build a volumetric face mesh from a surfacic face mesh, enabling the simulation of more anatomically accurate volumetric internal forces. We further enhance the realism of our system by adding internal forces suited to simulate lips contacts and sticky lips.

Our last contribution, exposed in chapter 6, is the realization of a realtime performance-driven facial animation system based on the physical system presented in the previous chapters. We describe how we leverage a simple RBG camera to drive our physical system, and detail improvements to a state-of-the-art physical simulation framework that enable realtime physical simulation of facial animation.

Chapter 7 concludes this thesis. We show how the techniques we proposed constitute an improvement of the state of the art, and reflect upon future improvements that could further enhance our system.

1.3 Mathematical Notation

We now describe the conventions used throughout this thesis for mathematical notations. We denote scalars using lowercase letters such as w. Vectors are represented as bold lowercase letters, e.g. $\mathbf{x}$. Matrices are represented using bold uppercase letters, for instance $\mathbf{A}$. With the notable exception of the self-contained section 3.3, we keep the same variables for the same concepts during this thesis. For instance, a recurring variable is $\mathbf{x}$, which represents the coordinate vector of a face mesh. Another recurring variable is $\mathbf{B}$, the matrix containing delta-blendshapes coordinate vectors as its columns.

Chapter 2

Related Work

Contents

Facial animation is the process of generating plausible facial expressions for digital characters. Since humans are used to interpret emotions through facial expressions, any person will be able to detect the smallest issues in computer generated facial animation. Creating realistic facial animation is thus a challenging task, which has been subject to a huge amount of research since the pioneering work of Parke [139]. In this chapter, we investigate the various approaches dedicated to the creation of realistic facial *animations*. We are however not interested in this work in describing the techniques dedicated to the *synthesis*

of realistic still facial imagery. Instead, we focus on methods able to produce animation on a pre-existing digital character, and focus particularly on methods relying on *performance capture* [185]. Indeed, producing high quality facial animation by hand is certainly an efficient method, as has been demonstrated by many movie studios, but it is more of an artistic process. Even though we are interested in motion capture, we do not investigate methods only targeted at the high-quality acquisition of facial geometry. These methods are often important in facial animation pipelines, but are more often concerned with computer vision aspects, and do not attempt to recover a semantic parameterization for the underlying facial animation. As we'll see throughout our state of the art investigation, several methods presuppose the existence of high quality facial geometry acquisition techniques, and describes methods to transfer the captured movements to digital characters.

We classify facial animation techniques with respect to their degree of proximity to the actual physical process underlying the formation of facial expressions. Machine learning methods (section 2.1) establish statistical *expression spaces* for the performance capture data and the target digital character, and learn mappings between these spaces to transfer the captured animation. Deformation techniques (section 2.2) attempt to parameterize the possible deformations of the face and establish techniques to transfer captured deformations to digital characters. Blendshapes methods (section 2.3) parameterize the facial deformation as a linear combination of basic shapes, each shape corresponding to the individual activation of a facial muscle. Physical simulation methods (section 2.4) aim at building a face physical model able to simulate the deformation of facial tissues caused by the activation of facial muscles.

This classification is naturally not absolute, and we'll see methods bringing together ideas from two or more domains. This thesis bridges between multiple domains: our work establishes links between blendshape methods and physical simulation methods, and combines the ease of modelling of the former with the accuracy of the latter.

2.1 Data-Driven Animation Transfer

Motion capture setups typically rely on capturing an actor's facial expressions using computer vision techniques such as tracking face contours or markers. However, these data do not map directly to an animation, and need to be *retargeted* to the specific controls of the target digital character.

2.1.1 Expression Mapping

Tracking parameters are often analyzed using *Principal Component Analysis* (PCA) [79]. This reduces the dimensionality of the input data, which makes the problem of mapping to a target space more tractable. The captured parameters can then be transferred using linear regression as showed by Chai et al. [39], Kuratate and colleagues [91], Hunty and collaborators [74], or Weise et al. [182]. Other regression methods have been used: Deng and coworkers [51] and Pyun et al. [144] used radial basis functions while Buck and colleagues [28] leveraged Delaunay triangulation. Akhter and colleagues [4] explored the

correlations between spatial components and time in facial animation, and devised a bilinear representation suited at filling gaps in animation data. Costigan et al. [49] compared the retargeting abilities of radial basis function networks versus artificical neural networks, and found that the limited number of retargeting examples made neural networks impractical.

Learning an accurate mapping between the space of performance captured points and rig parameters of the target digital characters requires the manual specification of lots of correspondences and is thus often a bottleneck in facial animation pipelines. To alleviate this issue, Bouaziz and Pauly [25] proposed to recast this problem as a semi-supervised learning problem, where non-corresponding examples in both the source space and the target space could be used to improve the quality of the mapping.

The source performance capture data can also be decomposed into keyshapes, which can form a *dictionary* onto which the data can be projected. An artist can then sculpt corresponding shapes, which can then be blended according to the projection weights, as illustrated by Chuang and Bregler [45, 46]. In speech generation, the phonemes can be grouped as visual keyshapes named the *visemes*. Cao and colleagues [36] identified patterns in phonemes sequences to retrieve the appropriate sequence of visemes to generate speech animation in realtime from audio data. Li and collaborators [107] performed image-based expression retargeting by matching optical flow components against pre-recorded keyshapes and blending the closest images.

Principal Component Analysis can also be used to model the expression space of a digital character, which enables intuitive facial posing interfaces. Kshirsagar and Magnenat-Thalmann generated speech animation by optimizing trajectories in a PCA space of facial expressions. Deng and Ma. [52], Li and Deng [108], Lau and colleagues [95], and Tena et al. [164] segmented the face in local regions, and performed PCA independently on each of these regions. Ma et al. [120] performed further analysis on these parameters to retrieve and transfer an animator's animation style. By representing their controllers in a PCA space, Rhee and colleagues [146] are able to transfer FACS-like measurements from video tracking to an animated mesh. Meyer and Anderson [125] designed a facial posing system where a set of intuitive control points was derived from a rotated PCA basis computed from a training set of facial expressions, allowing local and flexible animation controls. Feng et al. [62] leverage the same control points but perform the shape deformation using kernel Canonical Correlation Analysis, allowing for a better learning of the non-linear components of the deformation. Rhee and colleagues [146] identified an algebraic group structure in both the expression spaces of tracked facial landmarks and animation controls of a target mesh, and computed a morphism to transfer animation between these expression spaces.

2.1.2 Invariance to Identity Variations

Intuitively, the variation of facial geometry can be explained by differences in morphology or in expression, and these variations should be independent. This has led Vlasic et al. [171] to model facial geometry as a bilinear model, with separate modes for the variations of morphology and expression. The resulting model can then be used to transfer expressions between two individuals' videos, or photographs, as shown by Macedo and collabora-

tors [121]. Yang et al. [187] used different PCA models accounting for the variations due to identity for a set of expressions to be able to automatically transfer expressions of a subjet to a different photograph of the same subject. Wang and Ahuja [175] and Abboud and Davoine [1] performed a bilinear parameterization of face images, allowing for the synthesis of new facial imagery for unseen identities. Chang and Ezzat [41] relied on image-based multilinear decomposition to perform speech transfer. Huang and De La Torre [72] further extended image-based bilinear decomposition with bilinear kernel rank regression, a decomposition that is capable of capturing non linear patterns while enforcing separate modes for morphology and expression. While statistical models of face identity can be generated from anthropometric data, as did DeCarlo and colleagues [50], it is more accurate to build a statistical model from high-quality scan data, as illustrated by Blanz and Vetter [21]. Blanz and colleagues later showed [20] that building a standardized expression space on top of a statistical identity space could be used to transfer expression between images. Wang et al. [177] studied the influence of identity on the style of facial expressions using local linear embeddings.

Using PCA to represent the space of possible facial animations is widespread, but is not entirely satisfying. Stoiber and colleagues [162, 161] noticed that facial expressions tend to appear only in a manifold inside the PCA space of expressions, and used this insight to design intuitive controls for facial animation. Lewis et al. [97] studied the properties of the gaussian probabilty distribution implied by PCA decomposition, and concluded that these properties were not accurate when trying to model the space of facial identity variations.

2.2 Geometric Deformers

Facial geometry is usually represented as a polygonal mesh. To be able to deform the mesh to produce facial expressions, the space of possible deformations has to be reduced to only account for plausible skin movements. This can be done using the widespread linear blend skinning technique (section 2.2.1), or by transferring captured deformations to a new mesh, either high scale deformations (section 2.2.2) or details of the skin deformation (section 2.2.3).

2.2.1 Linear Blend Skinning

A simple yet versatile method for parameterizing the deformations of a facial mesh is Linear Blend Skinning (LBS) [78]. In this technique, each vertex's position is obtained by transforming the rest position using a transformation obtained by blending between the transformations of a predetermined set of control points: the *bones*. The simplicity and efficiency of the LBS technique has made it the most popular rigging technique for video games engines [53]. Rigging a digital character with LBS often leads to rigs with an intuitive set of controllers suitable for hand animation of the facial performance, however this formalism enables configurations outside the possible configurations of a human face. A widespread technique is thus to use motion capture to obtain the movements of points on the skin, and to leverage the simplicity of the LBS formulation to solve for the bones

transformation that lead to the desired transformation [60]. Von der Pahlen et al. [172] transformed a very detailed performance capture into bone transformations suitable to animate a digital double in realtime with the captured performance by computing the bones transformations that matched the deformations of the captured data. Dutreve et al. [54] developed a performance capture method where motion capture markers where transferred to bone positions using radial basis function network interpolation. Orvalho and colleagues [136] leveraged Thin Plate Spline interpolation to transfer rig elements, and applied their method to the transfer of linear blend skinning rigs. Lewis et al. [100] introduced Pose Space Deformation (PSD) to enhance deformers such as linear blend skinning with additional deformations able to compensate for the artifacts.

Even though LBS is an extremely popular scheme, alternative deformation-based rigging techniques have emerged. Kalra et al. [81] used Rational Free Form Deformations to model the action of muscles on the face, using a per-vertex weighting system to take into account the variation of mass and elasticity throughout the face. Lavagetto and Pockaj [96] deformed the face mesh by displacing control points and using radial basis function networks (RBFN) to propagate the deformation. Similarly, Noh et al. [133] controlled the deformations of the facial mesh by attributing a local influence areas to their control points and using RBFNs to generate smooth deformations.

2.2.2 Transferring Deformations

A common requirement in facial animation pipelines is to be able to transfer an existing facial performance to a new digital character. This can be used to produce raw animation data, to produce reference data onto which rig parameters such as bone positions are solved, or to transfer a rig altogether.

Noh and Neumann [134] introduced expression transfer, a technique that enabled tranferring an existing facial animation to a new character by computing a dense surface correspondence between the meshes using radial basis function network interpolation and transfering the deformation vectors from the source character to the target while taking local changes of scale and rotation into account. Na and Jung [130] transfer deformation using radial basis function networks, but only do so for a coarse mesh, and transfer details by transferring offsets to the coarse mesh as motion vectors. Li et al. [102] argued for the use of geodesic distances instead of euclidean distance when using radial basis function networks to transfer deformation, since the euclidean distance does not enable to distingusih between the upper and lower lips. Sumner and Popovìc [163] introduced *deformation transfer*, a technique where the local motion of triangles can be transferred to another mesh given an appropriate dense surface correspondence. This technique has since then seen wide use in most facial animation pipelines. Saito [149] added virtual triangles between the eyelids and the lips to improve the ability of deformation transfer to respect eye and lip contacts. Xu et al. [186] enhanced deformation transfer by specifying feature lines around lips and eyelids and adding energy terms to guide deformation transfer towards better respect of the deformations of these lines. Their technique also enabled manual editing of the transfer results, with temporal propagation.

Asthana and colleagues [6] used a 2D deformation transfer to build a tracking model.

Saragih et al. [151] performed image-based animation of an avatar from a single image by transferring the same known deformations to the avatar image and to the user's image and learning a mapping between them. Chang and Jenkins [40] proposed to edit facial expressions by sketching the desired result and deforming the facial mesh to match the sketch. Zhang et al. [192] generated new facial expressions from example images by deforming the examples to match the location of user-defined control points.

2.2.3 Fine Scale Deformations

While high-fidelity motion capture techniques record skin movements with a resolution capable to describe skin pores, these methods are expensive and require users to sit inside complex scanners such as the Light Stage [5]. These methods thus prevent the simultaneous capture of facial and body performance, making model-based or marker-based motion capture the only options for most practionners. However these methods cannot capture the movements with sufficient precision to encompass the formation of details such as the formation of wrinkles. It is therefore useful to transfer existing fine scale deformations on animations feature only a medium level of details.

In Zhang et al. [191], a cascade of PCA models is used to generate fine-scale details. Bickel and colleagues [17, 19] developed a face capture setup able to record geometric deformations of the face precise enough to include wrinkles, and designed a method to deform a target mesh to produce wrinkles similar to those captured based on its coarse configuration. Ma and coworkers [118] transfer high resolution wrinkles obtained with a Light Stage [5] using polynomial displacement maps. Bermano et al. [15] retrieved matching low resolution performances in a database of high quality facial deformations and transferred the high frequency components to improve the quality of animations obtained from sparse signals such as marker motion capture or model-based motion capture. Li and colleagues [106] built user-specific wrinkle models using and RGBD sensor and high quality examples of wrinkles, and used said model to produce realtime facial animation from RGBD input with accurate wrinkle capture. Cao and collaborators [31] learnt a mapping between image shading and facial wrinkles to augment a realtime facial tracker with high frequency details learnt from a database of high quality facial deformations. Kim et al. [85] designed a system where wrinkles could be added to a facial mesh following the stroke lines given by the user.

2.3 Blendshapes

Blendshape-based facial animation parameterize the space of possible facial expressions as a linear combination of localized deformations corresponding semantically to individual activation of facial muscles. Formally, the coordinates $\mathbf{x}$ of the face mesh's vertices are obtained as

$$\mathbf{x} = \mathbf{x}_0 + \sum_{i=1}^{N_b} w_i \mathbf{b}_i, \tag{2.1}$$

where $\mathbf{x}_0$ is the neutral configuration, N_b is the number of blendshapes, the vectors $\mathbf{b}_i$ are the *delta-blendshapes* and the scalars w_i are the activation weights for each blendshape, varying from 0 to 1.

The blendshapes paradigm is the dominant paradigm accross the movie industry [98]. This paradigm is attractive for high-quality facial animation since any facial expression can be obtained by enhancing the blendshapes set with a *corrective* shape if necessary. In section 2.3.1 we will see the close relationship between the blendshapes paradigm and the muscles governing facial expressions. In section 2.3.2 we will investigate performance capture methods based on blendshapes. We will further extend this analysis by focusing on techniques building user-specific blendshapes sets to increase the quality of performance capture (section 2.3.3). Our investigation of blendshapes techniques will end with the study of extensions to the classical blendshapes paradigm (section 2.3.4).

2.3.1 Blendshape Principles

Ekman and Friedsen [57, 58] identified that facial expressions could be decomposed as the combination of individual muscle activations. They developed the Facial Action Coding System (FACS) [57, 58], a codification of the muscle activations. Magnenat-Thalmann and colleagues introduced the Abstract Muscle Action procedure [122], a description of the action of facial muscles, similar in spirit to the Facial Action Coding System. The parametrisation of atomic facial movements was standardised as part of the MPEG4-Visual standard [127]: the Facial Animation Parameters (FAPs). Byun and Badler [30] proposed to directly modify the FAPs curves to introduce emotional changes in facial animations. Choe and Ko [43] identified blendshapes as the result of individual muscle contractions and proposed a method to optimize the blendshapes to better match motion capture data.

Havaldar [71] generated performance driven blendshape animation by evaluating the FACS parameters describing motion capture markers movements and retargeting these FACS parameters to a compatible facial rig. Sagar [148] defined a specific FACS space for gorillas to be able to transfer FACS parameters captured from actor Andy Serkis to the face of King Kong. Zhang et al. [193] performed facial landmark tracking on video data, and analyzed the tracked landmark as FAPs parameters that could be used to animate a blendshapes character. Zeng and collaborators [189] used texture classifiers to enhance the Facial Action Unit detection of video-based tracking, with the ability to discriminate cases where geometry alone was not sufficient to determine the facial expression. Seol and Lewis [154] argued that a simple weight transfer between the blendshape FACS basis of the actor and the character's blendshape basis was not sufficient for high quality results. Instead, they proposed to learn a weight mapping between both weight space using radial basis function networks, which enabled artistic control over the result. Ravikumar et al. [145] combined marker-based motion capture with image-based FACS classifier to produce blendshape weights from actor performances.

2.3.2 Facial Performance Capture Based on Blendshapes

The linear formulation underlying blendshapes makes them an easy tool for motion capture. Solving for blendshape weights can often be formulated as a non-negative least-squares solve, for which efficient linear algebra implementations exist. The linearity of the blendshapes model acts as a regularizer, filtering motion capture noise.

Kholgade and coworkers [84] retargeted motion capture markers to blendshape characters by assigning a facial region to markers and blendshapes, allowing to decorrelate the upper face from the eye region and from the mouth region. Ma et al. [117] automatically determined temporally coherent segmentations of the face wherein independent blendshape fittings were performed, enabling expressive and jitter free animations. Seol and colleagues [155] argued that in order to produce animation results editable by artists, it was beneficial to fit blendshapes sequentially, in decreasing order of intensity, as it would produce animation curves closer to the curves artists would produce by hand. Later on, Seol et al. [153] argued that solving for blendshape parameters in the speed domain produced more faithful animations, especially in the presence of priors on the blendshape weights. Bhat and colleagues [16] combined marker-based performance capture with facial contour tracking to improve the fidelity of the blendshape fitting. They also introduced geometric deformation to better fit the data, and transferred the deformation to the target blendshape character for accurate performance transfer. Wang and colleagues [173] augmented blendshape-based facial performance capture with a gaze tracker to be able to animate the eyes of 3d characters in realtime.

Occlusions can be problematic for unconstrained performance capture. Li et al. [103] captured facial animation for users of virtual reality helmets by fitting blendshapes to the visible mouth region and regressing the blendshape weights for the upper part of the face using data from stretch sensors incorporated to the helmet. Liu and collaborators [115] processed audio information with neural networks to infer blendshape weights for performance capture frames where the face was occluded or not visible. Edwards et al. [56] decomposed phonemes into jaw and lip components to animate a FACS-based blendshape model from audio only. Olszewski et al [135] used convolutionnal neural networks to regress blendshape weigths from videos of both the upper and lower parts of the face imaged from a virtual reality headset. Saito and collaborators [150] trained a convolutionnal neural network to determine occluded parts of the face and tuned the facial tracker of Cao et al. [32] to take this occlusion information into account, leading to facial performance capture robust to occlusions. Thies and colleagues [170] augmented their tracker [169] to take into account the presence of a virtual reality headset decorated with markers to infer the correct head orientation.

2.3.3 Automatic Generation of Blendshape Bases

In order to accurately solve for blendshape weights, an accurate blendshapes model for the captured user needs to be available. Therefore, research on blendshapes techniques has been largely focused on the creation of user-specific blendshapes models.

Pighin and colleagues [140, 141] constructed blendshape bases by reconstructing facial

geometry from multiple photographs of an actor performing basic expressions. Liu and colleagues [113] transfer the target blendshape space to the source marker space using the deformation technique of Noh and Neumann [134] and iteratively refine these "marker blendshapes" to ensure the best match with the captured movement. Once this training step is performed, computing blendshape weights on the source markers yields a matching animation on the target blendshapes. Li et al. [104] extended Summner and Popovič's [163] deformation transfer by allowing for example target shapes to be given as input, enabling the transfer of an exising blendshape model to a new face mesh while maintaining artistic control over the result. Ma and coworkers [116] developed a system able to create new blendshape models from a variety of example characters by selecting morphological features in different characters and blending them in the Poisson domain. Neumann et al. [132] showed that performing a sparse and localized factorization of high-fidelity performance capture of facial geometry led to deformation components similar to artistic blendshapes.

Alexander et al. [5] used a Light Stage to capture highly-detailed blendshapes of an actress and were able to produce compelling animation where distinguishing the real performance from the synthetic one was challenging. Blendshapes are defined in the referential frame of the skull. Beeler and Bradley [12] developed a technique to automatically infer the precise location of the skull in high-fidelity facial scans, reducing the need for manual alignment of the scans before a blendshape model could be built.

Leveraging widely available RGBD sensors such as the kinect, Weise et al. [181] presented a realtime performance capture where facial blendshapes of the user would be created from manual markup and example-based facial rigging [104], and be used to track the facial movements of the user from the depth maps. The expression was then transferred to a target blendshape basis with matching shapes. Bouaziz and colleagues [26] improved upon the work of Weise et al. by creating a performance capture system requiring no user calibration. Instead, the specific blendshapes of the user would be modelled on-the-fly while using the system. In a concurrent work, Li and coworkers [105] maintained an incremental PCA of facial expressions to improve the tracking accuracy on users who only calibrated their neutral expressions, and perfomed blendshape fitting on the PCA tracked shapes to get more accurate blendshape weights.

Cao et al. [34] leveraged RGBD sensors and example-based facial rigging to create a consequent bilinear model of facial geometry with respect to variations in identity and expression. Using this model, Cao and colleagues [33] built user specific blendshapes from 2D images, and used these blendshapes in a lightweight boosted regressor to infer blendshape weights from video performances in realtime. Weng and collaborators [184] managed to make the technique fast enough to run in realtime on a mobile phone. Cao and coworkers [32] improved the technique by estimating the user-specific blendshapes on-the-fly. Ichim et al. [75] built user-specific blendshapes from mobile phone videos by deforming a deformation transferred blendshape basis to match the landmarks detected by automated facial tracking methods. Cao and colleagues [35] constructed user-specific blendshapes along with image-based rendering from video, coupled with realtime tracking [32] to enable low-bandwidth transmission of facial animation across a network. Fried and coworkers [65] recovered the identity and expression parameters of photograph subjects to manipulate the

perspective of the image and produce viewpoints under different focals. Casas et al. [37] developed a technique to generate photorealistic blendshape models from a set of facial expressions captured using an RGBD sensor.

Thies and colleagues [168] recovered the identity and expression parameters of multiple individuals using depth and inverse lighting techniques, allowing realtime replacement of the expression of one user by those of another. Garrido et al [67] removed the depth requirement of the method to build user specific facial rigs from videos. Thies and coworkers [169] used a similar image-based technique to perform realtime facial reenactement of videos.

Getting a user-specific blendshape model can increase the precision of performance capture, but such methods are typically limited to estimating the value of about fifty blendshape weights, exposing the resulting animation to issues caused by the linearity of the blendshape model.

2.3.4 Beyond the Linearity of Blendshapes

Indeed, even though the blendshapes paradigm can produce convincing facial animation, it remains an approximation of the true underlying physical process that governs facial expression. In particular, the linearity of the blendshapes is not suited to represent some inherently non linear movements such as the rotation of the jaw. It is however possible to extend the blendshapes paradigm to address some shortcomings.

Lewis et al. [100] formulated Pose space deformation, a technique enabling to specify facial shapes at various locations in the blendshape weight space. Seol and colleagues [156] further enhanced the technique by allowing for localized edits, resulting in finer-grained control.

Not all values of blendshape weights produce reasonable face shapes: only weights between 0 and 1 correspond to valid activations, and the simulateneous activation of certain blendshapes can create unnatural configurations. To circumvent this *conflicting shapes* issue, Lewis et al. [101] introduced editing tools that enabled reaching the desired expression without activating shapes that would conflict with the currently activated shapes. Lewis and Anjyo [99] further improved editing techniques for blendshapes by allowing artists to directly move vertices from the facial mesh, leaving the blendshape weights determination to their automatic system. Blendshape models for highly detailed facial meshes are usually stored as a huge dense matrix despite most coefficients being zero, which is a huge performance penalty. Seo and coworkers [152] developed a specialized sparse matrix structure tailored at efficient storage and GPU usage, enabling realtime animation of very high resolution blendshape models.

It is possible to circumvent the limitations of linearity by introducing the ability to expand the space of possible configurations, either with geometric deformations or by adding new shapes. Joshi et al. [80] fit blendshapes on motion capture data, and deform the mesh using RBF to match the marker positions. This process is done on automatically detected segmentation regions, which increases the expressivity of the model. Kim et al. [86] performed automatic tuning of a blendshape basis while transferring an animation from a source blendshape model to a target. The tuning was enabled by geometrically

transferring "virtual markers" from the source model to the target and sculpting additional blendshapes for expression not well matched by the target blendshape model. Liu and colleagues [110] enhanced the expressive space of blendshape models by automatically segmenting blendshapes into more localized shapes, enabling a finer control over the deformations.

Lewis and collaborators [98] identified blendshapes as a first order Taylor expansion of a high-level muscle activation to geometry function, and called for an investigation of higer-order Taylor expansions. Liu and coworkers [114] generalized the blendshape formulation to cubic polynomials, showing that it enabled to model the non-linear motions of the jaw and of the eyelids. Ichim et al. [77] showed that extending the blendshape formulation to quadratic polynomials was sufficient to represent the results of physically-based facial animation.

2.4 Physical Models

Physically-based facial animation tries to model the formation of facial expressions by simulating the actual physical process as faithfully as possible. Inherently, facial expressions are the result of skin deformation under the actions of facial muscles (see figure 2.1), constrained by contact to the skull and jaw bones. In early works, the skin's elasticity has been represented by mass-spring systems (section 2.4.1), with various modelling tricks to try and account for the volumetric nature of the skin. Later on, finite element methods have enabled very accurate simulation of facial expressions (section 2.4.2). However, the expensive acquisition of physiological data required for such methods has led to the investigation of physical simulation methods taking advantage of pre-existing facial rigs (section 2.4.3). We conclude this section by studying recent works attempting to perform realtime physical simulation (section 2.4.4), and identifying the issues that have prevented so far realtime physical from being applied to facial animation.

2.4.1 Spring-Based Systems

A simple yet effective method to simulate the elastic behavior is the mass-spring system paradigm. The skin is viewed as a collection of mass points, connected by a network of springs. The first usage of mass-spring systems to perform physically-based facial animation was performed by Platt and Badler [143], who used vector muscles to set the face in motion. Waters [178] explored the design of a parametric muscle model influencing vertices near its point of attachment with a distance-based falloff.

Waters and Terzopolous [165, 179] used multiple layers of springs to account for the volume preservation properties of the skin, and a biphasic spring model to account for the non-linear elasticity of the skin. Their system could be animated from image data using an analysis by synthesis approach. Teschner et al. [166] used a similar mass-spring system, but used it to recover the configuration of the skin from skull positions. Kähler and collaborators [92] introduced virtual springs pulling the skinning outwards to simulate volume preservation. Pitermann and Munhall [142] formulated the problem of determining

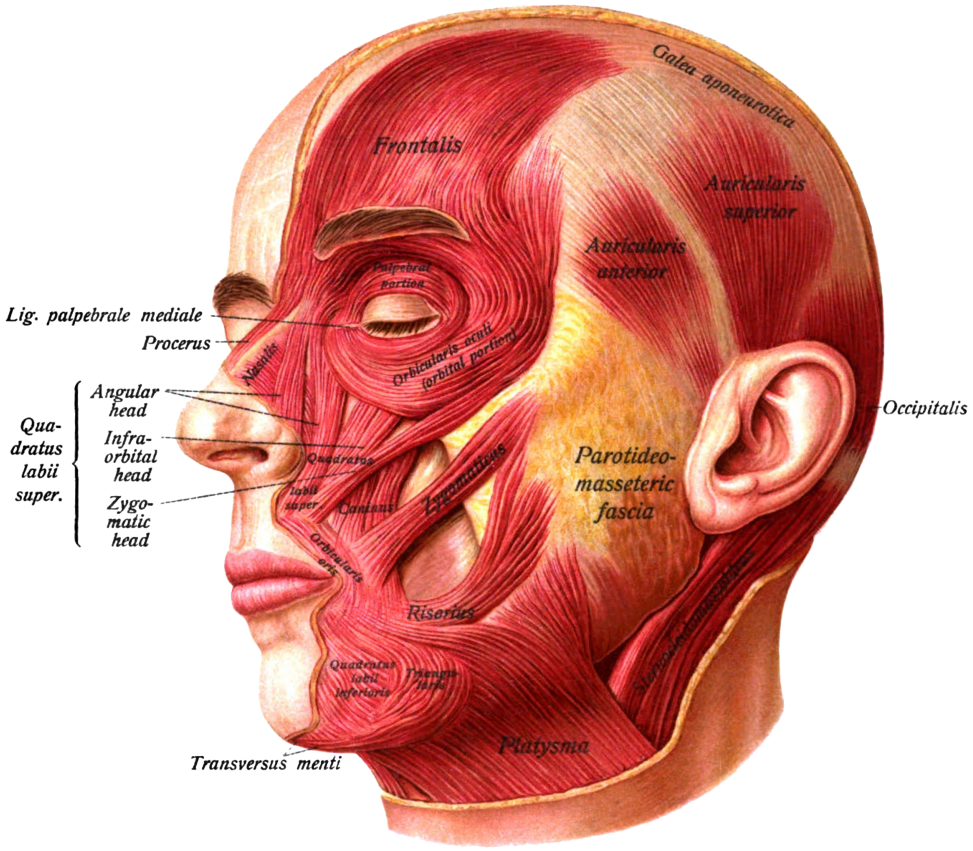


Figure 2.1 – Localization of the facial muscles.

Image in the public domain from Sobotta, J. *Sobotta's Atlas and Text-book of Human Anatomy.* 1909.

the muscle parameters that deform a face mass-spring system in motion as an inverse dynamics problem and generated animation from motion capture markers.

Building a mass-spring facial system with a complex muscle system is a daunting task. To alleviate this issue, Kähler and colleagues [94] developed a method to transfer an existing physical face system to a new character by relying on a dense surface correspondence built from manually specified correspondences and Thin Plate Spline interpolation, a method suited to deform a surface while minimising curvature changes. Later on, Kähler et al. [93] extended their method to be able to reanimate characters created to match the skulls of dead people. Orvalho and colleagues [136] similarly used Thin Plate Spline interpolation to transfer rig elements, among which facial muscle systems. Bui et al. [29] contributed a similar method to adapt a vector muscle model to a new character, and focused on adapting the muscles' shapes and intensity to enhance the quality of the results. Fratarcangeli [63] also constructs a physical face system by adapting a pre-existing skull and muscle model to a new face mesh, and calibrates the muscle contractions to match the deformations specified by the FAPs parameter system. Fanelli and Fratarcangeli [61] then combined this

model with facial feature tracking reinterpreted as FAPs to produce performance driven animation. Morishima et al. [126] automatically determined the muscle parameters of their system using a neural network trained to regress from neutral to expression optical flow fields. Gao and colleagues [66] developed a fully automated technique for transferring an existing vector muscle model to a new facial mesh.

2.4.2 Finite Element Systems

The finite element method enables the modelisation of more diverse models of skin elasticity than what can be accomplished with mass-spring systems. Consequently, the rise in processing power and the availability of high-quality finite element frameworks enabled research investigating the modelling of highly accurate elastic models of the skin and the muscles.

Essa and colleagues [59] cast facial motion capture as a control problem where the deformation of a finite element facial mesh must match the deformations of the performer as measured by optical flow. Choe et al. [44] show that modelling the face as a surfacic finite element model with linear elasticity of the skin can be reinterpreted as the linear combination of the shapes obtained by activating a single muscle. Their approach thus provides for a physical ground to the blendshape formulation. In concurrent works, Keeve et al. [83] and Koch et al. [89, 88] improved the realism of physical simulation of the face using physiological data and finite element simulation. Basu and colleagues [11, 10] used motion capture of the lips and the finite element method to derive a statistical model of the lip that could be fitted to new images of lips to recover their 3D shape. Chabanas et al. [38] took advantage of finite element facial simulation to predict the result of facial surgery. Gérard and colleagues [68] investigated the properties of human skin to derive a proper constitutive model for finite element simulation, and concluded that for best results, the skin should be modelled as an hyperelastic material. Bickel and coworkers [18] developed a non intrusive technique to capture and model the elastic properties of human skin, allowing to specialize a physical system to a subject. Nazari et al. [131] studied the muscle models adapted to producing convincing orofacial gestures in a finite element simulation. Berar and colleagues [14] leveraged a statistical model to reconstruct the skull and facial tissues from CT scans. Barbarino et al. [7] produced a finite element model of the face based on medical range imaging data. To reduce the need for physiological data acquisition, Aina et al. [2, 3] adapted a generic skull to a neutral facial mesh and automatically derived the location of muscles and of the derma. Cong and collaborators [47] adapted an existing finite element model to new facial meshes by automatically determining the location of key geometric facial features and deforming the existing physical model into the new mesh.

Sifakis et al. [158] built a highly accurate finite element model on top of medical scan data, and devised an algorithm to compute muscle actuation parameters from motion capture markers. Sifakis and colleagues [159] further extended this method by generating speech sequences from phoneme information.

Finite element simulation is generally very expensive. To alleviate this issue, Kim and James [87] developed a method to progressivley compute a linear approximation of the deformable space of the system, allowing to skip some physical simulation computations

when the performed movement belongs to a part of the deformation space that was explored earlier on.

2.4.3 Physical Simulation and Rigging

While physical simulation methods analog to Sifakis et al. [158] produce highly accurate results, their application requires the expensive acquisition of medical data and the manual building of a complex simulation model from this data. It is however interesting to leverage the qualities of physical simulation while requiring only already available artistic data.

Hahn et al. [69] devised a technique enabling the addition of physics-based secondary motion to traditionnal rigs. Li and colleagues [109] controlled a finite element model simulation to match the movements of existing triangle mesh animations, and designed vertices masks to allow parts of the facial mesh to be physically animated while the rest ofthe mesh remained controlled by artistic input. Yu et al. [188] augmented mesh-based facial animation with physics based animation of the tongue and teeth. Cong and coworkers [48] described a method to provide artistic control over the results of a physical simulation, allowing artists to position the muscles to reach the desired input. They further tightened the gap between physical simulation and other facial animation methods by showing that applying blendshape techniques or deformation transfer to the muscle layer was possible and produced visually pleasant results.

Ma and colleagues [119] introduced the idea of building physical simulation models from existing blendshape models. They built a mass-spring system from the mesh and interpolated the rest lengths of the springs based on the blendshape weights and the lengths of the edges in the blendshapes. Ichim et al. [77, 76] generalized this idea to volumetric blendshapes and showed that it could correctly model the motion of the jaw. Kozlov and coworkers [90] leveraged physical simulation to enrich blendshape animations with secondary motion caused by inertial effects.

2.4.4 Realtime Physical Simulation

The Projective Dynamics framework [111, 24] is a robust and fast framework for systems with constant constraints, but has not yet been successfully applied to realtime facial animation. Realtime physically-based facial animation is challenging: state of the art methods can simulate facial performances at around one frame per second [77].

The first obstacle to performance is the continuously changing constraints due to lips contacts that prevents efficient physical simulation techniques such as Projective Dynamics to use a pre-factorized solver. Recent works handled changing systems by using iterative solvers such as an accelerated Jacobi [174, 176] or a parallel Gauss-Seidel [64]. However, these methods are designed to take advantage of the computational power of the GPU and are likely to be less efficient on the CPU. Liu et al. [112] generalized Projective Dynamics to enable simulation of hyperelastic materials, improving the convergence of Projective Dynamics in the process, but their computational performance still suffers when constraints change. In section 6.2, we will present an iterative solver integrated into the L-BFGS acceleration technique of Liu et al.

The second obstacle to performance lies in the computation of the forces projections. In particular, most volumetric forces require performing numerous SVDs, which are computationally expensive, even with the fast SVD technique of McAdams and colleagues [124]. Fast polar decomposition methods have been used to simulate volumetric elasticity [147, 174, 128], however the polar decomposition cannot be used to simulate volume preservation, which is crucial to simulate the behavior of facial skin.

We propose to take advantage of the iterative nature of Projective Dynamics to memoize SVD computations, using a Taylor approximation to enhance the precision of our memoization strategy (section 6.3). In this respect, our method shares some similarity with the warm-started fast polar decomposition of Rivers and James [147]. However, our method differs in two important aspects: we compute a full singular value decomposition, and our Taylor analysis gives rise to an adaptive memoization scheme allowing to completely avoid some SVD computations.

Conclusion

We have so far divided facial animation techniques with respect to their degree of abstraction of the natural process underlying facial expressions. We can form new insight by attempting to describe the *interactions* between these categories.

Given the variety of rigging techniques, data-driven methods remain the most practical to convert one parameterization of facial animation into another, and will thus find its place in most performance capture setups. Blendshapes techniques heavily rely on deformers methods to build user-specific blendshapes models. Similarly, transferring a physical system relies on deformers techniques. Deformation models are also often close-form solutions to simple physical models.

Nevertheless, blendshapes techniques and physical simulation techniques present a much closer relationship. Both find their roots in a muscular explanation of facial expressions, and blendshapes can even be seen as the exact solution of some physical systems, under the assumption that the skin has a linear elasticity [44]. This close relationship set a new area of research, trying to combine the ease of modelling provided by blendshapes with the realistic skin deformations offered by physical simulation. Our work is set in this research area opened by Ma and colleagues [119], however we bring an additional insight by interpreting blendshapes as a muscular component. Another key contribution of our method resides in the realtime performance associated with our physical simulation method.

Chapter 3

Background

Contents

This thesis presents a practical facial animation method that takes advantage of the wide availability of blendshapes model while extending their capability to represent accurate facial motions. In this chapter, we introduce relevant background notions necessary to the presentation of our work. We first present the blendshapes framework (section 3.1). In section 3.2 we detail the physical simulation framework used in our work: Projective Dynamics [24]. We finish this chapter by introducing the formalism used to describe mesh deformation (section 3.3).

3.1 Blendshapes

The blendshapes framework approximates the deformations of the facial skin using an affine model. This affine model can be identified as the linear combination of the neutral expression and a set of facial expressions composed by the results of the activation of single facial muscles. An example of blendshape shapes is given in figure 3.1. In this section, we study the mathematical properties of blendshapes model (section 3.1.1), and investigate how these properties are applied to performance capture (section 3.1.2). We conclude this section

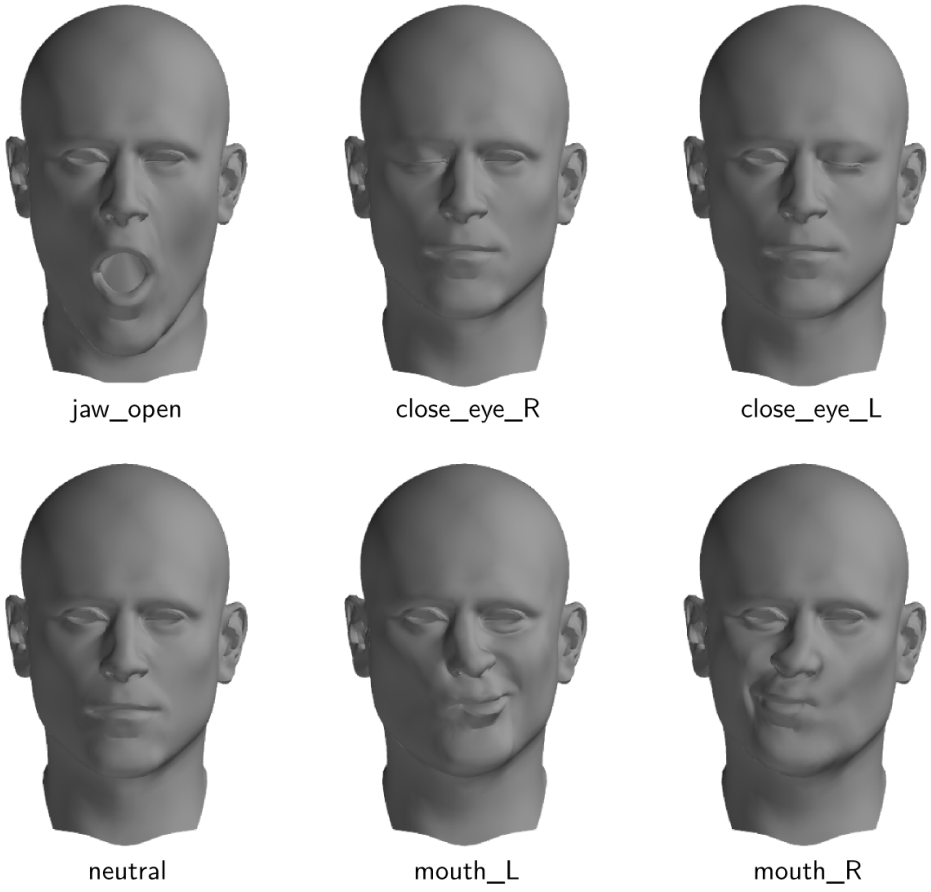


Figure 3.1 – Components of a basic blendshapes model. The neutral expression, and the unit activation of 5 basic expressions are shown.

by studying extensions to the blendshapes model trying to overcome the shortcomings of a linear representation (section 3.1.3).

3.1.1 Properties of the Blendshapes Framework

Blendshapes are widely represented in the delta-blendshapes formulation, namely the coordinates $\mathbf{x}$ of the $3 \times N_v$ vector containing the coordinates of the face mesh's vertices are obtained as

$$\mathbf{x} = \mathbf{x}_0 + \sum_{i=1}^{N_b} w_i \mathbf{b}_i, \tag{3.1}$$

where $\mathbf{x}_0$ is the neutral configuration, the vectors $\mathbf{b}_i$ are the *delta-blendshapes* and the scalars w_i are the activation weights for each blendshape, varying from 0 to 1. This formulation is equivalent to performing interpolation in the set of facial meshes comprising the

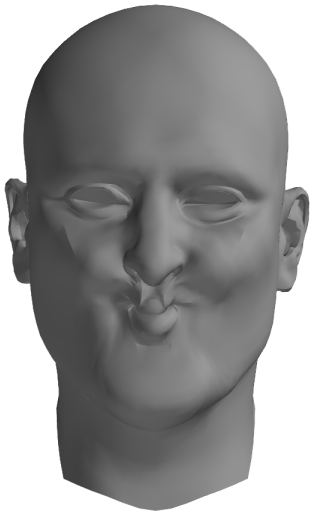


Figure 3.2 – Conflict between two blendshapes. The mouth_L and mouth_R shapes cannot be activated simultaneously.

neutral mesh $\mathbf{x}_0$ and the expression meshes $\mathbf{x}_0 + \mathbf{b}_i$. Indeed, we can transform equation 3.1 to show $\mathbf{x}$ as a linear combination of these shapes:

$$\mathbf{x}_0 + \sum_{i=1}^{N_b} w_i \mathbf{b}_i = \left(1 - \sum_{i=1}^{N_b} w_i\right) \mathbf{x}_0 + \sum_{i=1}^{N_b} w_i \left(\mathbf{b}_i + \mathbf{x}_0\right). \tag{3.2}$$

Consequently, it is equivalent to refer to a blendshape as either the delta-blendshape $\mathbf{b}_i$ or the full-blendshape $\mathbf{x}_0 + \mathbf{b}_i$, as long as we keep a consistent formulation. Throughout this thesis, we will use the delta-blendshapes formulation, and will, by abuse of notation, display full-blendshapes while refering to delta-blendshapes, as was done in figure 3.1.

The blendshapes equation can be conveniently represented in matrix form. By stacking the blendshape vectors $\mathbf{b}_i$ in the column of a matrix $\mathbf{B}$ and assembling the weights w_i in a vector $\mathbf{w}$, equation 3.1 reads

$$\mathbf{x} = \mathbf{x}_0 + \mathbf{B}\mathbf{w}. \tag{3.3}$$

While this matrix formulation may lead to analogies between the blendshapes formulation and a PCA of the expressions, there are a few key differences. First and foremost, the matrix $\mathbf{B}$ is not orthogonal, meaning that the weights w_i are not independent. Furthermore, the blendshapes vectors $\mathbf{b}_i$ have *local* support, and are subject to semantic interpretation. This property makes blendshape manipulation intuitive, and is an important reason for the success of blendshapes in industry [98].

In practice, not all values of $\mathbf{w}_i \in [0, 1]$ produce valid results. Some shapes are *conflicting* shapes, and their simultaneous activation results in unnatural forms. In figure 3.2, the

Figure 3.3 – Linearity of the trajectory of a vertex while activating the jaw opening blendshape.

shapes mouth_L and mouth_R were both set to unit activation, resulting in a collapse of the mouth. Some of these conflicts can be prevented by the rig, with controllers preventing the simultaneous activation of obviously conflicting shapes.

Another shortcoming of blendshapes comes from the intrinsic linearity of the model. Some facial expressions, mostly those involving the jaw, comprise a rotational component that cannot be represented exactly by a linear model. This phenomenon is illustrated in figure 3.3. This issue also affects the eyelids, which should follow the spherical shape of the eye but cannot due to the linearity of blendshapes. In some blendshapes implementation, this issue can be circumvented by specifying intermediate shapes targets for some blendshapes, effectively turning the blendshapes model into a piecewise affine model.

3.1.2 Blendshapes-Based Performance Capture

The simplicity of the blendshapes framework makes it excellent for performance capture. Indeed, its linear model can be efficiently optimized to match 3D motion capture data. Consider a typical marker-based performance capture setup. A set of user-specific blendshapes for an actor has been crafted. The actor performs with markers on his face while wearing a helmet comprising multiple cameras. Computer vision techniques can then be used to recover the 3D location of each marker throughout the performance. After rigid stabilization, this yields a set of markers $\mathbf{t}$ in the same referential frame as the blendshapes of the actor. The blendshapes activations corresponding to an animation frame can then be retrieved by solving the non-negative least-squares problem

$$\mathbf{w}^* = \underset{\mathbf{w} \geq 0}{\arg\min} \left\| \mathbf{x}_0 + \mathbf{B}\mathbf{w} - \mathbf{t} \right\|^2 . \tag{3.4}$$

However, this solve alone can result in the activation of conflicting blendshapes. Seol and colleagues [155] noticed that artists typically created their animations by activating only a few number of blendshapes at a given time. They therefore proposed a solving technique that would encourage the sparsity of the weight vector: they ordered the blendshapes by the magnitude of the changes they caused to the face mesh, and solved for weights one at a time, starting by the most influencial weight. Another technique to avoid conflicting shapes was devised by Bouaziz et al. [26], who augmented equation 3.4 with a sparsity inducing L1 prior:

$$\mathbf{w}^* = \underset{\mathbf{w} \geq 0}{\arg\min} \left(\|\mathbf{x}_0 + \mathbf{B}\mathbf{w} - \mathbf{t}\|^2 + \|\mathbf{w}\|_1 \right). \tag{3.5}$$

In some motion capture setups, no user-specific blendshapes exist, but the target character is blendshapes-based. In this case, it remains possible to use the solving methods outlined precedently. Seol et al. [153] devised a technique to create virtual markers matching the target character's morphology. To apply this technique, one needs to select vertices on the target mesh corresponding to the markers on the actor's face. Then, a radial basis function network with a linear kernel is learnt to map the position of the markers on the actor's neutral expression to the position of the selected vertices on the target mesh's neutral. The mapping learnt is a transformation between the morphology of the actor and the morphology of the target. The linear kernel ensures that the learnt mapping is a thin plate spline [136], which guarantees that the change of curvature caused by the mapping function is minimal. This mapping can then be used to transfer the sequence of markers to the reference frame of the target blendshapes, enabling the previous solving techniques to be used.

In section 4.2, we will see that these performance capture techniques can be extended to work with physically-based facial animation.

3.1.3 Extensions to the Blendshapes Framework

The linearity inherent to the representation of blendshapes is not suited for an exact representation of facial expressions. Several approaches exist to overcome the issues brought by linearity. A popular paradigm is the idea of *corrective shapes*, a set of specialized blendshapes designed to be activated to compensate for the artifacts caused by the simultaneous activation of conflicting shapes. In this paradigm, the activation of the correctives is automated by specifying a location in the weight space where the corrective should be active, and relying on a radial basis function network with a gaussian kernel, and hence a local support, to smoothly determine the activation of the corrective shape in the neighborhood of the chosen location [100, 156].

Another extension to the blendshapes framework comes from Lewis and colleagues [98]. They identified the blendshapes approach as a first order Taylor expansion of an hypothetized facial animation process, and proposed to extend the formulation by including

higher orders. For instance, an order two expansion can be formulated as:

$$\mathbf{x} = \mathbf{x}_0 + \sum_{i=1}^{N_b} w_i \mathbf{b}_i + \sum_{i=1}^{N_b} \sum_{j=i+1}^{N_b} w_i w_j \mathbf{b}_{ij}. \tag{3.6}$$

The additional $\mathbf{b}_{ij}$ term can be understood as a second order blendshape, used to compensate artifacts arising from the simultaneous activation of blendshapes $\mathbf{b}_i$ and $\mathbf{b}_j$. We will see in section 4.3 that our work can be related to these Taylor-based extensions of the blendshapes framework.

3.2 Projective Dynamics

Recently, Bouaziz and colleagues [24] developed Projective Dynamic, an efficient and robust framework for physical simulation. In this section, we recall the principles underlying physical simulation and describe the Projective Dynamics framework.

Newton's second law of motion states that the action of forces on a system equals its mass times its acceleration. In physical simulation, for a mesh described by the location of the vertices $\mathbf{x}$, this second law translates as

$$\mathbf{f} = \mathbf{M}\mathbf{a}, \tag{3.7}$$

where $\mathbf{M}$ is a matrix containing the discretization of the mass in the object, $\mathbf{f}$ is the sum of forces acting on the object, and $\mathbf{a}$ is the acceleration. This equation can be integrated numerically using an implicit Euler scheme with a timestep h:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + h\mathbf{v}_{t+1} \tag{3.8}$$

$$\mathbf{v}_{t+1} = \mathbf{v}_t + h\mathbf{M}^{-1}\mathbf{f}_{t+1}, \tag{3.9}$$

with the t subscript identifying the timestep and $\mathbf{v}$ the speed of the object. The forces acting on the object can be divided into two categories: external forces $\mathbf{f}_{\text{ext}}$ do not depend upon the configuration of the object (for instance, the gravity force, wind force), while internal forces $\mathbf{f}_{\text{int}}$ are a function of $\mathbf{x}$. Bouaziz et al. observed that the latter are generally conservative forces, implemented as a nonlinear function of the strain computed on geometric primitives. They noticed that this process was equivalent to formulating the potential as a function of the distance of the current configuration to the set of configurations with zero strain (undeformed configurations). They argued that most of the nonlinearity of the potential was contained in the projection operation necessary to compute the distance, and formulated Projective Dynamics potentials as functions of the form:

$$W_k(\mathbf{x}) = \frac{w_k}{2} \|\mathbf{G}_k\mathbf{x} - \mathbf{H}_k\mathbf{p}_k\|^2 \tag{3.10}$$

where $\mathbf{G}_k$ and $\mathbf{H}_k$ are constant differential operators, for instance Laplacian matrices or deformation gradient operators, and $\mathbf{p}_k$ is a projection of $\mathbf{x}$ on the manifold of undeformed

configurations. This formalism is expressive enough to represent springs, bending preservation, volumetric elasticity, volume preservation, or even example-based deformation models. In chapter 5, we will investigate the potentials useful for physically-based facial animation.

Following Baraff and Witkin [123], the time integration can be recast as an energy minimization problem. Indeed, one can combine equations 3.8 and 3.9 as:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + h\mathbf{v}_t + h^2\mathbf{M}^{-1}\left(\mathbf{f}_{\text{ext}} - \sum_k \nabla W_k(\mathbf{x}_{t+1})\right). \tag{3.11}$$

Solving equation 3.11 is equivalent to minimizing

$$E_{\text{PD}}(\mathbf{x}_{t+1}) = \frac{1}{2h^2}\|\mathbf{M}^{1/2}(\mathbf{x}_{t+1} - \mathbf{y}_t)\|_F^2 + \sum_k W_k(\mathbf{x}_{t+1}), \tag{3.12}$$

where $\mathbf{y}_t = \mathbf{x}_t + h\mathbf{v}_t + h^2\mathbf{M}^{-1}\mathbf{f}_{\text{ext}}$ is the predicted state of the system in the absence of internal forces. This minimization problem can be seen as a balance between the momentum of the system, represented by $\mathbf{y}_t$, and the elastic energy contained in the internal forces.

The classic method to solve this system is to use a Newton's method to minimize equation 3.12, using the predicted state $\mathbf{y}_t$ as a starting point. This method is however very computationaly intensive, since a new Hessian matrix has to be inverted for each iteration of Newton's method. In Projective Dynamics, the particular form of the potentials $\mathbf{W}_k$ enables a more efficient solve. Indeed, once the projections $\mathbf{p}_k$ have been computed, equation 3.11 can be rewritten as a linear solve:

$$\left(\frac{\mathbf{M}}{h^2} + \sum_k w_k\mathbf{G}_k^T\mathbf{G}_k\right)\mathbf{x}_{t+1} = \frac{\mathbf{M}}{h^2}\mathbf{y}_t + \sum_k w_k\mathbf{G}_k^T\mathbf{H}_k\mathbf{p}_k, \tag{3.13}$$

which can be further simplified as

$$\mathbf{x}_{t+1} = \left(\frac{\mathbf{M}}{h^2} + \mathbf{Z}\right)^{-1}\left(\frac{\mathbf{M}\mathbf{y}_t}{h^2} + \mathbf{Y}\mathbf{p}\right), \tag{3.14}$$

with $\mathbf{Z} = \sum_k w_k\mathbf{G}_k^T\mathbf{G}_k$, $\mathbf{p}$ a stacking of all the $\mathbf{p}_k$ and $\mathbf{Y} = \sum_k \mathbf{G}_k^T\mathbf{H}_k\mathbf{S}_k$, $\mathbf{S}_k$ being a selection matrix such that $\mathbf{p}_k = \mathbf{S}_k\mathbf{p}$.

Computing the projections $\mathbf{p}_k$ can be seen as decreasing the energy E_{PD}, and solving equation 3.14 also decreases the energy. Thus, alternating these two steps, respectively named the *local step* and the *global step* consistently decrease E_{PD}.

The key point to the performance of Projective Dynamics is that the global step matrix

$$\mathbf{A} = \left(\frac{\mathbf{M}}{h^2} + \mathbf{Z}\right) \tag{3.15}$$

is symmetric positive definite (SPD), and constant as long as the support of internal forces does not change. This property allows one to precompute the global step matrix inversion through a sparse Cholesky decomposition, which triggers excellent computational performance up to realtime rates. As we will see in chapter 6, the condition of constant support for internal forces is not valid when simulating physical facial animation, requiring a different solving technique (section 6.2).

3.3 Mesh Deformation

In this section we review the description of mesh deformation by deformation gradients and Singular Value Decompositions (SVDs). These two concepts are indeed key to build physically accurate elastic models. We only cover parts relevant for this thesis, for a complete reference on 3D elastic materials please refer to the work by Sifakis and Barbič [157].

While this thesis is written with the goal of having globally consistent notation, this section's notations slightly differ. In this section we only consider local quantities, hence vectors such as $\mathbf{x}$ or $\mathbf{c}$ denote 3-dimensional points. Subscripts $\mathbf{x}_i,\ i \in [0, 2]$ are used as the coordinate projection operator.

3.3.1 Deformation Gradient

In physical simulation, objects generally possess a *rest state*, an undeformed configuration that will serve as a reference. The deformation of an object can then be given by a deformation mapping $\mathbf{f}$ that maps the rest configuration $\mathbf{c}$ to the deformed configuration $\hat{\mathbf{c}} = \mathbf{f}(\mathbf{c})$. The *deformation gradient* $\mathbf{F}$ is the Jacobian matrix of the deformation $\mathbf{f}$:

$$\mathbf{F} = \begin{pmatrix} \frac{\partial \mathbf{f}_0}{\partial \mathbf{c}^0} & \frac{\partial \mathbf{f}_0}{\partial \mathbf{c}^1} & \frac{\partial \mathbf{f}_0}{\partial \mathbf{c}^2} \\ \frac{\partial \mathbf{f}_1}{\partial \mathbf{c}_0} & \frac{\partial \mathbf{f}_1}{\partial \mathbf{c}_1} & \frac{\partial \mathbf{f}_1}{\partial \mathbf{c}_2} \\ \frac{\partial \mathbf{f}_2}{\partial \mathbf{c}_0} & \frac{\partial \mathbf{f}_2}{\partial \mathbf{c}_1} & \frac{\partial \mathbf{f}_2}{\partial \mathbf{c}_2} \end{pmatrix}. \tag{3.16}$$

The deformation gradient provides a local linear approximation to the deformation. Since we work with meshes, we need a discrete version of this operator. We use the common discretization option of computing a deformation gradient matrix per tetrahedral element of a tetrahedral mesh. For a tetrahedron specified by its four vertices $\mathbf{x}^0, \mathbf{x}^1, \mathbf{x}^2, \mathbf{x}^3$, we define the edge matrix as

$$\mathbf{E} = \begin{pmatrix} \mathbf{x}_0^1 - \mathbf{x}_0^0 & \mathbf{x}_0^2 - \mathbf{x}_0^0 & \mathbf{x}_0^3 - \mathbf{x}_0^0 \\ \mathbf{x}_1^1 - \mathbf{x}_1^0 & \mathbf{x}_1^2 - \mathbf{x}_1^0 & \mathbf{x}_1^3 - \mathbf{x}_1^0 \\ \mathbf{x}_2^1 - \mathbf{x}_2^0 & \mathbf{x}_2^2 - \mathbf{x}_2^0 & \mathbf{x}_2^3 - \mathbf{x}_2^0 \end{pmatrix}. \tag{3.17}$$

This matrix contains the vectors $\mathbf{x}^j - \mathbf{x}^0$ that form a local reference frame around the tetrahedron. Given the edge matrix $\mathbf{E}$ of the undeformed configuration and $\hat{\mathbf{E}}$ the edge matrix of the deformed configuration, the tetrahedron's deformation gradient is

$$\mathbf{F} = \hat{\mathbf{E}}\mathbf{E}^{-1}. \tag{3.18}$$

In other words, the discretized deformation gradient is the change of basis matrix between the local frames associated to the deformed and undeformed configurations. Note that this formulation highlights the fact that the deformation gradient is insensitive to global translations.

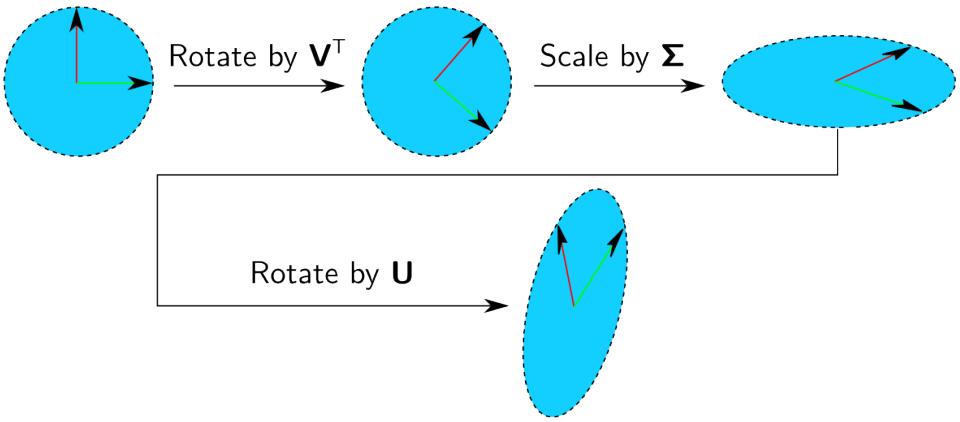


Figure 3.4 – Geometric interpretation of the SVD of a 2×2 matrix $\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$.

3.3.2 Singular Value Decomposition

The local properties of the deformation can be determined from the *singular values* of the deformation gradient. We recall the definition of the Singular Value Decomposition: the SVD of a matrix $\mathbf{M}$ is the decomposition

$$\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T, \tag{3.19}$$

where $\mathbf{\Sigma}$ is a diagonal matrix containing the singular values, and $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices. The SVD exists for every matrix, and is unique provided that an appropriate convention on the sign and order of diagonal values of $\mathbf{\Sigma}$ and the determinants of $\mathbf{U}$ and $\mathbf{V}$ is chosen. In physics, we consider the SVD of 3×3 matrices, and choose the convention that $\mathbf{U}$ and $\mathbf{V}$ should be rotation matrices.

With this convention, the deformation can be decomposed between a purely rigid transformation by the rotation matrix $\mathbf{U}\mathbf{V}^T$ and a local anisotropic scaling along axes defined by the $\mathbf{V}$ rotation. We illustrate this interpretation in 2 dimensions in figure 3.4. Thus, a deformation gradient with all singular values equal to 1 will represent a locally rigid motion. Similarly, a deformation gradient verifying $\det \mathbf{\Sigma} = 1$ will locally preserve the volume. A single negative singular value indicates a transformation with a reflexion component.

In physical simulation, a common model for internal forces is the model of hyperelastic materials, where the potential energy associated with a deformed configuration is a function of the deformation gradient. In general, the simulation of isotropic material can be performed by using potentials that are functions of the singular values of the deformation gradient. Liu and colleagues [112] showed that the Projective Dynamics potentials introduced in section 3.2 could be cast into this framework.

The computation of SVDs of the deformation gradients is thus a crucial part in physical simulation pipelines. However, SVDs are computed using an expensive iterative process, which often proves to be a bottleneck in physical simulations. While highly-optimized

implementations can help reduce the computational cost [124], this remains an expensive process. In section 6.3, we will present an approximation and memoization strategy that can greatly reduce the costs of SVD computations.

Chapter 4

Blendforces: Unifying Blendshapes and Physical Simulation

Contents

In this thesis we propose to enhance the widespread blendshapes paradigm with physical simulation. This chapter presents the main idea behind our framework. The core of our method is based on a reinterpretation of delta-blendshapes as a basis of forces, which allows us to cast a blendshapes character in a physical simulation framework (section 4.1). We will show how our framework can be used to produce physically-based facial animation from performance capture (section 4.2). We will conclude this chapter by demonstrating how the blendforces concept can be seen as a generalization of the blendshapes representation that can be used to more closely match the true facial deformations (section 4.3).

4.1 Blendshapes as Forces

Blendshapes are the dominant paradigm for producing facial animation. However, the linearity of the framework is not without defaults. Some facial movements are inherently non-linear, and conflicting shapes can hinder the intuitivity of the paradigm. We will describe in section 4.1.1 the issues arising from the linear formulation of blendshapes, and will present in section 4.1.2 our *blendforces* framework designed to overcome these issues.

4.1.1 Blendshapes Shortcomings

As we have seen in section 3.1.1, blendshapes are an affine paradigm which maps a vector of activation weights $\mathbf{w}$ to a facial configuration $\mathbf{x}$ using the equation $\mathbf{x} = \mathbf{x}_0 + \mathbf{B}\mathbf{w}$. The columns of the matrix $\mathbf{B}$ typically represent the effect of activating a single muscle, in the spirit of the Facial Action Coding System [57]. However, in practice limiting the set of blendshapes to this intuitive set is not enough to reach high-quality facial animations. Indeed, not only are some facial movements, such as jaw opening, inherently non-linear, but combining blendshapes linearly can produce non-intuitive artifacts. Moreover, complex phenomenons such as the interaction of two lips pressing against each other cannot be represented in a simple linear model. These issues are routinely circumvented by sculpting additional, corrective blendshapes which are meant to be activated only around specific weight regions. 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 $C(\mathbf{B})$. This can be used to encode the non-linearity of the skin deformation, but this comes at the expense of the semantic meaning of FACS-based blendshape weights: reaching these new configurations will necessitate modifying the weights affected to the FACS-based blendshapes. Effectively, corrective blendshapes do not solve the problem of representing non-linear motions, they displace the problem by complexifying the space of valid blendshapes weights.

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.

4.1.2 Blendforces

We propose a new paradigm building upon the blendshapes paradigm, and bridging together the *data-based* approach of blendshapes with *physically-based* approaches. We notice that, with notable exceptions such as Seol et al. [153], blendshapes-based works typically consider blendshapes solves as a quasi-static process, treating consecutive motion capture frames as independent. On the contrary, we believe that the formation of facial expressions needs to be represented in a *dynamic* framework. For instance, the sticky lips phenomenon cannot

be explained without a notion of time: lips stick together because they have collided before. Traditionnaly, this missing dynamic component was retrieved from motion capture data, but for a subtle phenomenon such as sticky lips, capturing it is extremely challenging.

In chapters 2 and 3, we noticed that blendshapes entertained a special relationship with physically-based facial animation methods. Typical FACS-based blendshapes sets are built to reflect the individual actions of muscles. From a geometric perspective, the delta-blendshapes point towards the configuration that would be obtained with a single muscle activated. Based on these observations, we form a new framework for facial animation, wherein we interpret delta-blendshapes as a basis of forces, the *blendforces*, that can be used to steer the deformation of a facial mesh in a physical simulation. From this perspective, we no longer interpret blendshapes as a prior on the possible face configuration, but as a prior on how the face can deform through time.

In our framework, the face is represented as a mesh given by its coordinates $\mathbf{x}$, subject to Newton's second law of motion:

$$\mathbf{M}\ddot{\mathbf{x}} = \mathbf{f}_{\text{ext}} + \mathbf{f}_{\text{int}}, \tag{4.1}$$

where $\mathbf{M}$ is a diagonal mass matrix, $\mathbf{f}_{\text{int}}$ is the sum of internal forces that react to the deformation of the face mesh, and $\mathbf{f}_{\text{ext}}$ is the sum of external forces that put the face into motion. We will discuss internal forces in chapter 5, and for now suppose the existence of "realistic" internal forces. External forces typically comprise gravity, and the blendforces that represent the activations of individual muscles:

$$\mathbf{f}_{\text{ext}} = \mathbf{f}_{\text{gravity}} + \mathbf{B}\mathbf{u}, \tag{4.2}$$

where the $\mathbf{B}\mathbf{u}$ term represent the blendforces term, lying in the span of the blendshapes matrix.

Let us briefly compare this formulation to the delta-blendshapes equation $\mathbf{x} = \mathbf{x}_0 + \mathbf{B}\mathbf{w}$. If we assume a uniform unit mass and ignore gravity and internal forces, the delta-blendshapes equation and the integrated blendforces of equation (4.1) can be related by choosing $\mathbf{u} = \ddot{\mathbf{w}}$ on every time step where $\mathbf{w}$ is twice differentiable. In other words, given appropriate boundary conditions on $\mathbf{w}$ and $\dot{\mathbf{w}}$, blendforces without internal forces are equivalent to blendshapes. The key difference between blendshapes and our proposed blendshapes paradigm thus lies in the presence of internal forces that shape the space of possible facial configurations.

Indeed, when using the blendshapes paradigm it is common to design a prior on the space of blendshape weights $\mathbf{w}$ aimed at restricting blendshapes solves to plausible facial configurations. This prior is generally accounted for by some loss function on various norms of $\mathbf{w}$. Namely, we've seen in section 3.1.1 that the sparse nature of the blendshapes encoding led to the use of L1 priors. Statistics-based (PCA) priors can also be used. The problem is even more salient from a temporal perspective: apart from smoothing energies, blendshapes methods typically do not encode the dynamics of facial expressions.

In the blendforces framework however, the space of possible physical configurations is encoded in the formalism. The blendforces term, lying in the span of the blendshapes matrix, forms a strong prior on the motions that can be applied to the face. This prior builds

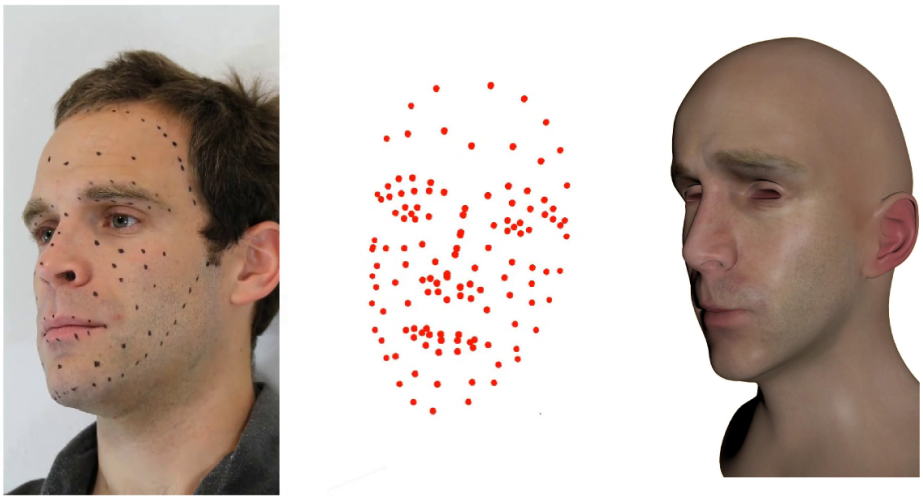


Figure 4.1 – Our performance capture setup. We use marker points tracked from an actor's performance to animate a physical system built from user-specific blendshapes.

on the original insight of a FACS basis: facial motions are the result of a small number of individual muscle actions. Moreover, internal forces can be designed to restrain the system from attaining degenerate configurations, and physical simulation enforces a coherent equilibrium between these forces through time.

4.2 Performance Driven Physical Simulation

In the previous section we introduced the idea of producing facial animation by interpreting it as the evolution of a face physical system whose deformations are driven by a set of actuators derived from blendshapes that we named *blendforces*. In this section we present a method to compute the blendforces actuations from performance capture data. Our method will compute a sequence of blendforces parameters $\mathbf{u}$ that best fit the trajectory of a sparse set of markers. We propose an *online* control method, where the actuation parameters at timestep t are determined from the state of the physical system at timestep $t-1$ and the current markers.

In our setup, we consider an actor performing with markers on his face, and suppose we have user-specific blendshapes for this actor (see figure 4.1). The markers are tracked using the commercial software Performer [55], yielding a sequence of vectors $\mathbf{d}_t$. Our position-control framework 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

Algorithm 1: Marker-based blendforces animation

$\mathbf{x}_1, \mathbf{v}_1 \leftarrow \mathbf{x}_0, 0$;
for $t \in [2, N_{frames}]$ **do**
  $\mathbf{x}_t \leftarrow \mathbf{x}_{t-1} + h\mathbf{v}_{t-1} + h^2\mathbf{M}^{-1}\mathbf{f}'_{\text{ext}}$;
  for *numRelinearizations* **do**
    $\mathbf{p} \leftarrow$ linearizeForces($\mathbf{x}_t$);
    compute $\mathbf{\Phi}_t$ and $\mathbf{q}_t$;
    $\hat{\mathbf{u}}_t \leftarrow$ nnlsSolve($\mathbf{\Phi}_t, \mathbf{q}_t, \mathbf{d}_t$);
    $\mathbf{v}_t, \mathbf{x}_t \leftarrow$ simulateTimeStep($\mathbf{x}_{t-1}, \mathbf{v}_{t-1}, \hat{\mathbf{u}}_t$);
  end for
end for
return $\{\mathbf{x}_t\}_{t\in[1,N_{frames}]}$

cost:

$$\arg\min_{\mathbf{u}_t} \|\mathbf{d}_t - \mathbf{S}\mathbf{x}_t(\mathbf{u}_t)\|^2, \tag{4.3}$$

where $\mathbf{S}$ is a vertex selection matrix matching the data markers with vertices on the facial mesh. 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 $\mathbf{u}_t$. Let's recall the equations of motion in the Projective Dynamics framework presented in section 3.2. Once the projection vector $\mathbf{p}$ linearizing the internal forces have been computed, equation 3.14 yields:

$$\mathbf{x}_t = \mathbf{A}^{-1}\left(\frac{\mathbf{M}\mathbf{y}'_t}{h^2} + \mathbf{B}\mathbf{u}_t + \mathbf{Y}\mathbf{p}\right), \tag{4.4}$$

where $\mathbf{y}'_t = \mathbf{x}_{t-1} + h\mathbf{v}_{t-1} + h^2\mathbf{M}^{-1}\mathbf{f}'_{\text{ext}}$, with $\mathbf{f}'_{\text{ext}}$ the external forces except blendforces. We rewrite equation 4.4 as

$$\mathbf{x}_t = \mathbf{\Phi}_t\mathbf{u}_t + \mathbf{q}_t, \text{with} \tag{4.5}$$

$$\mathbf{\Phi}_t = \mathbf{A}^{-1}\mathbf{B}, \tag{4.6}$$

$$\mathbf{q}_t = \mathbf{A}^{-1}\left(\frac{\mathbf{M}\mathbf{y}'_t}{h^2} + \mathbf{Y}\mathbf{p}\right). \tag{4.7}$$

This allows to compute an estimated command $\hat{\mathbf{u}}_t$ that minimizes equation (4.3) under the linear approximation allowed by the computation of the projections $\mathbf{p}$:

$$\hat{\mathbf{u}}_t = \arg\min_{\mathbf{u}_t \geq 0} \|\mathbf{S}\mathbf{\Phi}_t\mathbf{u}_t - (\mathbf{d}_t - \mathbf{S}\mathbf{q}_t)\|^2. \tag{4.8}$$

We do not allow negative actuations of the blendforces to mimic the fact that facial muscles can only pull and never push.

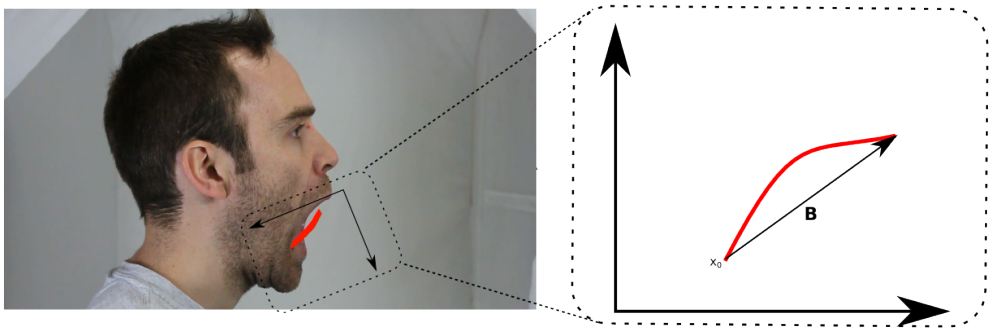


Figure 4.2 – Projection of the facial expression manifold $\mathcal{M}_{\text{true}}$ (in red) to the camera plane for a cheek point, limited to the jaw opening expression. The projection of the blendshape approximation of this manifold is depicted by the black arrow.

Notice the similarity with equation 3.4 that depicted blendshapes-based performance capture. However, in our case this equation is only an approximate solution enabled by the linearization of internal forces induced by the projection step of Projective Dynamics. We therefore iteratively recompute the projections $\mathbf{p}$ and 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.

4.3 Manifold of Facial Expressions

4.3.1 Non-Linear Blendforces Manifold

In section 4.1.2 we saw that the main factor distinguishing blendforces from the traditional blendshapes approach was the presence of internal forces. We will now investigate more closely how the effect of internal forces influence the reachable geometric configurations, and how this relates to the actual possible facial configurations.

Blendshapes form an affine approximation of the manifold of possible facial expressions, able to exactly approximate the neutral expression and a handful of basic muscle activations. Far from these particular points, the approximation gets less and less precise. With blendforces, we construct a new manifold of reachable geometries, and we will see that it is possible to ensure this new manifold is close to the true manifold of facial expressions. To illustrate this abstract concept of expression manifold, we can reduce the number of dimensions to be able to represent it on a picture. The true manifold of facial expressions is the set of values of the coordinate vector $\mathbf{x}$ that correspond to valid facial expressions:

$$\mathcal{M}_{\text{true}} = \{\mathbf{x} \mid \mathbf{x} \text{ is a valid face}\}. \tag{4.9}$$

We can picture a projection of this manifold by projecting $\mathbf{x}$ to two coordinates of a point, and analyzing only part of this projection by restricting to a particular movement. In

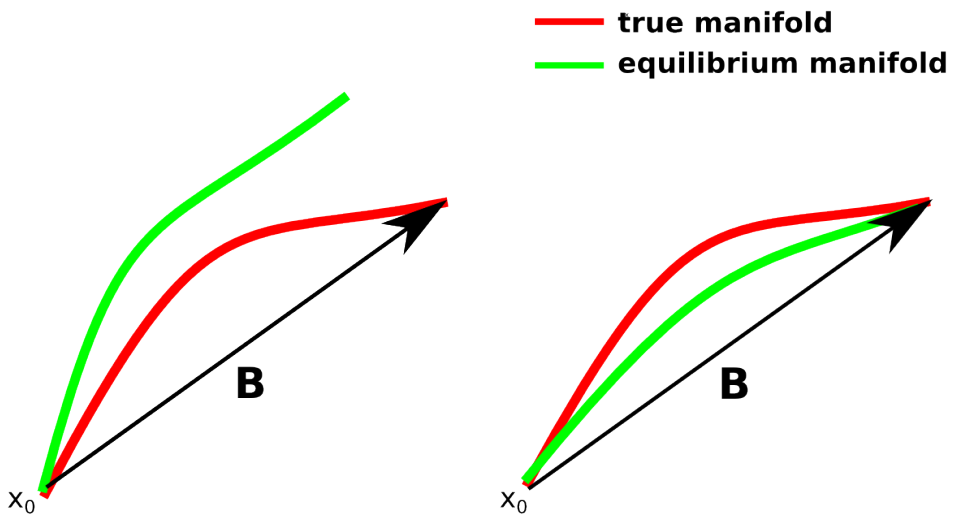


Figure 4.3 – Without further precaution, the manifold of static equilibrium $\mathcal{M}_{\text{eq}}$ is only guaranteed to contain the neutral expression $\mathbf{x}_0$ (left). Optimizing the manifold to contain blendshapes as in section 4.3.2 leads to a manifold closer to the real manifold (right).

figure 4.2 we picture the projection of $\mathcal{M}_{\text{true}}$ restricted to the movement of a facial point during jaw opening.

Due to the presence of internal forces, blendforces generate configurations close to the *static equilibrium manifold* $\mathcal{M}_{\text{eq}}$. This manifold is the set of points where static equilibrium between the blendforces $\mathbf{Bu}$ and the internal forces $\mathbf{f}_{\text{int}}$ is reached:

$$\mathcal{M}_{\text{eq}} = \{\mathbf{x} \mid \exists \mathbf{u} \text{ such that } \mathbf{Bu} + \mathbf{f}_{\text{int}}(\mathbf{x}) = 0\}. \tag{4.10}$$

The nature of internal forces is to resist deformations from a rest configuration. Since the neutral expression is defined as the expression obtained when all facial muscles are relaxed, we choose the neutral expression as the rest configuration for all internal forces. Thus, the neutral expression belongs to the static equilibrium manifold:

$$\mathbf{x}_0 \in \mathcal{M}_{\text{eq}}. \tag{4.11}$$

From equation 4.10 we deduce that $\mathcal{M}_{\text{eq}}$ is a non-linear manifold if the internal forces are non-linear, which is the case for internal forces defined in the Projective Dynamics framework. We will show in section 4.3.2 that it is possible to design internal forces to ensure that $\mathcal{M}_{\text{eq}}$ is close to particular face configurations, for instance the full blendshapes $\mathbf{x}_0 + \mathbf{b}_i$ (figure 4.3).

The manifold of static equilibrium $\mathcal{M}_{\text{eq}}$ is *not* the manifold of configurations obtained while using the blendforces framework, but these manifolds are closely related. Indeed, at any point of the physical simulation, internal forces will generate forces pointing towards

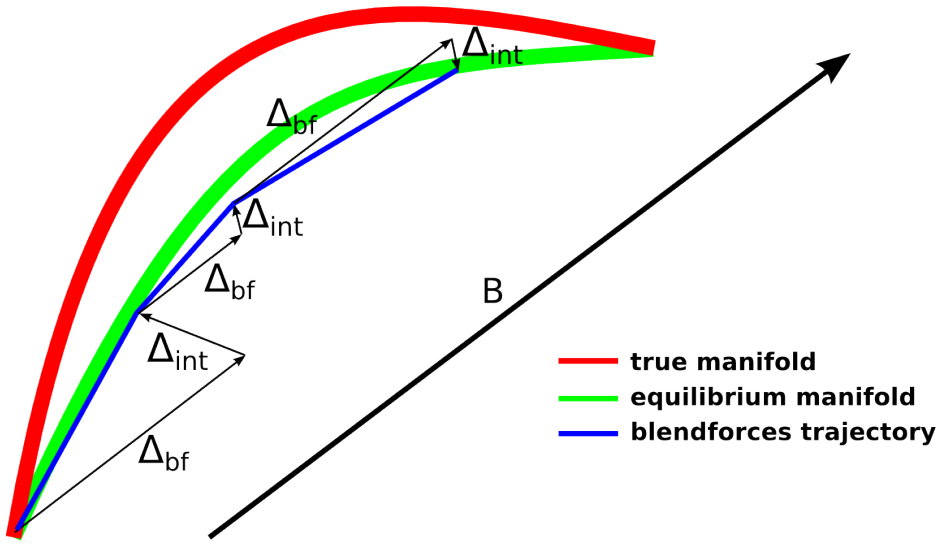


Figure 4.4 – Trajectory of the facial mesh coordinates during a blendforces animation. The trajectory is influenced by the integrated blendforces push and the integrated internal forces pull towards the equilibrium manifold. For simplicity we did not represent the contribution of the momentum on this figure.

the projection of the current configuration on $\mathcal{M}_{\text{eq}}$. Once the blendforces control procedure has converged, equation 3.11 can be reinterpreted as:

$$\mathbf{x}_t = \mathbf{x}_{t-1} + h\mathbf{v}_{t-1} + h^2\mathbf{M}^{-1}\left(\mathbf{f}'_{\text{ext}} + \mathbf{f}_{\text{int}} + \mathbf{B}\mathbf{u}_t\right). \tag{4.12}$$

This equation can be split into three parts. The configuration moves from $\mathbf{x}_{t-1}$ under the influence of its momentum $\Delta_m = h\mathbf{v}_{t-1} + h^2\mathbf{M}^{-1}\mathbf{f}'_{\text{ext}}$, the blendforces push $\Delta_{\text{bf}} = h^2\mathbf{M}^{-1}\mathbf{B}\mathbf{u}_t$, and the pull of the internal forces towards the equilibrium manifold $\Delta_{\text{int}} = h^2\mathbf{M}^{-1}\mathbf{f}_{\text{int}}$. This leads to a trajectory similar to the trajectory depicted in figure 4.4.

4.3.2 Optimizing the Equilibrium Manifold

In the previous section we mentionned that it was possible to optimize internal forces such that the full-blendshapes expression belong to the static equilibrium manifold $\mathcal{M}_{\text{eq}}$. We enforce this by computing sitffnesses weights w_k for internal forces such that the unit activation of each blendforce corresponds to static equilibrium on the corresponding blendshape.

For a delta-blendshape $\mathbf{b}_j$, the equilibrium forces can be formulated using the Projective Dynamics approximation of the internal conservative forces:

$$\min_{\mathbf{k}} \sum_j \left\| \mathbf{b}_j - \sum_k w_k \left(\mathbf{G}_k^T \mathbf{G}_k (\mathbf{x}_0 + \mathbf{b}_j) - \mathbf{G}_k^T \mathbf{H}_k \mathbf{p}_{\mathbf{k}}^{\mathbf{j}} \right) \right\|^2 \tag{4.13}$$

where $\mathbf{w}_k$ stores the stiffnesses distributions for all vertices and edges and $\mathbf{p}_k^j$ is the projection of the configuration $\mathbf{x}_0 + \mathbf{b}_j$ on the rest manifold. Thanks to the Projective Dynamics approximation the optimization term is now linear in $\mathbf{w}$, the vector containing all the w_k, and we can write it as the following linear system:

$$\min_{\mathbf{w}} \|\mathbf{\Xi}\mathbf{w} + \mathbf{f}\|^2 \tag{4.14}$$

where $\mathbf{f}$ is a vector built by stacking all columns of $\mathbf{B}$ and matrix $\mathbf{\Xi}$ 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 4.14 under the constraint $\mathbf{w} > 0$. We also penalize very small stiffness values by adding a prior term to the optimization function:

$$\min_{\mathbf{w}>0} \left(\|\mathbf{\Xi}\mathbf{w} + \mathbf{f}\|^2 + \lambda \sum_k \frac{1}{w_k^2} \right) \tag{4.15}$$

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 [42], 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.

4.4 Comparison with Recent Methods

In this chapter we introduced blendforces, a technique leveraging existing blendshapes sets to deliver facial animation in a physical simulation framework. As we have already mentionned in chapter 2, our work is not the first trying to merge data-based approaches with physical simulation. In this section we present recent alternative frameworks and highlight key differences.

4.4.1 Artist-Controlled Physical Systems

Concurrently to this work, Cong and colleagues developed techniques to automatically transfer an existing physical model to a new character [47]. They subsequently enhanced their system by enabling artistic control over the simulation results [48]. The technique they developed lets artists directly control the shape and location of facial muscles along simulation frames, while the external face vertices are driven from the muscles positions by quasi-static physical simulation. They notice that the required muscle shapes do not vary, and recommend using a blendshapes formulation to drive the muscles.

While the initial objective of Cong and colleagues is different from ours, the similarity between the resulting concept is striking. Indeed, both their work and ours rest on the assumption that driving phyiscal simulation with an underlying affine model can deliver

convincing facial animation. The difference between the two approaches lies in the applicability of the methods. While our method was designed to be able to animate any character for which blendshapes already exist, their technique requires artist intervention to build the muscle blendshapes, a new task for which artists have to be trained. Another difference between our approaches is the type of physical simulation used. Cong and colleagues favor a quasi-static framework, while we prefer using a dynamic simulation. We will later show that a dynamic framework is necessary to achieve some realistic effects.

4.4.2 Interpolation of Physical Parameters

Ma et al. [119] initiated the field of merging physical simulation and blendshapes. In their framework, the facial mesh is used to build a mass-spring system, and the blendshapes are used to drive the rest-lengths of the springs. Namely, for each spring, the rest-length is determined by

$$l = l_0 + \sum_i w_i(l_i - l_0). \tag{4.16}$$

Ma et al. demonstrated that this framework was capable of parameterizing non-linear skin movement. Concurrently to our work, Ichim and coworkers [77] extended this framework to a volumetric face physical system. In their work, the face is a tetrahedral mesh subject to volumetric elasticity and volume preservation, and forces mimicking the activations of blendshapes are represented by additional internal forces whose rest configurations are derived by interpolating between the configurations of the blendshapes.

These works share with ours the objective of producing physical simulation from a set of blendshapes. However, we argue that their representation of the forces setting the face in motion is purely synthetic. From a biomechanical point of view, muscular forces do not arise from each skin volumetric element, and thus should belong to the external forces that deform the face away from its rest configuration. From a computational point of view, a key issue with their approach is the inability to drive the muscle activations from performance capture. Indeed, their choice of muscle forces create difficult to invert forces, in contrast to our direct approach leading to the simple control equation 4.8.

Even though we disagree on the nature of forces that should be used for the simulation, we share with the work of Ichim and coworkers the goal of building an accurate face physical system. We expose our face physical system in the next chapter.

Chapter 5

Face Physical System Modelling

Contents

In the previous chapter we introduced the *blendforces* framework, wherein we present facial animation as the interaction between muscular forces lying in the span of delta-blendshapes and the response of *internal forces* that drive the face mesh towards its rest configuration. In section 4.3.1 we saw that these internal forces were crucial to the realism of our system, as their nature determines the manifold of expressions reachable under the blendforces

framework. In this chapter, we present the internal forces that we used to produce facial animation within the blendforces framework, and showcase the properties of the resulting animations.

We start this chapter by presenting how we encode the influence of the skull in internal forces (section 5.1). We subsequently show a simple surfacic model of the skin and demonstrate that it is already capable of interesting results (section 5.2). In section 5.3 we replace this simple surfacic model with a more anatomically correct volumetric model of skin. We then build upon this volumetric model to simulate interesting effects around the lips region, namely the prevention of lips self intersection and the simulation of the sticky lips effect (section 5.4).

5.1 Influence of the Skull

The skin is an elastic component completely lacking rigidity. Without the underlying skull, the facial skin would have no recognizable rest shape, and would exhibit a behavior comparable to a thick piece of clothing. The skull is thus a crucial component to model in any facial physical simulation framework. Since our objective is to build a physical system with no additional data apart from blendshapes, we cannot use existing skull-fitting methods such as Aina et al. [2]. We rely instead on the skull information already encoded in the neutral expression. Indeed, in the neutral expression, all muscles are relaxed and the skin is not put under tension. The only remaining component explaining the shape of the face is thus the presence of the skull and the jaw, and the attachment of the skin to these bones. We thus model the presence of the skull and the jaw using an internal force composed of *virtual zero-length springs* connecting each facial vertex to its location on the neutral face:

$$W_{\text{skull}}(\mathbf{x}) = \sum_j \frac{w^j_{\text{skull}}}{2} \left\| \mathbf{x}^j - \mathbf{x}^j_0, \right\|^2, \tag{5.1}$$

where $\mathbf{x}^j$ denotes the coordinates of the j'th vertex of the facial mesh. Since some vertices of the face are connected neither to the skull nor to the jaw, we leave the possibility to paint on the face mesh the location of vertices which should not be affected by this skull attachment force.

5.2 Skin Elasticity

5.2.1 Surfacic Skin Model

5.2.1.1 Spring Network

The skin is a non-homogeneous and non-isotropic viscoelastic surface, with lower stiffness along the Borges lines [179] (see figure 5.1). We can model this kind of surface using a spring energy: the variation of stiffnesses across the face can be handled by varying the stiffness of springs, and the non-istropic nature of elasticity can be encoded by the direction of the face mesh's edges. It's a standard practice for animators to build edges along Borges

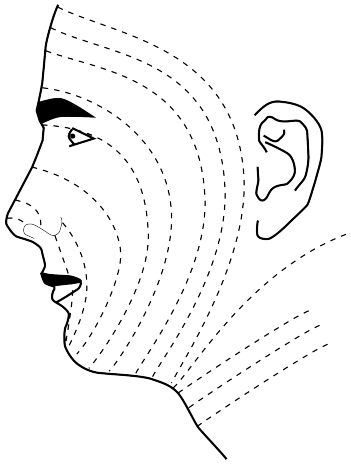


Figure 5.1 – Schematic representation of Borges lines of relaxed skin tension [22].

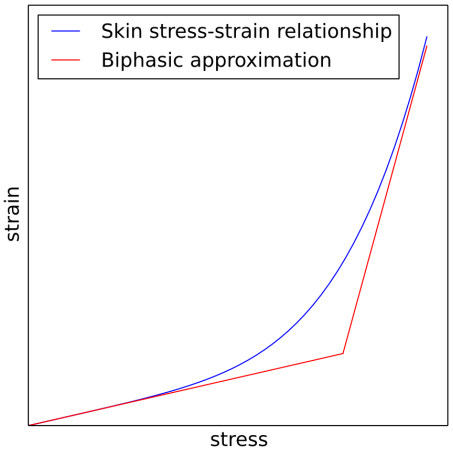


Figure 5.2 – Stress-strain relationship for the skin and its biphasic approximation. For illustrative purposes only, the curves are not accurate.

lines [137], meaning we don't have to explicitly model this non-isotropic behaviour. The Projective Dynamics potential corresponding to this elasticity is

$$W_{\text{skin}} = \sum_{i,j \in \varepsilon} \frac{w_{\text{skin}}^{ij}}{2} \left\| (\mathbf{x}^i - \mathbf{x}^j) - \mathbf{p}_e^{ij} \right\|^2, \tag{5.2}$$

where ε is the set of edges of the facial mesh, and with $\mathbf{p}_e^{ij}$ the projection term, computed by rotating the rest edge in the direction of the current edge:

$$\mathbf{p}_e^{ij} = \left\| \mathbf{x}_0^i - \mathbf{x}_0^j \right\| \frac{\mathbf{x}^i - \mathbf{x}^j}{\|\mathbf{x}^i - \mathbf{x}^j\|}. \tag{5.3}$$

Waters and Terzopolous also noted that the stiffness of the skin's elasticity had a biphasic behavior [179]. We model this by adding a second spring for each edge with a greater stiffness, but modify its projection term $\hat{\mathbf{p}}_e^{ij}$ such that it is equal to the edge if the elongation is not greater than 10%:

$$\hat{\mathbf{p}}_e^{ij} = \mathbf{x}^i - \mathbf{x}^j \text{ if } \left\| \mathbf{x}^i - \mathbf{x}^j \right\| \leq 1.1 \left\| \mathbf{x}_0^i - \mathbf{x}_0^j \right\| \tag{5.4}$$

$$= \mathbf{p}_e^{ij} \text{ otherwise.} \tag{5.5}$$

This second spring is thus inactive for short elongations, and augments the stiffness for long elongations, approximating the true stress-strain relationship of skin as shown in figure 5.2.

5.2.1.2 Curvature Preservation

Using the facial mesh's edges for the spring network cannot simulate the incompressibility of the skin. It is possible however to compensate this shortcoming by adding internal forces to preserve the skin's curvature. This is usually done by adding additional springs to connect vertices separated by two edges [119], however in the Projective Dynamics framework it is possible to directly formulate curvature preservation:

$$W_{\text{bend}}(\mathbf{x}) = \sum_j \frac{w^j_{\text{bend}}}{2} \left\| \mathbf{L}^j \mathbf{x} - \mathbf{p}^j_{\text{bend}} \right\|^2, \tag{5.6}$$

where $\mathbf{L}^j$ is the j-th row of the cotan-weighted laplacian matrix of the facial mesh, and $\mathbf{p}^j_{\text{bend}}$ is the projection term, obtained by rotating the one-ring around vertex $\mathbf{x}^j_0$ to be aligned with the one-ring of $\mathbf{x}^j$. Following Bouaziz et al. [24], this can be computed simply as

$$\mathbf{p}^j_{\text{bend}} = \left\| \mathbf{L}^j \mathbf{x}_0 \right\| \frac{\mathbf{L}^j \mathbf{x}}{\|\mathbf{L}^j \mathbf{x}\|}. \tag{5.7}$$

As an additional benefit, using a curvature preservation term preserve the curvature imposed by the skull, improving the skull handling depicted in section 5.1.

5.2.2 Experiments

In this section we evaluate the blendforces method with the skull attachment term of section 5.1 and the surfacic internal forces presented in this section. We quantitatively and qualitatively compare animations produced with our method to animations produced using blendshapes fitting. The animation are produced from performance capture data. Hence, the blendforces animation will use the control procedure described in section 4.2, and blendshapes animation will be obtained using the fitting techniques described in section 3.1.2. We will compare to two variations of blendshapes fitting. In equation 3.4 the blendshapes are fitted with a positivity constraint. This solving method will be labeled Static Blendshapes Solve (SBS) in the following. Adding an L1-norm term (equation 3.5) encourages sparsity, as combining too many shapes often produces artifacts due to conflicting deformation patterns. We will label Regularized Static Blendshapes Solve (RSBS) the solution of this minimization problem including the L1 prior. Results obtained with the blendforces method (algorithm 1) will be labeled BF.

5.2.2.1 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 5.3 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 5.4 highlights this property

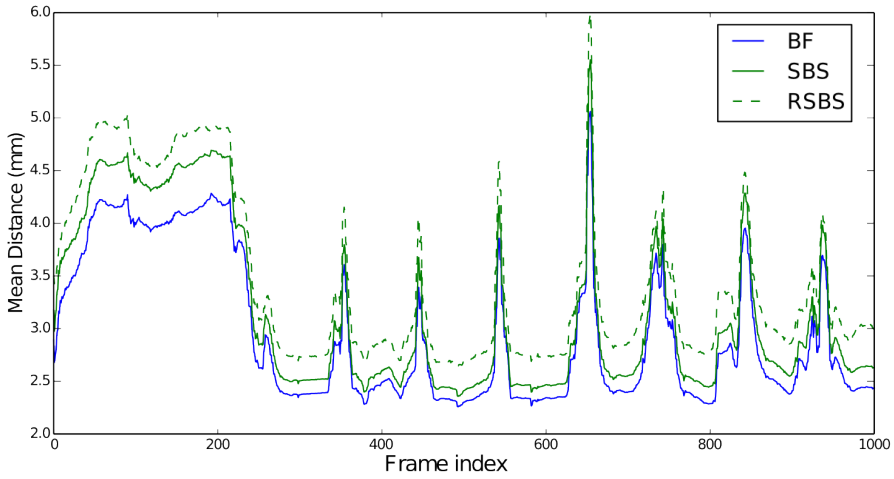


Figure 5.3 – Mean Euclidean distance between motion capture marker positions and corresponding vertices for different solvers on a sequence of speech and varied facial expressions. The same set of artist-crafted blendshapes was used for all solves. As it is limited to facial configurations within the 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.

by comparing for each frame the magnitudes of the projections of the deformation in $C(\mathbf{B})$ and its orthogonal complement $C(\mathbf{B})^{\perp}$.

5.2.2.2 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 [13, 190]. In this case we adapt an existing blendshape model to the morphology of the performer using deformation transfer [163]. 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 5.5), but this time we also evaluate the error on a *dense* set of vertices all over the face. The results presented in figure 5.7 suggest that even our relatively modest set of physical forces provides a more accurate and realistic reconstruction of skin deformation than (R)SBS.

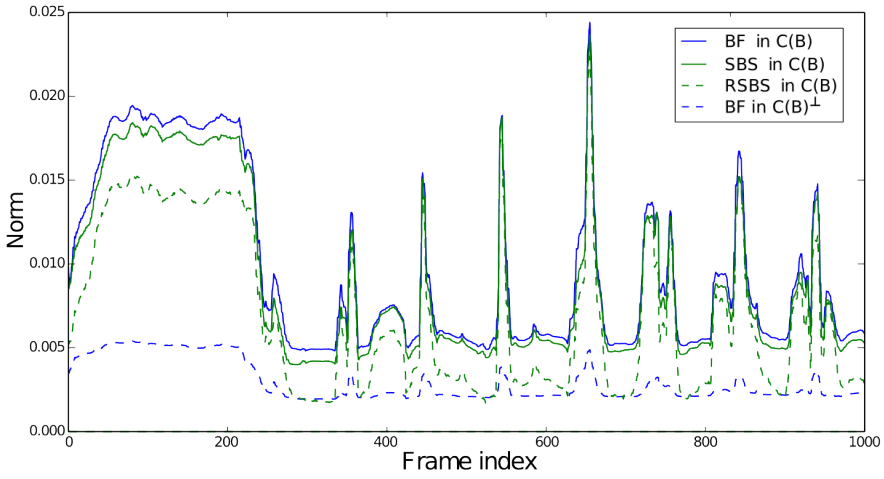
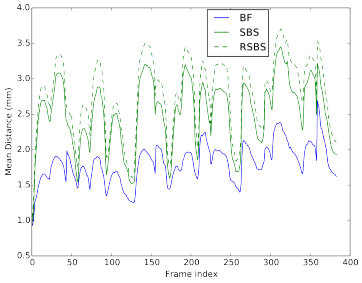


Figure 5.4 – Measure of the projections of $\mathbf{x}$ in the respective subspaces $C(\mathbf{B})$ and $C(\mathbf{B})^{\perp}$ for different solvers. All measures are normalized by the number of vertices. The blendforces meshes shows a higher contribution of components from $C(\mathbf{B})$, as well 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.

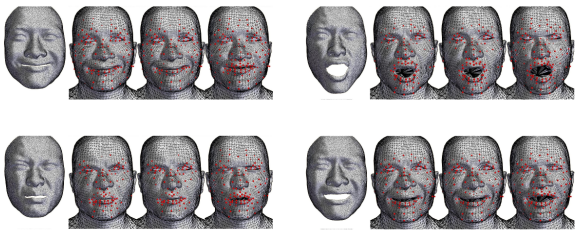
5.2.2.3 Animation Retargeting

The experiments above were focused on animating a performer-specific facial mesh in motion capture setups. It is however common to animate a character who is morphologically, and sometimes even topologically different from the source geometry. Another similar scenario consists in 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 [153].

The blendforces framework can be beneficial in such retargeting scenarios. Animation for a target character can be generated by warping a sparse set of vertices from a source animation, using the method of Seol et al. [153] 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 5.6 shows that the obtained motion for



(a) Mean Euclidean distance on the sparse set of vertices acting as marker constraints (red dots in figure 5.5b).



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

Figure 5.5 – Comparison of animation results for (R)SBS and BF solvers. Data courtesy of Zhang et al. [190]. 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 [163]. 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.

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 observed in video, showing varying behaviours upon muscle activation and relaxation [83].

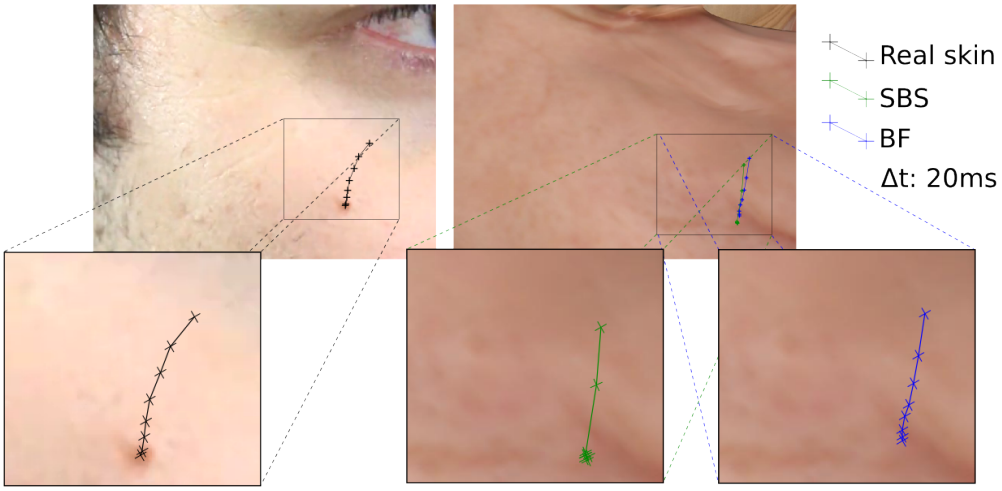
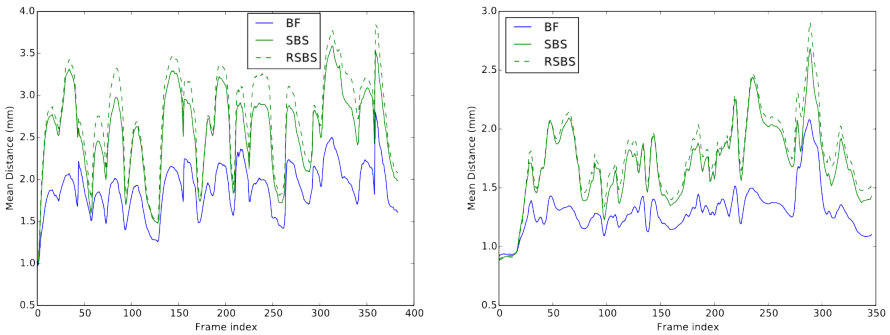
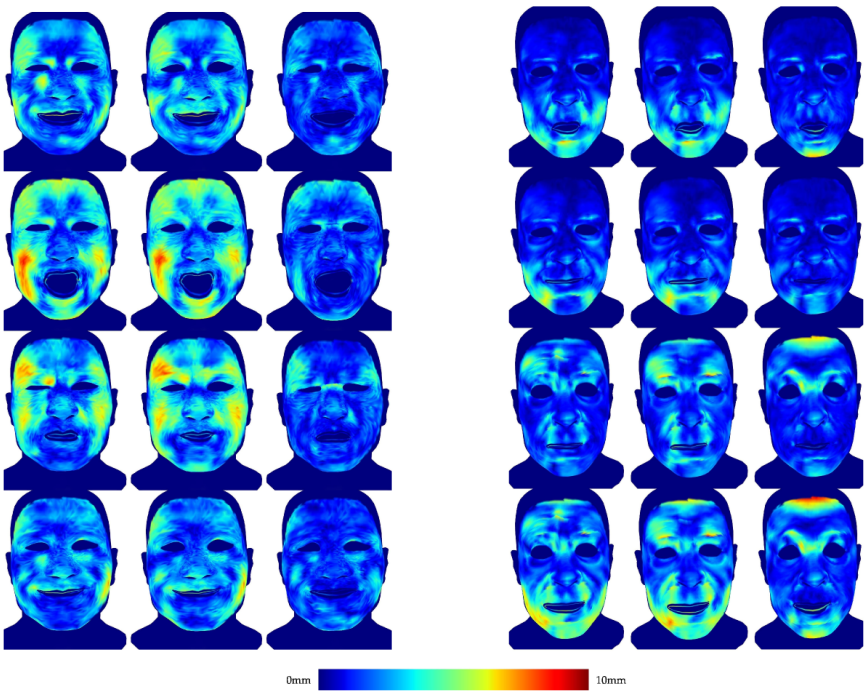


Figure 5.6 – Temporal evolution of a skin point when relaxing a muscle. While the SBS animation (green) returns to a neutral position in 40ms, BF (blue) produces a slower and smoother trajectory, comparable to trajectories observable on real-world data (left image). This effect is best seen dynamically in video.



(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 5.7 – Comparison of ground-truth error distribution on the face for static blendshape solving and blendforces. Data courtesy of Zhang et al. [190] (left) and Beeler et al. [13] (right). A dense set of motion capture points were used as ground-truth, with only a reduced sparse set acting as data constraint. Figure 5.7a displays a consistently reduced mean ground-truth error showing that blendforces reconstruct more accurate skin movements. Figure 5.7b 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 Beeler et al. sequence for SBS, RSBS, and BF are 1.71mm, 1.78mm and 1.30mm respectively. Mean distances over the whole Zhang et al. sequence for SBS, RSBS, and BF are 2.59mm, 2.76mm and 1.90mm respectively.

5.3 Volumetric Forces

While the surfacic model presented in the previous section achieves pleasing results, it is unsatisfying to model the skin using surfacic forces. One issue with the surfacic approach is that no pleasant results could be achieved without the curvature preservation internal force. However, several natural processes observed in facial expressions manifest a change of skin curvature: the formation of wrinkles requires the skin to bend, lips change their curvature while pressed against each other or when sticking to each other. For these reasons, we want to model the skin using volumetric internal forces. However, we are only provided with a blendshapes set, containing only surfacic meshes. This issue could be circumvented as in Ichim et al. [77] by fitting a template skull and layered skin model into the neutral mesh, but this would restrict the applicability of our method to human-like characters. We therefore choose to only consider a fraction of the facial volume, and build an *hypodermal surface* by extruding the neutral mesh, and connecting this surface to the neutral mesh as a tetrahedral mesh layer (section 5.3.1). We can then use this layer to integrate volumetric forces (section 5.3.2).

5.3.1 Hypodermal Surface Construction

We build an hypodermal surface by offsetting the neutral mesh's surface and smoothing its high frequency components. We assign to each vertex $\mathbf{v}^j$ a skin thickness value τ_i. These thickness values can be either equal for all vertices or painted on the mesh's surface. We compute an extruded surface $\mathbf{t}$ by subtracting the thicknesses along the normal directions:

$$\mathbf{t}^j = \mathbf{v}^j - \tau^j \mathbf{n}^j, \tag{5.8}$$

where $\mathbf{n}^j$ is the normal direction at vertex $\mathbf{v}^j$. We then compute the hypodermal surface $\mathbf{x}_h$ by minimizing the following energy:

$$\arg\min_{\mathbf{x}_h} \left(\|\mathbf{x}_h - \mathbf{t}\|^2 + w_{\text{smooth}} \|\mathbf{L}\mathbf{x}_h\|^2 + w_{\text{lap}} \|\mathbf{L}\mathbf{x}_h - \mathbf{L}\mathbf{x}_0\|^2 \right), \tag{5.9}$$

where $\mathbf{L}$ is the cotan-weighted laplacian matrix associated with the model's triangle mesh [23]. The parameters w_{smooth} and w_{lap} respectively control the smoothness of the hypodermal surface and how closely its curvature matches that of the neutral mesh.

As each vertex of the hypodermal surface has a corresponding *epidermal* vertex, it is straightforward to connect these two surfaces with a set of tetrahedrons $\mathcal{T}$. Each triangle in the epidermal surface has a correspong triangle in the hypodermal surface. The volume defined by this extrusion structure can be discretized as 3 tetrahedrons. Since there is more than one possiblity to create these tetrahedrons, we enforce a coherent structure by propagating an initial choice during a traversal of the triangle adjacency graph.

5.3.2 Volumetric Internal Forces

Based on the volumetric face mesh presented in the previous section, we now define the volumetric internal forces that will govern how the face deforms in reaction to external forces. The skin's elasticity is modelled using an as-rigid-as-possible potential:

$$W_{\text{strain}}(\mathbf{x}) = \sum_{s \in \mathcal{T}} \frac{w^s_{\text{strain}}}{2} \left\| \mathbf{D}_s \mathbf{x} - \mathbf{p}^s_{\text{strain}} \right\|^2, \tag{5.10}$$

where $\mathbf{D}_s$ is the matrix to form the deformation gradient of the tetrahedron s with respect to the rest configuration [174], and the projection term is defined by

$$\mathbf{p}^s_{\text{strain}} = \mathbf{U}_s \mathbf{V}_s^T, \tag{5.11}$$

with $\mathbf{U}_s \mathbf{\Sigma}_s \mathbf{V}_s^T = \mathbf{F}_s = \mathbf{D}_s \mathbf{x}$ the SVD of the the deformation gradient $\mathbf{F}_s$. Intuitively, this potential constrains each tetrahedron to stay close to its rest configuration, rotated to preserve the possibility of rigid motion.

Like most biological soft tissues, human skin is nearly incompressible [183]. We model this with a potential that penalizes volume changes:

$$W_{\text{vol}}(\mathbf{x}) = \sum_{s \in \mathcal{T}} \frac{w^s_{\text{vol}}}{2} \left\| \mathbf{D}_s \mathbf{x} - \mathbf{p}^s_{\text{vol}} \right\|^2, \tag{5.12}$$

with the projection term $\mathbf{p}^s_{\text{vol}}$ defined by

$$\mathbf{p}^s_{\text{strain}} = \mathbf{U}_s \mathbf{\Sigma}^*_s \mathbf{V}_s^T, \tag{5.13}$$

where $\mathbf{\Sigma}^*_s$ is the closest diagonal matrix of determinant 1 to $\mathbf{\Sigma}_s$.

Since the skull is only connected to the hypodermal surface, we modify the skull internal force of section 5.1 by connecting only the hypodermal vertices $\mathbf{x}_h$ to their rest positions using zero-rest-length springs.

5.4 Lips Interactions

The lips are the most flexible part of the human face, and can take a wide variety of shapes, and collision between lips is frequent in speech. It is however extremely challenging to accurately capture the fine movements of the lips region, since wearing markers in that region is inconvenient, and such markers would be occluded most of the time. Consequently, the movement of lips is often inferred from tracked points outside the lips region. The captured lips movement is therefore never completely faithful, and artifacts such as lips self-intersection can occur. Since we are building a physical simulation framework, we can leverage physics to prevent said self-intersections (section 5.4.1). We can then use the collision information to simulate the sticky lips phenomenon (section 5.4.2). We present simulation results focused on lips in section 5.4.3.

5.4.1 Prevention of Lips Self-Intersections

We detect possible collisions between the lips by pruning the collision search space using optimized spatial hashing [167], and detecting vertex-triangle and edge-edge collisions using the continuous collision detection method of Bridson et al. [27]. Using a continuous

collision detection method guarantees that no collision is ever missed. 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

$$\mathbf{s} = w_0\mathbf{x}^0 + w_1\mathbf{x}^1 + \alpha w_2\mathbf{x}^2 - w_3\mathbf{x}^3, \tag{5.14}$$

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:

$$\mathbf{n} \cdot \mathbf{s} > 0. \tag{5.15}$$

Following Harmon et al. [70], 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:

$$W_{\text{col}}(\mathbf{x}) = \begin{cases} \frac{k_{\text{col}}}{2}(\mathbf{c} \cdot \mathbf{x} - \tau)^2 & \text{if } \mathbf{c} \cdot \mathbf{x} < \tau \\ 0 & \text{otherwise.} \end{cases} \tag{5.16}$$

In addition, we simulate friction by damping the velocities of the colliding vertices in the plane perpendicular to $\mathbf{n}$ [129].

5.4.2 Simulation of Sticky Lips

Our physical system prevents interpenetration of lips by detecting collisions and adding non-collision constraints to the system. Additionaly, we introduce a new force to handle the sticky lips phenomenon. For each lips collision, we instantiate springs with zero-rest-length between the colliding points. These springs remain active on the following timesteps, but break when their length gets too long. More specifically, the probability for a sticky lips spring of length l to break is

$$p_{\text{break}}(l) = \frac{1 - \tanh(2 - rl)}{2} \tag{5.17}$$

where r is a constant chosen to ensure that the break probability for a caracteristic length l_0 is 0.1. We experimentally set l_0 to 2mm. At the end of each simulated frame, random numbers in the $[0, 1]$ range are generated for each sticky lips spring, and we break the springs where the generated random number is lower than $p_{\text{break}}(l)$. The break probability as a function of the sticky spring length is displayed in figure 5.8.

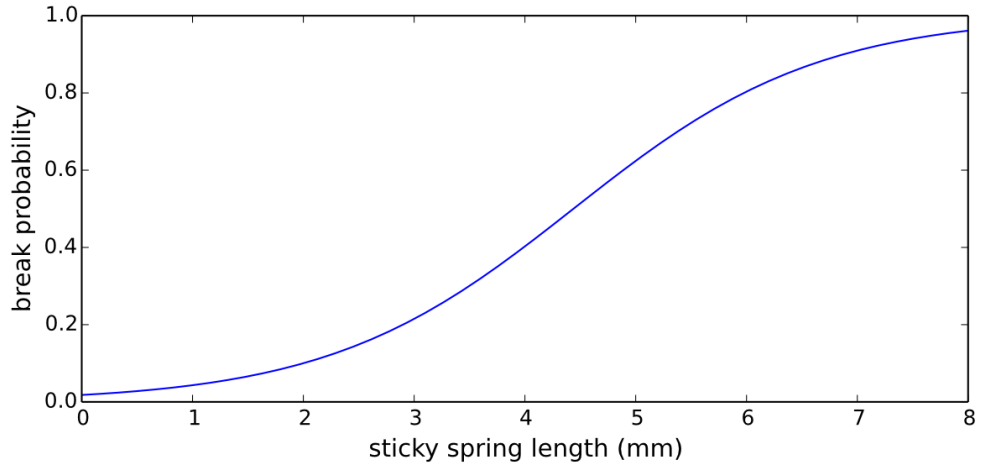


Figure 5.8 – Probability of a sticky lips spring to break as a function of its length.

The actual sticky lips phenomenon depends on lips humidity, making each sticky lips occurence different. Our random breaking scheme reproduces the non-predictible impression given by this natural process. For instance, note that the non-zero probability of breaking sticky springs of length zero accounts for the fact that sticky lips do not always occur.

5.4.3 Experiments

In this section we evaluate animations produced with the contact handling scheme of section 5.4.1 and sticky lips simulated with the method presented in section 5.4.2.

5.4.3.1 Contact Handling

Self-intersections often occur 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 section 5.4.1 prevent self-intersections, and emulate a plausible physical reaction of the mesh to lip compression. Illustrations are presented on figures 5.9, 5.10 and 5.11. It is worth noting that in the case of lip collision blendforces produces shapes outside the affine space of blendshapes (see figure 5.12). This can be interpreted as temporarily enhancing the blendshape set with new shapes specially adapted to the collision scenario.

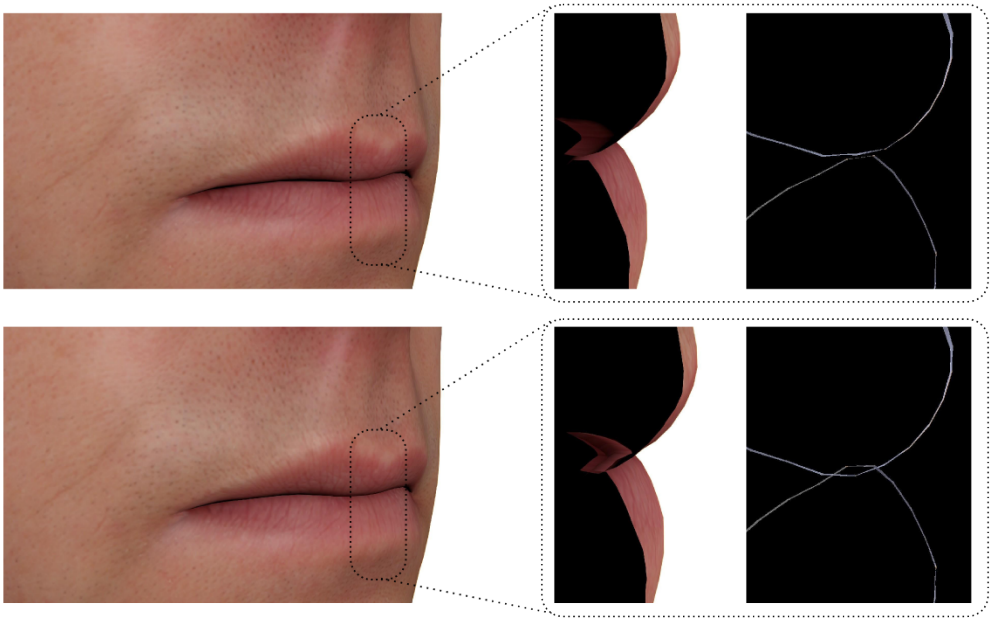


Figure 5.9 – Prevention of lips self-intersection. Top: results of our system. The lips are touching but not intersecting. Bottom: results with collision detection turned off. The lips are intersecting.

5.4.3.2 Sticky Lips

Our system produces sticky lips shapes matching those observable in the real world, as can be seen in figure 5.13. Furthermore, the timing of our sticky lips is also realistic, as demonstrated in figure 5.14. Since our system does not *capture* sticky lips but *simulates* the phenomenon, one cannot expect the system to produce the same sticky lips that would happen in the input sequence. However, we observe that the timing of the phenomenon, as well as its characteristics, closely matches natural sticky lips. Dynamic results of our simulated sticky lips can be best observed in video.

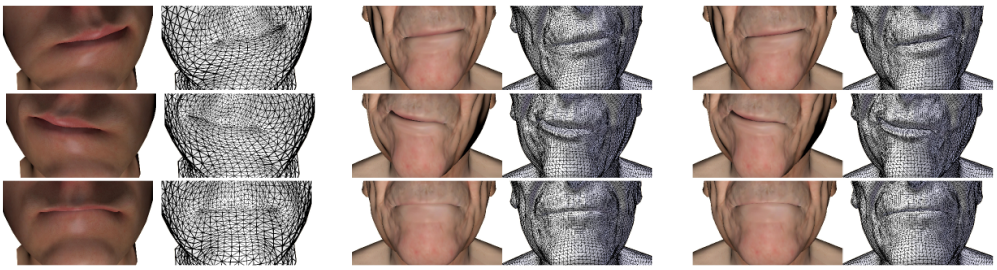


Figure 5.10 – Handling of mesh self-intersections in animation retargeted from another character. Differences in character's facial morphologies result in physical inconsistencies for the target character, such as unnatural lip intersection. Blendforces unrolls a *target-specific* physical simulation, which enables to simulate plausible physical behavior specifically for the vertices of target mesh. This effect is best viewed on video. *(left):* source animation *(center):* SBS *(right):* BF.

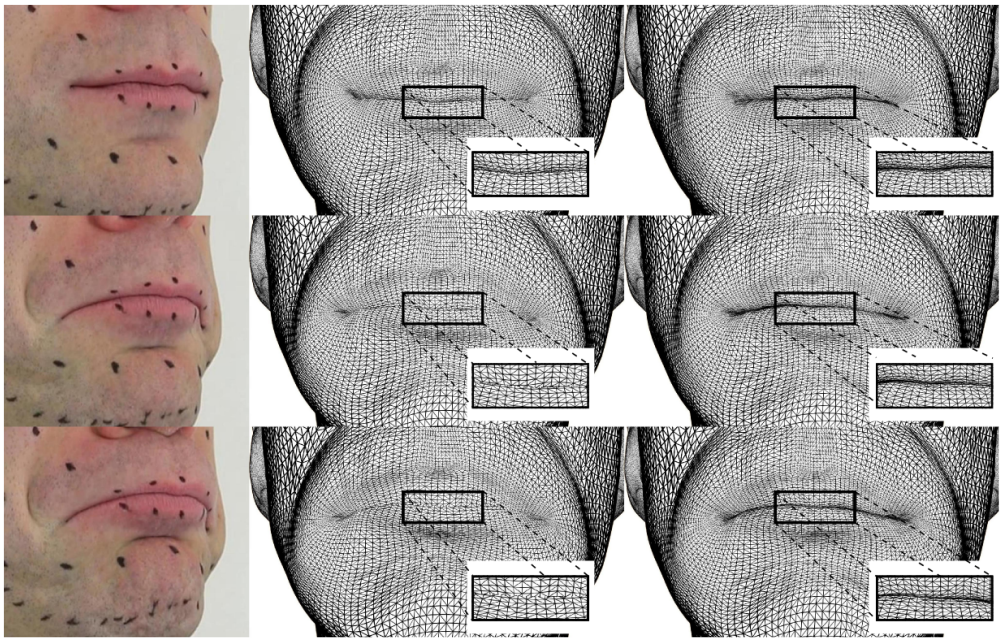


Figure 5.11 – Handling of mesh self-intersections in a performance capture scenario. The contact response is necessary for the solve to produce the correct lip shape. (R)SBS are purely data-based, and do not handle contacts and collisions, while blendforces (BF) simulates a plausible physical reaction of the mesh to lip compression. This effect is best viewed on the accompanying video. *(left):* SBS *(right):* BF.

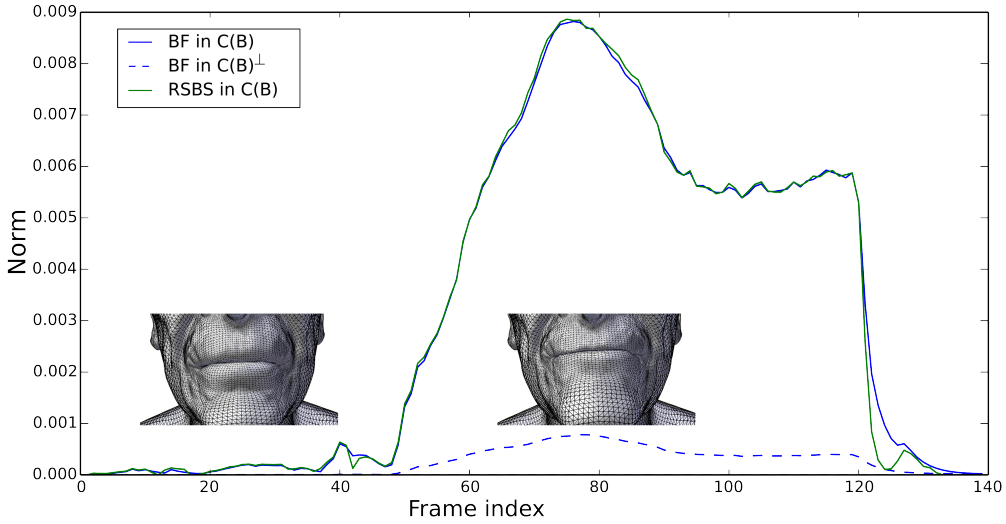


Figure 5.12 – Measure of the projections of $\mathbf{x}$ in the respective subspaces $C(\mathbf{B})$ and $C(\mathbf{B})^{\perp}$ in a collision sequence. The blendforces solver escapes the affine subspace to create collision free shapes.

Figure 5.13 – Comparison of the shapes of natural sticky lips phenomenon and our simulated sticky lips. The latter generates plausible sticky lips shapes, matching what can be observed in nature.

Figure 5.14 – Comparison of the timing of natural sticky lips phenomenon and our simulated sticky lips. Left: a natural sticky lips sequence, with frames separated by 120ms. Right: a simulated sticky lips sequence, at the same rate.

5.5 Perspectives

In this chapter we have presented techniques to generate a plausible face physical system using only a set of blendshapes as input. We have shown that the resulting physical system could be used to fit more precisely motion capture data, to produce more accurate skin dynamics, to handle lips contact and to generate sticky lips. Our contact system can potentially be extended to work in the eye region, which would enable simulating the contact between the eyelids and the eyes. Such a contact could then be used to obtain a non-linear trajectory for the eyelids during eye closing. Since the eye is not purely circular, simulating this contact could also induce subtle eyelid movements following the gaze direction, enabling more realistic facial animations.

An interesting area of improvement would be the simulation of plausible wrinkles. Our current system does not produce them, but we believe it could be extended to incorporate them. We believe wrinkles could be simulated by modifying the stiffness distribution of internal forces. Some wrinkle lines would be drawn on the face, and the stiffness of the skull attachment force (section 5.1) on vertices close to this line would be strengthened. We conjecture that this procedure could produce plausible frontal wrinkles.

A great use case for performance-driven facial animation is realtime applications. In the next chapter, we show that our system is amenable to realtime use.

Chapter 6

Realtime Physics-Based Facial Animation

Contents

In the previous chapters we have presented the *blendforces* framework, a method that produces performance-driven facial animation with physical simulation using only a set of blendshapes as its input. Performance-driven facial animation has nice applications in contexts requiring realtime performance: live conferencing with virtual avatar, theme parks, live theater, previsualization of performance capture results. Therefore, to widen the applicability of our method, we need to ensure that realtime performance can be achieved. Another factor that could hinder the applicability of our method is the requirement to use marker-based performance capture.

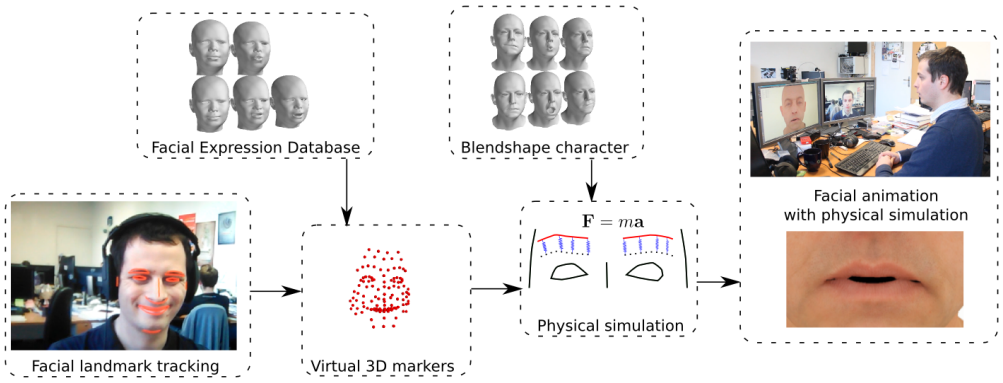


Figure 6.1 – Overview of our method to create realtime facial animation with physical effects from an RBG camera.

In this chapter, we address both shortcomings. We design a realtime video based markerless performance capture control method for blendforces that also enables the capture of inertial effects generated by head motion (section 6.1). We also contribute improvements to the Projective Dynamics framework to enable realtime physical simulation in the challenging context of facial animation. Indeed, changing internal forces between timesteps, as is required by the simulation of lips contacts and sticky lips, prevents from using a pre-factorized Cholesky solver for the global step of Projective Dynamics (see section 3.2). In section 6.2, we present a new Projective Dynamics solver that works in realtime even in the context of changing internal forces. Another performance obstacle is the high computational cost required to compute volumetric forces. We contribute an SVD approximation scheme in section 6.3 that greatly reduces the computational cost of volumetric forces. In section 6.4, we evaluate the resulting realtime performance driven physically-based facial animation system. To the best of our knowledge, our method is the first realtime physical simulation method applied to facial animation. It should also be noted that contrary to recent realtime physical simulation methods, our method does not require the computational power of a GPU to run, and works on a standard CPU.

6.1 Performance Driven Realtime Control

In this section we present our method to produce physically enhanced facial animations in realtime driven by a performance captured by a simple RGB camera. An overview of the method is presented in figure 6.1. In section 6.1.1 we present the enhancements to the control system presented in section 4.2 that enable this application. We investigate the new physical effects enabled by this capture in section 6.1.2.

6.1.1 RGB Camera Control

We drive our physical simulation by computing the blendforces actuation parameter $\mathbf{u}$ that reproduces as faithfully as possible facial performances captured by an RGB camera. The blendforces framework enables computing these actuations from motion capture markers, however no markers are available in our setup. We overcome this issue by generating "virtual markers" matching the captured facial performance.

From the video frames we extract facial landmarks $\mathbf{l}$ using a state-of-the-art method [82]. Using the 3D blendshape database FaceWarehouse [34], we build a parameteric 3D facial shape model capable of modeling variations of identity and expression as in Thies et al [168]:

$$\mathbf{s}(\beta_{\text{id}}, \beta_{\text{exp}}) = \mu + \mathbf{\Psi}_{\text{id}}\beta_{\text{id}} + \mathbf{\Psi}_{\text{exp}}\beta_{\text{exp}}. \tag{6.1}$$

We fit this model to the 2D landmarks by minimizing the energy:

$$E_{\text{fit}} = E_{\text{proj}} + w_{\text{id}}E_{\text{id}} + w_{\text{exp}}E_{\text{exp}}, \tag{6.2}$$

where E_{id} and E_{exp} are PCA priors preventing the identity and expression parameters from leaving their validity range, and

$$E_{\text{proj}} = \sum_{\mathbf{v}\in S(\mathbf{s})} \left\| \Pi\left(\mathbf{Q}\mathbf{v} + \mathbf{t}\right) - \mathbf{l}_{\mathbf{v}} \right\|_F^2, \tag{6.3}$$

where Π is the camera projection operator, $S(\mathbf{s})$ is the set of vertices in our parametric face model that have a corresponding landmark point, $\mathbf{l}_{\mathbf{v}}$ is the landmark corresponding to vertex $\mathbf{v}$, and $\mathbf{Q}$ and $\mathbf{t}$ represent the rigid motion of the face model.

We then apply the computed expression parameters β^*_{exp} to identity parameters β^+_{id} corresponding to the neutral mesh of our face physical system, and select some "virtual markers" $\mathbf{t}$ on the resulting mesh:

$$\mathbf{t} = \mathbf{T}\left(\mu + \mathbf{\Psi}_{\text{id}}\beta^+_{\text{id}} + \mathbf{\Psi}_{\text{exp}}\beta^*_{\text{exp}}\right), \tag{6.4}$$

where $\mathbf{T}$ is a selection matrix for our virtual markers. These markers are then used to compute the blendforces actuation parameters $\mathbf{u}^*$ that bring the facial system closest to the markers, using equation 4.8. In this setup, the control problem can be solved using a linear least-squares solve, which features a constant left-hand side when only considering internal forces with constant support. For this reason, we solve for $\mathbf{u}^*$ without taking transient forces such as lips collisions or sticky lips into account. We justify this approximation by noting that a realistic facial mesh configuration featuring lips self-intersection is typically really close in euclidean distance to a configuration with no self-intersection. Therefore, blendforces actuations computed without considering collision forces should be very close to those computed while taking these forces into account. A similar argument can be used for sticky lips forces. The computed actuation parameters $\mathbf{u}^*$ depend upon the linearization of the internal forces enabled by the Projective Dynamics projection step. To get the most accurate result, it would be required to iteratively recompute $\mathbf{u}^*$ while recomputing the projections, as in algorithm 1. However, we found that for realtime use, using the projections from the previous timestep and solving equation 4.8 only once per timestep was a good compromise between accuracy and computational performance.

6.1.2 Inertial Forces

Since we now use a static RGB camera to drive our facial animation, we have access to the rigid pose of the head, namely the rotation matrix $\mathbf{Q}$ and the translation vector $\mathbf{t}$ of equation 6.3. We can use this pose information and its variation over time to compute a correctly oriented gravity force vector and inertial forces acting on the face physical system. We choose to perform our physical simulation in the skull's referential frame. This requires us to compute a rectified gravity force. Let $\mathbf{g}$ be the acceleration of gravity in the camera's referential frame. Then the acceleration of gravity in the skull's frame is

$$\mathbf{g}_{\text{skull}} = \mathbf{Q}^T \mathbf{g}. \tag{6.5}$$

Since we perform our simulation in a non-inertial reference frame, we need to explicitly account for inertial forces by adding fictious external forces to our simulation. These inertial forces derive from the acceleration of the skull computed in the camera's referential frame, and can be formulated as:

$$\mathbf{f}_{\text{inertial}} = -\mathbf{M}\left(\ddot{\mathbf{Q}}\mathbf{x} + \ddot{\mathbf{t}} + 2\dot{\mathbf{Q}}\dot{\mathbf{x}}\right). \tag{6.6}$$

While inertial forces and gravity usually have no observable effects on most human faces, they can produce important effect for fleshy faces or for non-human characters. In section 6.4 we will demonstrate the interest of taking these forces into account when simulating a non-human character which has tentacles on its face.

6.2 Progressive Projective Dynamics Solver

6.2.1 Shortcomings of Projective Dynamics Solvers

The computational performance of Projective Dynamics comes from having a constant system matrix $\mathbf{A}$ in equation 3.15. This enables the use of efficient pre-factorized solvers, typically a Cholesky solver. However, in the face of changing constraints, the global step matrix is no longer constant, and alternative methods have to be explored. Wang [174] solves the global step system using one Jacobi iteration, and relies on Chebyshev multipliers to accelerate convergence, therefore requiring no pre-factorization of the system. However, Jacobi iterations have no convergence guarantees for matrices that are not diagonally dominant, and the system matrix for tetrahedral strain and volume constraints contains strong off-diagonal components. Wang proposes to use under relaxation to overcome this issue, but in our case an under relaxation factor of 0.3 was necessary to avoid divergence, making the convergence too slow to be usable in practice. Besides, Wang recommends turning Chebyshev multipliers off for the first 10 iterations, requiring even more iterations to provide a benefit over Jacobi iterations, which in practice is only affordable on the GPU. Ichim et al. [77] propose to handle collisions by augmenting the global step system with equality constraints using Lagrange multipliers, however this method cannot handle other kinds of changing constraints. Most notably, this method is not suited to express the additional spring constraints we introduced to simulate sticky lips.

An effective method to boost the convergence properties of Projective Dynamics has been devised by Liu et al. [112]. They identify Projective Dynamics as a quasi-Newton method, showing that equation 3.14 is equivalent to updating $\mathbf{x}_t$ by $-\mathbf{A}^{-1}\nabla E_{\text{PD}}$. They thus see $\mathbf{A}^{-1}$ as an initial inverse Hessian approximation, and propose to improve upon this approximation by integrating the limited-memory BFGS (L-BFGS) procedure into the Projective Dynamics procedure. They show that integrating L-BFGS enables more diverse simulation capabilities and accelerated convergence. Their method shows excellent results, but is not directly suited for the simulation of systems with constraint changes, as it still relies on a pre-factorization of the global step matrix $\mathbf{A}$ for optimal performance. We propose to extend their method to handle systems with constraint changes. To this end, we rely on the Gauss-Seidel iterative process as a replacement of the pre-factorized Cholesky solver. We thereby take advantage of the fact that Projective Dynamics guarantees that the global step matrix is SPD and Gauss-Seidel iterations are guaranteed to converge on SPD matrices.

6.2.2 Adaptive Gauss-Seidel Iterations

Since our face physical system features changing constraints, we cannot directly employ the L-BFGS accelerated Projective Dynamics method of Liu and colleagues [112]. We propose a variation of their method where we integrate iterations of the Gauss-Seidel solver within the L-BFGS procedure itself to play the role of $\mathbf{A}^{-1}$. We call this optimization scheme Gauss-Seidel-L-BFGS (algorithm 2).

Integrating Gauss-Seidel into L-BFGS The L-BFGS procedure requires an initial Hessian approximation defined only by its ability to be multiplied by a vector. The sparse Cholesky solver used by Liu and colleagues satisfies this condition, and so does an iterative solver such as the Gauss-Seidel solver we propose. However, integrating an iterative solver presents additional difficulties. With an iterative solver used for a *limited* number of iterations, there is no guarantee that the method will effectively produce a descent direction.

To see how we can make sure we get a descent direction, let's briefly recall the functionning of L-BFGS. L-BFGS first transforms the gradient ∇E_{PD} using curvature information accumulated in previous iterations (procedure `forwardLbfgs()` in algorithm 2). Then the initial inverse Hessian is applied, and another transformation by the curvature information (procedure `backwardLbfgs()`) yields the update. The update is guaranteed to be a descent direction if the initial inverse Hessian is SPD. Since the Gauss-Seidel iterative method is guaranteed to converge on SPD matrices, given any inital value, there exists a Gauss-Seidel iteration count such that the update obtained is a descent direction. We take advantage of the cheap computational cost of the `backwardLbfgs()` procedure to iteratively check if, after an initial number of iterations, the computed update is a descent direction, augmenting the number of Gauss-Seidel iterations otherwise.

To ensure that this procedure does not result in too many additional Gauss-Seidel iterations, we introduce an heuristic for the choice of the starting point of the iterative method. We found experimentally that using the result of the `forwardLbfgs()` procedure as a

Algorithm 2: L-BFGS Gauss-Seidel procedure

Function `gsLbfgs` $(\mathbf{A}, n)(\mathbf{g}, k, m)$
  Data:
  $\mathbf{A}$: global step system matrix
  n: number of Gauss-Seidel iterations
  $\mathbf{g}$: gradient of E_{PD}
  k: projective dynamics iteration count
  m: history size
  Result: $\mathbf{d}$, the descent direction
  $\mathbf{q} \leftarrow$ `forwardLbfgs`$(\mathbf{g}, k, m)$
  `/* n-1 GS iterations, starting from q */`
  $\mathbf{s} \leftarrow$ `Gauss-Seidel`$(\mathbf{A}, n-1, \mathbf{q}, \mathbf{q})$
  `/* Ensure we computed a descent direction */`
  repeat
    $\mathbf{s} \leftarrow$ `Gauss-Seidel`$(\mathbf{A}, 1, \mathbf{s}, \mathbf{q})$
    $\mathbf{d} \leftarrow$ `backwardLbfgs`$(\mathbf{s}, k, m)$
  until $\mathbf{g}^T\mathbf{d} < 0$
  return $\mathbf{d}$
end

starting point leads to a descent direction without additional iterations most of the time.

Number of Gauss-Seidel iterations We place our Gauss-Seidel-L-BFGS procedure as the global step optimization procedure within Projective Dynamics (algorithm 3). Besides avoiding the need for re-factoring matrix $\mathbf{A}$ in the face of changing constraints, using the Gauss-Seidel iterative method allows skipping computations that yield unnecessary precision. Indeed, solving $\mathbf{A}^{-1}\mathbf{q}$ to full Gauss-Seidel convergence wastes computations, since computing new constraint projections in each local step implies solving on a *different* right hand side $\mathbf{q}$ on the following global step. A more efficient scheme is to compute an approximate solution, while ensuring that this approximation is accurate enough to represent an improvement for the overall simulation. We contribute an heuristic to determine an appropriate number of Gauss-Seidel iterations, balancing computational resource usage and accuracy of the solve. Our heuristic rests on the Armijo condition of sufficient decrease. This condition requires that, for a descent direction $\mathbf{d}$:

$$E_{\text{PD}}(\mathbf{x} + \alpha\mathbf{d}) \leq E_{\text{PD}}(\mathbf{x}) + c\alpha\nabla E_{\text{PD}}(\mathbf{x})^T\mathbf{d}, \tag{6.7}$$

where α is the line search step length, and $c \in [0, 1]$ a constant (we use $c = 0.1$). Our L-BFGS scheme includes a backtracking line search step that checks if the decrease of E_{PD} verifies the Armijo condition. We interpret a number of line search iterations greater than one as a hint that the number of Gauss-Seidel iterations was not sufficient. Consequently, we increase the number of iterations for the next global steps in the current timestep. The complete procedure is outlined in algorithms 2 and 3. The procedure `updateLbfgsHistory()`

Algorithm 3: Simulation with adaptive Gauss-Seidel solver

for $t \in [1, N_{timesteps}]$ **do**
  $\mathbf{y}_t \leftarrow \mathbf{x}_t + h\mathbf{v}_t + h^2\mathbf{M}^{-1}f_{\text{ext}}$
  $\mathbf{x}_{t+1,1} \leftarrow \mathbf{y}_t$
  $n \leftarrow 2$
  for $k \in [1, N_{iterations}]$ **do**
    $\mathbf{p} \leftarrow \text{projectOnConstraints}(\mathbf{x}_{t+1})$
    $e_0 \leftarrow E_{\text{PD}}(\mathbf{x}_{t+1,k}, \mathbf{p})$
    $\mathbf{g} \leftarrow \nabla E_{\text{PD}}(\mathbf{x}_{t+1,k}, \mathbf{p})$
    $\mathbf{d} \leftarrow \texttt{gsLbfgs}(\mathbf{A}, n)(\mathbf{g}, k, m)$ /* See alg. 2 */
    $n_{\text{ls}} \leftarrow 0$
    $\alpha \leftarrow 1$
    repeat
      $\mathbf{x}_{t+1,k+1} = \mathbf{x}_{t+1,k} + \alpha\mathbf{d}$
      $e \leftarrow E_{\text{PD}}(\mathbf{x}_{t+1,k+1}, \mathbf{p})$
      $\alpha \leftarrow \alpha/2$
      $n_{\text{ls}} \leftarrow n_{\text{ls}} + 1$
    until $e \leq e_0 + c\alpha\mathbf{g}^T\mathbf{d}$ /* Armijo condition */
    if $n_{ls} \neq 1$ **then**
      $n \leftarrow 2n$
    end if
    $\texttt{updateLbfgsHistory}(\mathbf{x}_{t+1,k+1}, \mathbf{g})$
  end for
  $\mathbf{x}_{t+1} \leftarrow \mathbf{x}_{t+1,k+1}$
end for

records the curvature information for the latest m iterations. As Liu and colleagues, we choose $m = 5$.

While a full Cholesky factorization is not an option for realtime scenarios, in offline cases, we might find that increasing the number of Gauss-Seidel iterations could lead to a solve slower than a Cholesky solve, albeit less precise. While this issue does not arise in practice for realtime simulation, handling this case is important to use our solver for offline simulations. In offline scenarios, we propose a simple strategy, consisting of computing the Cholesky factorization of the new system matrix in a background thread, and using the Cholesky solve instead of the Gauss-Seidel solve once the factorization is complete. This way, the factorization time has little influence on the time to compute a timestep.

6.2.3 Convergence Properties

We assert the effectiveness of our solver by measuring its convergence properties on a typical simulation timestep involving collisions and sticky lips. We measure the relative error to

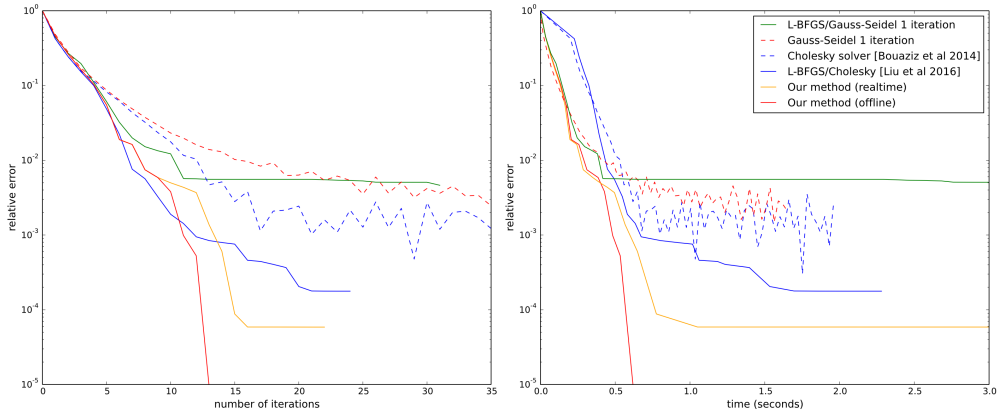


Figure 6.2 – Decrease of the relative error on a typical simulation timestep involving collisions and sticky lips. The relative error is measured with respect to the best converged state among all methods. Timings were measured without using our SVD memoization strategy (section 6.3).

the converged state $\mathbf{x}^*$ of the method that achieved the lowest energy. The decrease of the relative error $\epsilon = (E_{\text{PD}}(\mathbf{x}) - E_{\text{PD}}(\mathbf{x}^*))/(E_{\text{PD}}(\mathbf{x}_0) - E_{\text{PD}}(\mathbf{x}^*))$ is shown in Figure 6.2. We compare against Projective Dynamics with a Cholesky solver [24] and Projective Dynamics with a solver made of 1 iteration of Gauss-Seidel. Due to the non-smooth derivative of the collision term, these methods without a line search present oscillations once the system gets close to its converged state. We showcase the convergence behavior of two variants of our method. In the realtime variant, the varying number of Gauss-Seidel iterations is used, while in the offline variant, as described previously, we switch to a Cholesky solver once the offloaded factorization has been computed, at iteration 8 in this experiment. To assess the importance of our Gauss-Seidel iteration strategy, we compare our method to a version with a single Gauss-Seidel iteration per solve. After a few global iterations, a single Gauss-Seidel iteration does not provide a good descent direction, causing more line search steps to be performed and a stall in relative error reduction.

We also compare to the method of Liu and colleagues [112]. Their method and ours share similar convergence properties, but our method shows better computational performance since it doesn't pay the price of recomputing a Cholesky factorization. One can notice that our offline method achieved the lowest energy here, but that is not always the case, and on other frames the method of Liu and colleagues may get a lower energy. Still, the difference in relative error is always on the order of 10^{-4}, and the convergence curves still exhibit similar shapes in the first iterations. We interpret this phenomenon as an artifact of the line search procedure. Indeed, as Liu and coworkers, we chose a computationally efficient backtracking line search procedure based on Armijo conditions, however the recommended line search strategy for L-BFGS line search should also check the Wolfe conditions, which prevent too short updates, but comes at higher computational cost. While Liu et al. did not find issues with this approach, we think our setup, and more particularly collision forces,

can lead to very small updates, which cause the procedure to stall sooner than it would with a better line search scheme. For our realtime approach, we still advocate for the simple backtracking line search approach, but for offline simulation integrating Wolfe conditions could be interesting.

These results show that the convergence properties of our method are particularly interesting for realtime use, since it consistently achieves the best relative error reduction in the early parts of the simulation.

6.3 Fast SVD Approximation

Simulating volumetric skin in the Projective Dynamics framework requires computing expensive per-tetrahedron SVDs. Indeed, volume preservation is an important component for the realistic simulation of skin, and cannot be handled without computing the singular values, as opposed to volumetric strain which can be computed with a lighter polar decomposition [174]. Wang noted that, even on the GPU with a fast global step solver, SVD computation would dominate the simulation time. This problem is even more relevant on the CPU, with less computational power than the GPU for the highly parallel local steps. To tackle this issue, we note that, since Projective Dynamics iteratively computes the vertices positions, the deformation gradients of tetrahedrons are unlikely to have changed much between two iterations. It is therefore possible to take advantage of SVDs evaluated on previous iterations to reduce the computational burden. We propose to leverage previous computations with a differential scheme using Jacobians of the SVD to form its first-order Taylor approximation.

6.3.1 SVD Taylor Expansion

From the SVD decomposition $\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}$ of a matrix $\mathbf{F}$ given by its indices $(F_{ij})_{i,j}$, it is possible to compute the Jacobians of the decomposition $\frac{\partial \mathbf{U}}{\partial F_{ij}}, \frac{\partial \mathbf{\Sigma}}{\partial F_{ij}}, \frac{\partial \mathbf{V}}{\partial F_{ij}}$ [138]. However, computing these Jacobians is an $O(n^4)$ operation, even more than the already too expensive $O(n^3)$ of the SVD. While this could suggest that using these Jacobians for a first-order Taylor approximation is too expensive for real-time needs, we show that it is possible to formulate a really cheap procedure to compute them in our setup. In the following, please note that we consider an unusual version of the SVD, where the $\mathbf{U}$ and $\mathbf{V}$ matrices are actual rotations, meaning that the diagonal of $\mathbf{\Sigma}$ can have negative elements.

Consider the deformation gradient of a tetrahedron $\mathbf{F} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$. Instead of forming the Jacobians of this SVD, we compute the Jacobians of matrix $\mathbf{\Sigma} = \mathbf{U}^T\mathbf{F}\mathbf{V}$. In the following Projective Dynamics iteration, given a new deformation gradient $\mathbf{F}'$, we use these Jacobians in a first-order Taylor expansion to approximate all three SVD matrices of $\mathbf{U}^T\mathbf{F}'\mathbf{V}$. We can then recover an aproximate SVD for F' as:

$$\begin{aligned} \text{SVD}\left(\mathbf{U}^T\mathbf{F}'\mathbf{V}\right) &\simeq \left(\hat{\mathbf{U}}', \hat{\mathbf{\Sigma}}', \hat{\mathbf{V}}'\right) \\ \text{SVD}\left(\mathbf{F}'\right) &\simeq \left(\mathbf{U}\hat{\mathbf{U}}', \hat{\mathbf{\Sigma}}', \mathbf{V}\hat{\mathbf{V}}'\right). \end{aligned} \tag{6.8}$$

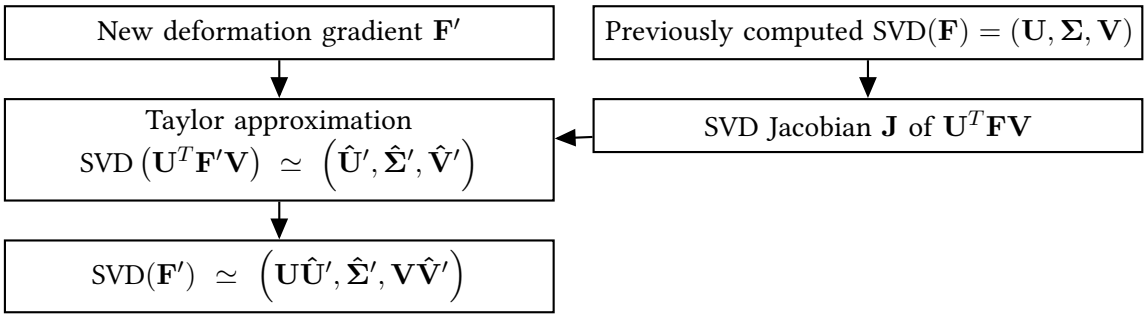


Figure 6.3 – Differential SVD approximation process

The process for computing an approximate SVD this way is outlined in figure 6.3. With this setup, we have reduced our problem to computing the SVD Jacobians on a diagonal matrix, $\mathbf{U}^T\mathbf{F}\mathbf{V}$. It turns out that this problem is significantly less expensive, as many steps of the process can be simplified. In fact, the SVD Jacobians turn out to have a close form solution and to exhibit strong sparsity. We only present the resulting Jacobians here, but interested readers are invited to read the complete derivation in appendix A. For a 3×3 diagonal matrix $\mathbf{C}$, let $\tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^T$ be its SVD. The Jacobian for the $\tilde{\mathbf{\Sigma}}$ SVD matrix coefficients is given by

$$\frac{\partial \tilde{\mathbf{\Sigma}}}{\partial \mathbf{C}} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}, \tag{6.9}$$

where we vectorize matrix $\mathbf{C}$ in row-order fashion. Recalling equation 6.8, this means that we approximate the $\hat{\mathbf{\Sigma}}'$ component of the SVD of $\mathbf{F}'$ by the diagonal of $\mathbf{U}^T\mathbf{F}'\mathbf{V}$. The Jacobians of $\tilde{\mathbf{U}}$ and $\tilde{\mathbf{V}}$ have a similar sparsity pattern and similar nonzero values, given by

$$\frac{\partial \tilde{\mathbf{U}}}{\partial \mathbf{C}} \equiv \frac{\partial \tilde{\mathbf{V}}}{\partial \mathbf{C}} \equiv \begin{pmatrix} \bullet & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \bullet & 0 & \bullet & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \bullet & 0 & 0 & 0 & \bullet & 0 & 0 \\ 0 & \bullet & 0 & \bullet & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \bullet & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \bullet & 0 & \bullet & 0 \\ 0 & 0 & \bullet & 0 & 0 & 0 & \bullet & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \bullet & 0 & \bullet & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \bullet \end{pmatrix}, \tag{6.10}$$

where the bullets $\bullet$ represent nonzero values, which have the forms

$$\bullet \equiv \frac{1}{2}\left(\frac{\pm 1}{\tilde{d}_k + \tilde{d}_l} + \frac{1}{\tilde{d}_k - \tilde{d}_l}\right) \text{ or } \bullet \equiv \frac{\pm 1}{\tilde{d}_k - \tilde{d}_l}, \tag{6.11}$$

with $\tilde{d}_0, \tilde{d}_1, \tilde{d}_2$ the diagonal values of $\tilde{\mathbf{\Sigma}}$.

The SVD Jacobians in our case thus only require to compute about 15 values, much less than the 108 values required by the method described by Papadopoulo and Lourakis [138].

Algorithm 4: SVD Jacobian approximation

Function $\texttt{svdApprox}\ (\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}, \mathbf{J}, \mathbf{F}')$

Data:
$(\mathbf{U}, \mathbf{\Sigma}, \mathbf{V})$: SVD of previous deformation gradient $\mathbf{F}$
$\mathbf{J}$: components of the Euler-SVD Jacobian
$\mathbf{F}'$: new deformation gradient
Result: $(\tilde{\mathbf{U}}, \tilde{\mathbf{\Sigma}}, \tilde{\mathbf{V}})$, the approximate SVD

$\mathbf{C} \leftarrow \mathbf{U}^T\mathbf{F}'\mathbf{V}$
$\tilde{\mathbf{\Sigma}} \leftarrow \texttt{diag}(\mathbf{C})$
$\theta_0 \leftarrow j_0 C_{12} + j_1 C_{21}$
$\theta_1 \leftarrow j_2 C_{02} + j_3 C_{20}$
$\theta_2 \leftarrow j_4 C_{01} + j_5 C_{10}$
$\theta'_0 \leftarrow j_0 C_{21} + j_1 C_{12}$
$\theta'_1 \leftarrow j_2 C_{20} + j_3 C_{02}$
$\theta'_2 \leftarrow j_4 C_{10} + j_5 C_{01}$
$\tilde{\mathbf{U}} \leftarrow \mathbf{U}\mathbf{R}(\mathbf{\Theta})$
$\tilde{\mathbf{V}} \leftarrow \mathbf{V}\mathbf{R}(\mathbf{\Theta}')$
return $(\tilde{\mathbf{U}}, \tilde{\mathbf{\Sigma}}, \tilde{\mathbf{V}})$

end

As shown by equation 6.11, these values have simple close-form values, while the full method would require 2×2 linear solves for each value. As the next section will show, we can further reduce the number of Jacobian values computed to 6, leading to an even more efficient implementation. Timings of our methods show that computing our sparse Jacobian values takes around 9.13ns, whereas computing the full Jacobians takes around 456ns.

6.3.2 Euler Angles SVD Expansion

Using a first-order Taylor expansion to approximate rotation matrices yields non-orthogonal matrices. To overcome this issue, we formulate our approximations of the SVD's rotation matrices in the Euler angles parameter space, ensuring that we only output true rotation matrices. Since we compute our SVD Jacobians on $\mathbf{U}^T\mathbf{F}\mathbf{V}$, we need to compute the Euler angles Jacobian around the identity. Let us recall the Euler angles definition. For Euler angles $\mathbf{\Theta} = (\theta_0, \theta_1, \theta_2)$, we form the rotation matrix:

$$\mathbf{R}(\mathbf{\Theta}) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & c_0 & s_0 \\ 0 & -s_0 & c_0 \end{pmatrix} \begin{pmatrix} c_1 & 0 & -s_2 \\ 0 & 1 & 0 \\ s_2 & 0 & c_1 \end{pmatrix} \begin{pmatrix} c_3 & s_3 & 0 \\ -s_3 & c_3 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{6.12}$$

with c_i, s_i the cosine and sine of θ_i. Inverting the relation and differentiating around $\mathbf{R} = \mathbf{I}$ yields

$$\frac{\partial \mathbf{\Theta}}{\partial \mathbf{R}} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}. \tag{6.13}$$

Combining this Euler angles Jacobian to our SVD Jacobians yields a sparse matrix, detailed in Appendix A, that can be defined by six values $\mathbf{J} = (j_0, j_1, j_2, j_3, j_4, j_5)^T$ that are computed using equation A.5. Once Euler angles for $\tilde{\mathbf{U}}$ and $\tilde{\mathbf{V}}$ have been computed, we reconstruct the rotation matrices with equation 6.12. For additional performance gain we approximate the cosine and sine functions with their second and first order Taylor expansions, respectively. Our complete procedure for computing an approximate SVD is described in algorithm 4.

6.3.3 SVD Memoization Strategy

In the iterative scheme of Projective Dynamics vertex positions update progressively, which opened the way for our Taylor-based approximation of section 6.3.1. In case of very small changes, we argue that the SVD may not need to be updated at all. We therefore opt for a strategy where we choose for each tetrahedron whether to keep the previous SVD as is, use our Taylor approximation of section 6.3.1, or compute a full SVD anew. We propose to determine the precision of our Taylor expansion using the matrix norm of our Euler-SVD Jacobian. Since the spectral norm of a matrix is always smaller than its Frobenius norm, we can safely use the Frobenius norm $\|\mathbf{J}\|_F$ as an approximation of the spectral norm. We can then measure an upper bound on the Euler angles change:

$$\|\Delta\Theta\| \leq \|\mathbf{J}\|_F \left\|\mathbf{F}' - \mathbf{F}\right\|_F. \tag{6.14}$$

Using this upper bound, we can adapt at runtime the required computation for a given tetrahedron. We consider two thresholds $\tau_0 < \tau_1$. If $\|\mathbf{J}\|_F \|\mathbf{F}' - \mathbf{F}\|_F < \tau_0$, we keep the results of the previous SVD computation. If $\tau_0 \leq \|\mathbf{J}\|_F \|\mathbf{F}' - \mathbf{F}\|_F < \tau_1$, we use our Taylor approximation. Otherwise, we compute a full SVD decomposition of $\mathbf{F}'$. To avoid drifting, we only compute our approximation with respect to states for which the regular SVD has been computed. Full SVDs are computed for all tetrahedrons at the beginning of a timestep. For improved precision, we progressively reduce the values of τ_0 and τ_1, halving them after each Projective Dynamics iteration.

6.3.4 Accuracy of the SVD Approximation

To assess the accuracy of our SVD memoization scheme, we sample random deformation gradient-like matrices in the following manner: we sample the Euler angles of the $\mathbf{U}$ and $\mathbf{V}$ matrices, and sample the diagonal matrix $\mathbf{\Sigma}$ by drawing from a normal distribution of mean 1 and standard deviation 0.5. We then sample perturbations to the Euler angles and the diagonal values by drawing from normal distributions of mean 0 and standard deviations σ_{angle} and σ_{diag}, yielding perturbations $\delta\mathbf{U}, \delta\mathbf{\Sigma}, \delta\mathbf{V}$. We compute our SVD Jacobian on the

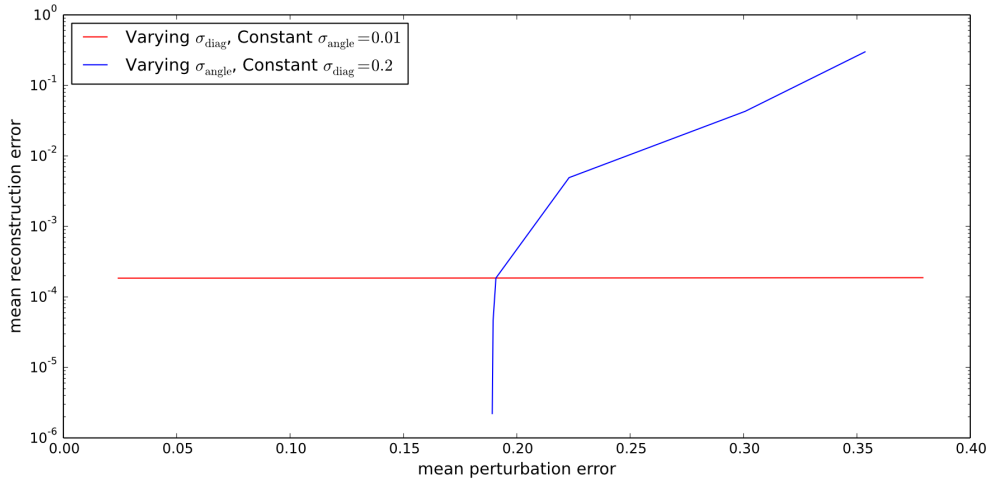


Figure 6.4 – SVD approximation error as a function of the distance to the reference matrix.

unperturbed matrices, apply our method to the perturbed matrices, yielding reconstructed matrices $\hat{\mathbf{U}}$, $\hat{\mathbf{\Sigma}}$, $\hat{\mathbf{V}}$, and measure the error. In figure 6.4, we measure the mean reconstruction Frobenius error $\|\hat{\mathbf{U}}\hat{\mathbf{\Sigma}}\hat{\mathbf{V}}^T - \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\|_F$ as a function of the initial Frobenius error $\|(\mathbf{U} + \delta\mathbf{U})(\mathbf{\Sigma} + \delta\mathbf{\Sigma})(\mathbf{V} + \delta\mathbf{V})^T - \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\|_F$. Since forming $(\mathbf{U} + \delta\mathbf{U})(\mathbf{\Sigma} + \delta\mathbf{\Sigma})(\mathbf{V} + \delta\mathbf{V})^T$ is linear in $\delta\mathbf{\Sigma}$, we expect the precision of our order 1 approximation to not depend on σ_{diag}, which is confirmed in figure 6.4. Up to a certain amount of deviation, our Taylor expansion method yields a satisfying approximation of the SVD. This comforts us in our use of the estimated Euler angle change as a way to predict the approximation error of our method. The reconstructed Euler angle error as a function of σ_{angle} is shown in figure 6.5. We can observe that our approximation leads to a substantial amelioration over plain memoization.

We measure the performance of our method by applying it to 2^{24} randomly sampled matrices. The time to compute our approximated SVD for all those matrices amount to 281ms, translating to 16.7ns per SVD. In comparison, the method of McAdams et al. [124], which consists of hand-written SIMD intrinsics, yields 369ms for the same number of matrices, translating to 22.0ns per SVD. The per-SVD cost of our method is thus 25% lower.

In addition to this raw computation gain, our method brings further performance improvements with the automatic determination of which SVD computations need not be performed (Section 6.3.3). When used inside our simulation, for a typical timestep, around 44% of tetrahedrons do not need recomputation of the SVD, 40% are computed using our Taylor SVD approximation, and the rest requires a full SVD computation. Without halving the τ_0 and τ_1 thresholds after each iteration, the computed state has a relative error of 10^{-3} compared to the converged solution, but halving them on each of the first five iterations gives a computed state with relative error smaller than 10^{-5}.

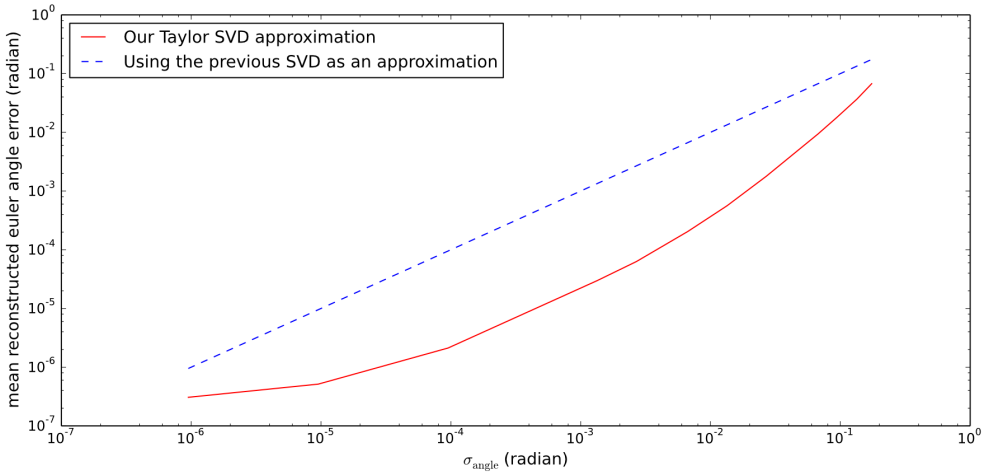


Figure 6.5 – SVD approximation error in terms of Euler angles.

6.4 Experiments

6.4.1 Performance Evaluation

We evaluate the performance of our system by simulating a physical face system with 14k vertices and 45k tetrahedrons. The simulation is performed on a desktop computer with a 3.2 GHz Intel Xeon E5-1650 processor, 6 physical cores and 16GB of main memory. All computations are parallelized using OpenMP where applicable, most notably the collision detection and the forces projection. For realtime performance, we simulate the system with 8 Projective Dynamics iterations. Simulating a frame without collisions or sticky lips interaction takes 28ms, while frames with collisions and sticky lips (around 120 additional constraints) take up to 40ms, making it possible to achieve 25fps, the capture rate of our RGB camera. We provide a breakdown of our simulation timings in figure 6.6. Our two main bottlenecks are the computation of projections, and the evaluation of E_{PD} in the line search procedure. The projections cost is still dominated by the remaining SVD computations. In particular, computing the SVDs for all tetrahedrons at the beginning of a timestep takes 4ms on its own. It should be noted that for these SVD computations, we use the SVD method provided by the Eigen C++ library, which is not as optimized as that of McAdams et al [124]. Therefore, switching to a more performant SVD implementation could bring even more performance.

6.4.2 Simulation Results

We evaluate our performance driven facial physical simulation system by capturing the performance of individuals in front of a webcam and displaying the simulation results in realtime. The blendshapes used for our blendforces are based on the shapes displayed in figure 6.11. Example simulations can be seen in video. An overview of our results is

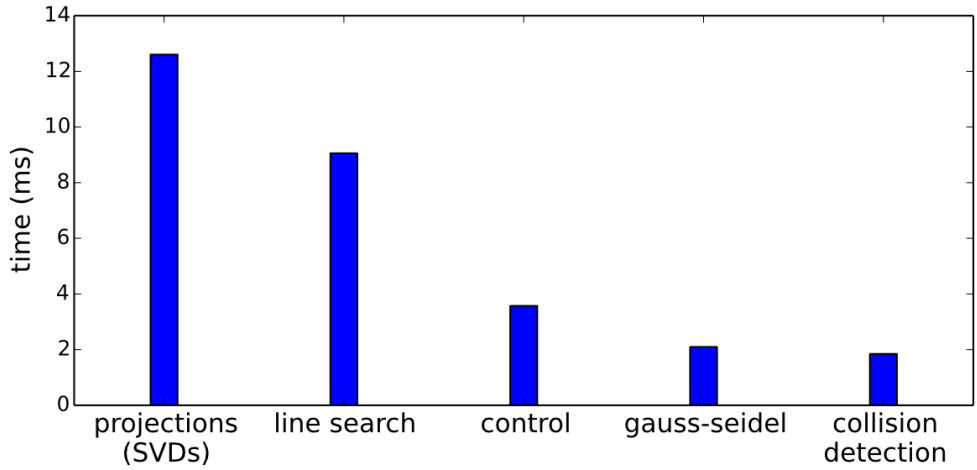


Figure 6.6 – Breakdown of computation times within a timestep involving 50 collision constraints and 110 sticky lips constraints.

shown in figure 6.10. Not only does our system faithfully convey the expression of the user, but contrary to previous realtime facial animation systems, our results showcase physical simulation effects as well. Lips contacts and sticky lips are simulated in realtime. In fact, figures 5.9, 5.13, and 5.14 presented in section 5.4.3 were all generated using this realtime system.

Our physical system can also be used to simulate non-human faces, enabling entertaining inertial effects. We experimented our system on a character with tentacles on the face. Since the tentacles are not attached to the skull and are only a passive element, we disabled the skull attachment force (section 5.1) for tentacle vertices and excluded the tentacles from the blendforces matrix $\mathbf{B}$. We demonstrate the interest of physical simulation for such a character by showing that our method takes the effects of gravity and inertial forces into account (figure 6.7). We show the results of our method versus a simple blendshape animation, and observe that using physical simulation gives the impression of a flesh-made character while the blendshape animation looks plastic-made. Since the interaction with gravity can cause the tentacles to collide, we use our contact forces to prevent self-intersection, as shown in figure 6.8. We can also use our sticky lips force (section 5.4.2) to let tentacles stick to each other for additional artisitic effects (figure 6.9).

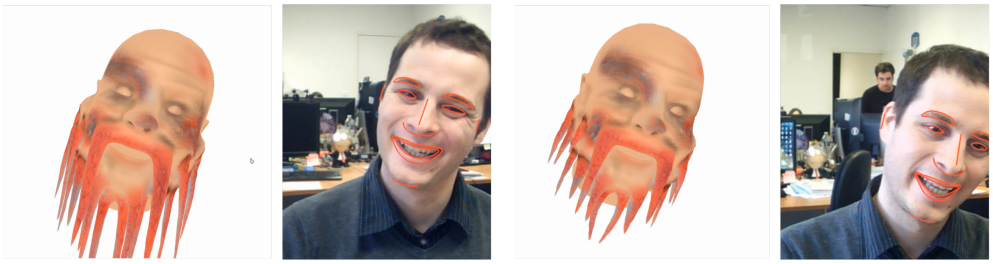


Figure 6.7 – Our method enables integrating facial animation in a realistic environment, allowing the tentacles of the character to react to gravity (left). By comparison, with a plain blendshape animation (right), the character seems to be plastic-made rather than flesh made. This effect is best seen in video.

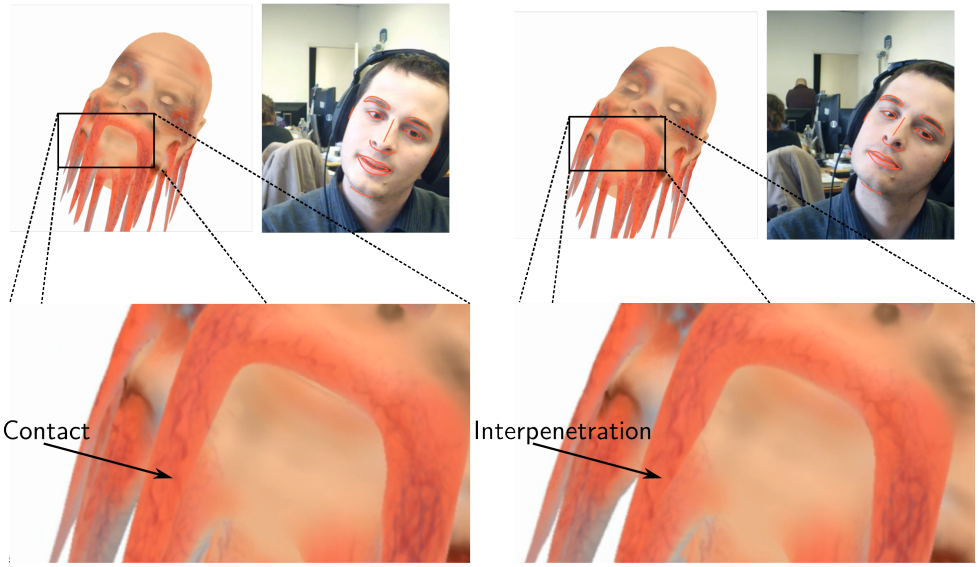


Figure 6.8 – Our system can also prevent self collisions between the tentacles of a stylized character. Left: with collision handling. Right: without collision handling. Without collision handling, the tentacle would penetrate the character's face.

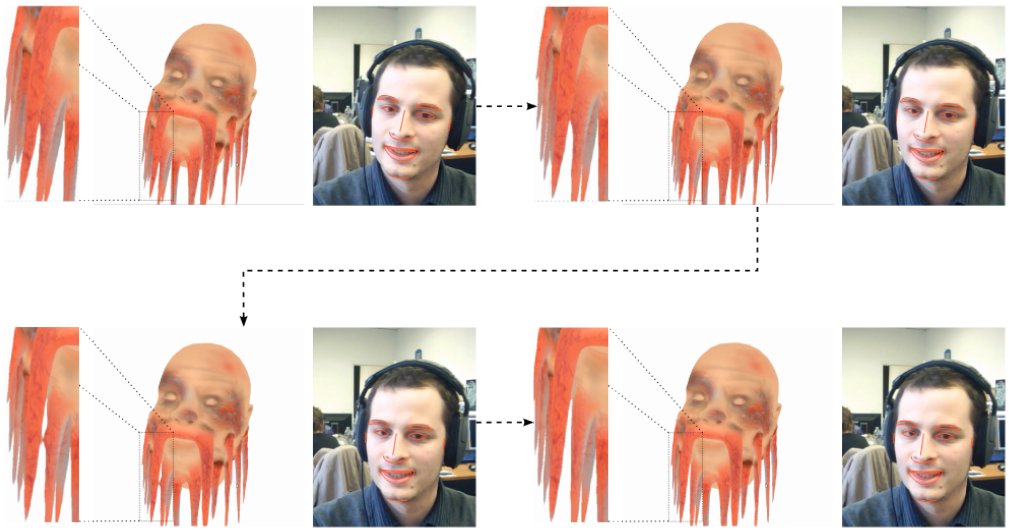


Figure 6.9 – Applying our sticky lips process to tentacles. Gravity causes tentacles to collide, and later on, our sticky force delays the separation between the tentacles. This effect is best seen in video.

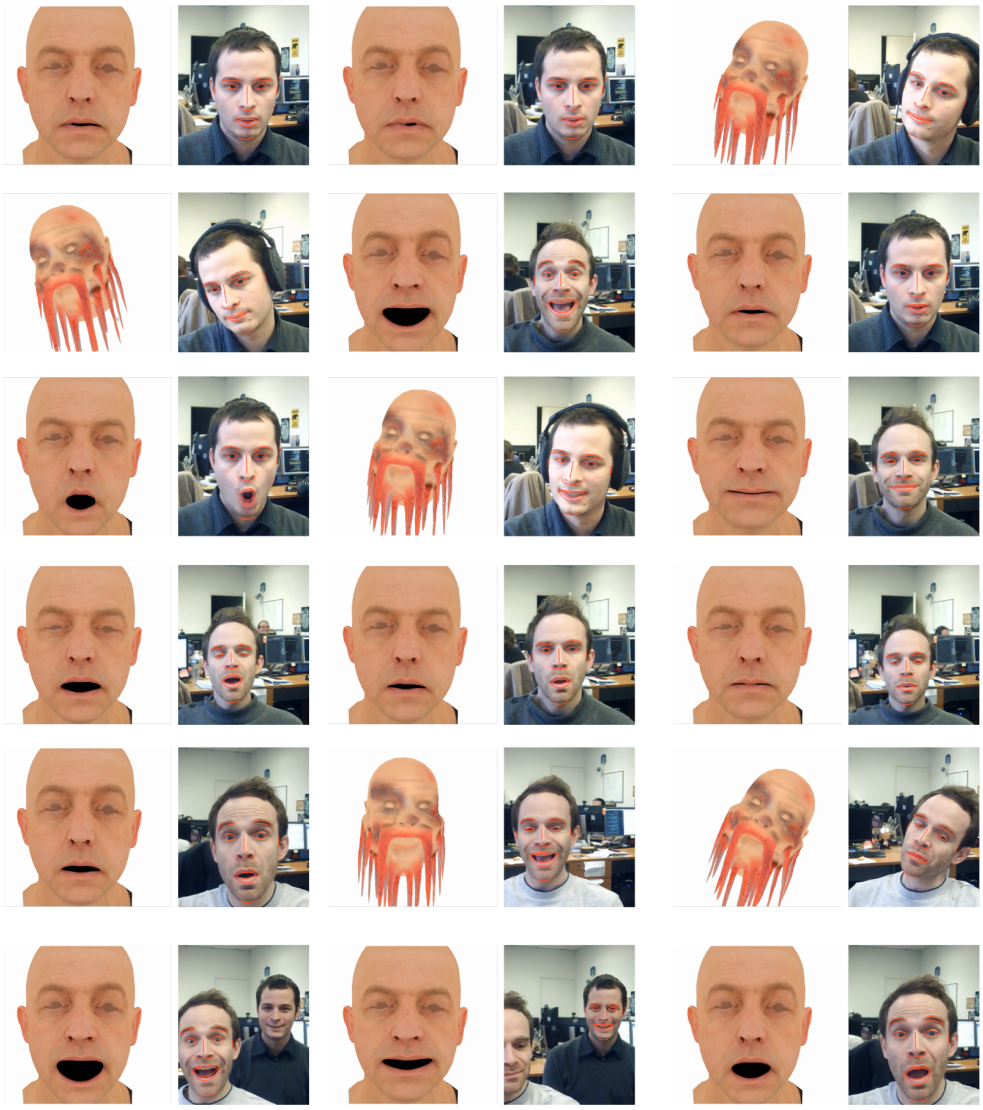


Figure 6.10 – Our system in action. Our method enables the realtime transfer of facial performance from a webcam feed, while enhancing the realism of the resulting animation with physical simulation. The simulated physical effects are the prevention of lips/tentacles self intersection, the sticky lips effect, and the influence of gravity on tentacles.

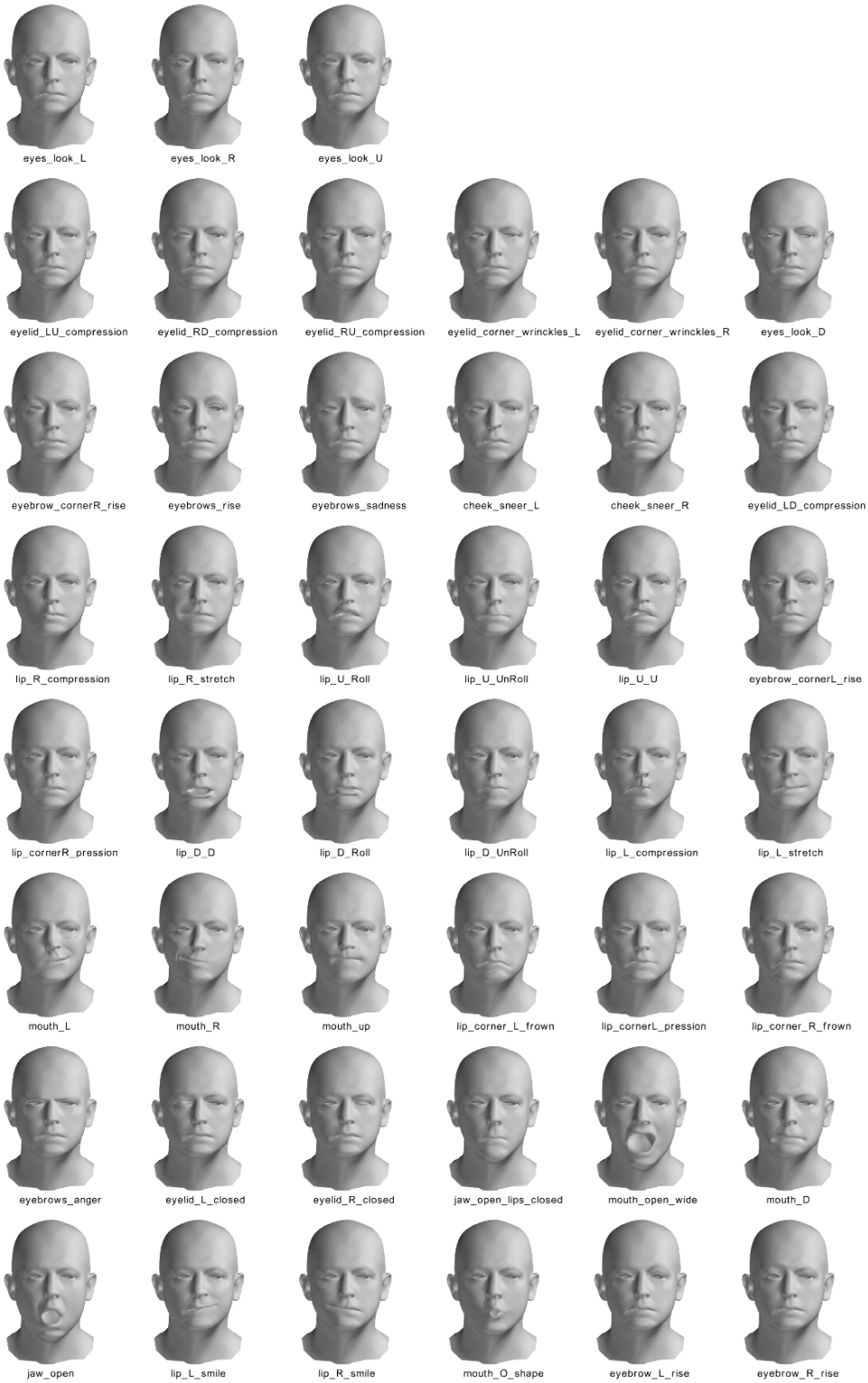


Figure 6.11 – Input blendshapes for our realtime blendforces simulation.

Chapter 7

Conclusion

Contents

7.1 Summary

We now review the contributions and results that we developed throughout this thesis. As announced in the introduction, we structured this thesis as a progressive construction of our main production, namely a realtime practical facial animation based on physical simulation and blendshapes.

In chapter 4 we introduced the concept of blendforces, which lets us define a physical simulation system from a set of facial blendshapes. We derived a control method for this system, enabling motion capture to steer the resulting facial animations. We compared our approach with other recent works that also tried to unify physical simulation and blendshapes, and showed that a key advantage of our method rests with the *simplicity* of the blendforces framework, a key component to enable the controlability by motion capture. We also argued that our formulation is in some sense close to the biomechanical reality, since facial muscles do pull in generally constant directions.

We further explored the blendforces concept in chapter 5, where we demonstrated how realistic internal forces can be derived from surfacic blendshapes to build a face physical system. This chapter brought several contributions: we developed a technique to simulate volumetric forces even though we only require surfacic meshes, and introduced novel internal forces designed to handle lips contacts and the sticky lips phenomenon. We showed that our internal forces, used inside the blendforces framework, were able to produce faithful facial animations, precisely fitting motion capture data, while presenting pleasant qualitative improvements to the resulting animation, such as more lifelike skin

motions, and realistic sticky lips.

In chapter 6, we pushed the limits of our system to be able to produce physically simulated facial animation in realtime, using realtime performance capture from a simple RGB camera as input. To achieve this challenging objective, we contributed key improvements to the Projective Dynamics physical simulation framework, with a new solver more adapted to the simulation of physical systems exposed to sudden changes in the number and nature of internal forces; and an efficient SVD approximation and memoization technique that greatly reduces the computational cost required for simulating volumetric forces. Thanks to these contributions, our system is, to the best of our knowledge, the first instance of a realtime physical simulation system applied to facial animation.

With these contributions we have fully achieved the objectives we had derived in section 1.1. We produced a physical simulation system by leveraging the wide availability of blendshapes; this system is practical since it requires only a blendshapes character and a webcam to produce facial animation from performance capture; and these animations can be produced in realtime, providing the user with realtime feedback.

Publications

This thesis has led to academic publications, which we list below.

Journals

- Barrielle, V., Stoiber, N., and Cagniart, C. BlendForces: a dynamic framework for facial animation. *Computer Graphics Forum (Proceedings of Eurographics)* (2016), 341–352. This paper has received an honorable mention during the Eurographics conference.

- Barrielle, V., and Stoiber, N. Realtime performance driven physical simulation for facial animation. *Conditionally accepted with minor revisions to Computer Graphics Forum* (2017).

International Conferences

- Weber, R., Barrielle, V., Soladié, C., and Séguier, R. High-level geometry-based features of video modality for emotion prediction. In *Proceedings of the 6th International Workshop on Audio/Visual Emotion Challenge* (New York, NY, USA, 2016), AVEC '16, ACM, pp. 51–58.

7.2 Perspectives

Bridging physical simulation and blendshapes methods is only a recent research domain, started in 2011 with the work of Ma et al. [119]. Therefore, there remains a substantial amout of work to bring this domain to its full potential.

While our blendforces approach is simple and produces pleasant results, it remains partly unsatisfying from a biomechanical point of view. Indeed, since some facial muscles, such as sphincters, are not attached to the skull, it is possible, by activating other muscles, to make the muscle slide along the skull, rotating the direction of the pull the displaced muscle will exert when contracted. This breaks the blendforces assumption that the direction of muscle pull remains constant. While this defect is in practice quite limited by the rare occurence of such facial expressions, and by the action of internal forces that prevent this approximation from generating implausible results, a refinement to the blendforces system would be welcome. However, this issue cannot be circumvented by replacing the action of muscles by the interpolation of non-biological internal forces, as attempted by Ichim and colleagues [77]. Indeed, not only is this approach less satisfying from a biomechanical standpoint, it removes the controlability from the system. We conjecture that a piecewise blendforces formulation might be sufficient to tackle this issue. In such a formulation, the blendforces matrix $\mathbf{B}$ would change in some facial configurations, taking into account the rotation of some muscles. This would however present new challenges for performance-driven control, but they possibly could be handled by extending our control strategy augmented with a discrete search among the affine blendforces domains, provided that there are not too many such domains. Since the only part of the face subject to this issue is the sphincter around the lips, this may well be a reasonable assumption.

We have taken advantage of physical simulation to handle interactions that were previously impossible in data-based facial animation techniques. However, we have not exhausted the space of interactions that could be simulated with our system. It would thus be interesting to leverage our contact forces to handle the presence of the eye below the eyelid. This would enable a non-linear trajectory for the eyelids, with the simple additional requirement of a digital character with a mesh representation of the eye. If the eye was also animated, for instance by gaze capture [173], then this contact handling could also be used to deform the eyelids according to the direction of the gaze, further reinforcing the realism of the resulting animation.

Our sticky lips generation process does not require performance capture to produce its effects. As such, it could be used to enhance facial animations produced from other sources. An interesting use case would be the enhancement of animations produced from acoustic data, such as those produced by the audiovisual feedback system of Hueber et al. [73].

Another interesting physical phenomenon taking place on the face is the formation of wrinkles. We identify three axes of research for the simulation of wrinkles within our framework. First, modifications to our internal forces, or to the distribution of their respective stiffnesses, must be investigated to allow for the formation of wrinkles. Then, an optimization procedure similar to the technique described in section 4.3.2 should be devised, in order to be able to recover the appropriate internal forces parameters to simulate wrinkles already present in a blendshapes model. Finally, a procedural model capable of generating plausible wrinkles locations could be used to create wrinkles on a facial model originally lacking such subtle details.

We have presented an effective method to build a face physical system featuring volumetric forces, however not simulating the full facial volume can be problematic. Indeed,

Ichim and colleagues [77] showed that, with a good engineering of internal forces for tetrahedrons belonging to the skull and the jaw, it was possible to recover the circular motion of the jaw. Yet, their technique required the adaptation of a volumetric template for a *human* skull, preventing the application of this technique to non-human characters. Therefore, an interesting future direction would be the extension of the volume building technique of section 5.3.1 to include articulated components such as the jaw.

From an industrial standpoint, our system could be included in animation software packages, bringing enhanced facial animations to numerous users. However, before this can be realized, further work should be realized to improve even more the performance of our system. Our SVD approximation process could be even faster if implemented with careful manual usage of vector instructions, similar to the optimized SVD implementation of McAdams et al. [124]. The recent work by Fratarcangeli and colleagues [64] showcased an efficient Gauss-Seidel solver on the GPU, which could be used to port our entire physical simulation system to the GPU.

Appendix A

SVD Jacobian Derivation

In this annex we derive the closed form expressions of the SVD Jacobian for a "centered" deformation gradient. To this end, we recall results from Papadopoulo and Lourakis [138]. For a 3×3 matrix $\mathbf{C}$ with elements $(C_{ij})_{i,j}$, let $\tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^T$ be its SVD. The Jacobian for the SVD diagonal is defined by:

$$\frac{\partial \tilde{d}_k}{\partial C_{ij}} = \tilde{u}_{ik}\tilde{v}_{jk}. \tag{A.1}$$

The Jacobians for the rotation matrices are given by:

$$\begin{aligned} \frac{\partial \tilde{\mathbf{U}}}{\partial C_{ij}} &= \tilde{\mathbf{U}}\mathbf{\Omega}^{ij}_{\tilde{\mathbf{U}}} \\ \frac{\partial \tilde{\mathbf{V}}}{\partial C_{ij}} &= -\tilde{\mathbf{V}}\mathbf{\Omega}^{ij}_{\tilde{\mathbf{V}}}, \end{aligned} \tag{A.2}$$

where the matrices $\mathbf{\Omega}^{ij}_{\tilde{\mathbf{U}}}$ and $\mathbf{\Omega}^{ij}_{\tilde{\mathbf{V}}}$ are defined by the linear systems:

$$\begin{cases} \tilde{d}_l\mathbf{\Omega}^{ij}_{\tilde{\mathbf{U}}kl} + \tilde{d}_k\mathbf{\Omega}^{ij}_{\tilde{\mathbf{V}}kl} &= \tilde{u}_{ik}\tilde{v}_{jl} \\ \tilde{d}_k\mathbf{\Omega}^{ij}_{\tilde{\mathbf{U}}kl} + \tilde{d}_l\mathbf{\Omega}^{ij}_{\tilde{\mathbf{V}}kl} &= -\tilde{u}_{il}\tilde{v}_{jk}. \end{cases} \tag{A.3}$$

In our case, $\mathbf{C} = \mathbf{U}^T\mathbf{F}\mathbf{V}$, the matrices $\tilde{\mathbf{U}}$ and $\tilde{\mathbf{V}}$ are the identity, which gives a trivial diagonal Jacobian:

$$\frac{\partial \tilde{\mathbf{\Sigma}}}{\partial \mathbf{C}} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}, \tag{A.4}$$

where we vectorize the matrix $\mathbf{C}$ in row-order fashion. Similarly, equation A.3 is greatly simplified, with only four possible right-hand sides: $(0,0)^T, (1,0)^T, (0,-1)^T, (1,-1)^T$.

For right-hand sides $(1,0)^T, (0,-1)^T$, the solutions are of the form:

$$\begin{cases} \mathbf{\Omega}^{ij}_{\tilde{\mathbf{U}}kl} = & \frac{1}{2}\left(\frac{s}{\tilde{d}_k + \tilde{d}_l} + \frac{1}{\tilde{d}_k - \tilde{d}_l}\right) \\ \mathbf{\Omega}^{ij}_{\tilde{\mathbf{V}}kl} = & \frac{1}{2}\left(-\frac{s}{\tilde{d}_k + \tilde{d}_l} + \frac{1}{\tilde{d}_k - \tilde{d}_l}\right), \end{cases} \tag{A.5}$$

for k and l such that $|\tilde{d}_k| \neq |\tilde{d}_l|$ and with $s = \pm 1$. For the $(1,-1)^T$ right-hand side, the solution is:

$$\begin{cases} \mathbf{\Omega}^{ij}_{\tilde{\mathbf{U}}kl} = & \frac{1}{\tilde{d}_k - \tilde{d}_l} \\ \mathbf{\Omega}^{ij}_{\tilde{\mathbf{V}}kl} = & -\frac{1}{\tilde{d}_k - \tilde{d}_l}, \end{cases} \tag{A.6}$$

again with diagonal values of different magnitudes. As suggested by Papadopoulo and Lourakis [138], the equality case must be handled with a least-squares solve. This least-squares solve has a closed form solution as well. Suppose $\tilde{d}_k = \pm\tilde{d}_l$. Then only the $(1,0)^T, (0,-1)^T$ right-hand sides yield nonzero solutions, of the form:

$$\mathbf{\Omega}^{ij}_{\tilde{\mathbf{U}}kl} = \mathbf{\Omega}^{ij}_{\tilde{\mathbf{V}}kl} = \pm\frac{1}{2}\frac{1}{\tilde{d}_k \pm \tilde{d}_l}. \tag{A.7}$$

In practice, since we're only interested in the Euler angles Jacobian $\mathbf{J}$ defined in section 6.3.2, we don't need to compute the solutions from equation A.6 (these values correspond to all-zeros columns in equation 6.13). Putting aside the equality case, the close form for the values $\mathbf{J} = (j_0, j_1, j_2, j_3, j_4, j_5)^T$ is:

$$\begin{cases} j_0 = & \frac{1}{2}\left(\frac{1}{\tilde{d}_2 + \tilde{d}_1} + \frac{1}{\tilde{d}_2 - \tilde{d}_1}\right) \\ j_1 = & \frac{1}{2}\left(-\frac{1}{\tilde{d}_2 + \tilde{d}_1} + \frac{1}{\tilde{d}_2 - \tilde{d}_1}\right) \\ j_2 = & \frac{1}{2}\left(-\frac{1}{\tilde{d}_2 + \tilde{d}_0} - \frac{1}{\tilde{d}_2 - \tilde{d}_0}\right) \\ j_3 = & \frac{1}{2}\left(\frac{1}{\tilde{d}_2 + \tilde{d}_0} - \frac{1}{\tilde{d}_2 - \tilde{d}_0}\right) \\ j_4 = & \frac{1}{2}\left(\frac{1}{\tilde{d}_1 + \tilde{d}_0} + \frac{1}{\tilde{d}_1 - \tilde{d}_0}\right) \\ j_5 = & \frac{1}{2}\left(-\frac{1}{\tilde{d}_1 + \tilde{d}_0} + \frac{1}{\tilde{d}_1 - \tilde{d}_0}\right). \end{cases} \tag{A.8}$$

Equation A.7 informs us that in case of equal diagonal values, the fraction that would become undefined should be replaced by zero.

Bibliography

[1] Abboud, B., and Davoine, F. Appearance factorization for facial expression analysis. In *British Machine Vision Conference* (2004), BMVA, pp. 1–10.

[2] Aina, O. O. Generating anatomical substructures for physically-based facial animation. part 1: A methodology for skull fitting. *Vis. Comput. 25*, 5-7 (2009), 617–625.

[3] Aina, O. O., and Zhang, J. J. Automatic muscle generation for physically-based facial animation. In *ACM SIGGRAPH 2010 Posters* (New York, NY, USA, 2010), SIGGRAPH '10, ACM, pp. 105:1–105:1.

[4] Akhter, I., Simon, T., Khan, S., Matthews, I., and Sheikh, Y. Bilinear spatiotemporal basis models. *ACM Trans. Graph. 31*, 2 (2012), 17:1–17:12.

[5] Alexander, O., Rogers, M., Lambeth, W., Chiang, M., and Debevec, P. The digital emily project: Photoreal facial modeling and animation. In *ACM SIGGRAPH 2009 Courses* (New York, NY, USA, 2009), SIGGRAPH '09, ACM, pp. 12:1–12:15.

[6] Asthana, A., Khwaja, A., and Goecke, R. Automatic frontal face annotation and AAM building for arbitrary expressions from a single frontal image only. In *Proceedings of the 16th IEEE International Conference on Image Processing* (Piscataway, NJ, USA, 2009), ICIP'09, IEEE Press, pp. 2417–2420.

[7] Barbarino, G., Jabareen, M., Trzewik, J., and Mazza, E. Physically based finite element model of the face. In *Proceedings of the 4th International Symposium on Biomedical Simulation* (Berlin, Heidelberg, 2008), ISBMS '08, Springer-Verlag, pp. 1–10.

[8] Barrielle, V., and Stoiber, N. Realtime performance driven physical simulation for facial animation. *Conditionally accepted with minor revisions to Computer Graphics Forum* (2017).

[9] Barrielle, V., Stoiber, N., and Cagniart, C. BlendForces: a dynamic framework for facial animation. *Computer Graphics Forum (Proceedings of Eurographics)* (2016), 341–352.

[10] Basu, S., Oliver, N., and Pentland, A. 3d lip shapes from video: A combined physical-statistical model. *Speech Commun. 26*, 1-2 (1998), 131–148.

[11] Basu, S., Oliver, N., and Pentland, A. 3d modeling and tracking of human lip motions. In *Proceedings of the Sixth International Conference on Computer Vision* (Washington, DC, USA, 1998), ICCV '98, IEEE Computer Society, pp. 337–343.

[12] Beeler, T., and Bradley, D. Rigid stabilization of facial expressions. *ACM Trans. Graph. 33*, 4 (July 2014), 44:1–44:9.

[13] Beeler, T., Hahn, F., Bradley, D., Bickel, B., Beardsley, P., Gotsman, C., Sumner, R. W., and Gross, M. High-quality passive facial performance capture using anchor frames. In *ACM SIGGRAPH 2011 papers* (New York, NY, USA, 2011), SIGGRAPH '11, ACM, pp. 75:1–75:10.

[14] Berar, M., Desvignes, M., Bailly, G., and Payan, Y. 3d statistical facial reconstruction. In *ISPA 2005. Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis, 2005.* (Sept 2005), IEEE, pp. 365–370.

[15] Bermano, A. H., Bradley, D., Beeler, T., Zund, F., Nowrouzezahrai, D., Baran, I., Sorkine-Hornung, O., Pfister, H., Sumner, R. W., Bickel, B., and Gross, M. Facial performance enhancement using dynamic shape space analysis. *ACM Trans. Graph. 33*, 2 (2014), 13:1–13:12.

[16] Bhat, K. S., Goldenthal, R., Ye, Y., Mallet, R., and 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.

[17] Bickel, B., Botsch, M., Angst, R., Matusik, W., Otaduy, M., Pfister, H., and Gross, M. Multi-scale capture of facial geometry and motion. *ACM Trans. Graph. 26*, 3 (July 2007).

[18] Bickel, B., Bächer, M., Otaduy, M. A., Matusik, W., Pfister, H., and Gross, M. Capture and modeling of non-linear heterogeneous soft tissue. In *ACM SIGGRAPH 2009 Papers* (New York, NY, USA, 2009), SIGGRAPH '09, ACM, pp. 89:1–89:9.

[19] Bickel, B., Lang, M., Botsch, M., Otaduy, M. A., and Gross, M. Pose-space animation and transfer of facial details. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2008), SCA '08, Eurographics Association, pp. 57–66.

[20] Blanz, V., Basso, C., Poggio, T., and Vetter, T. Reanimating faces in images and video. *Computer Graphics Forum 22*, 3 (2003), 641–650.

[21] Blanz, V., and Vetter, T. A morphable model for the synthesis of 3d faces. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1999), SIGGRAPH '99, ACM Press/Addison-Wesley Publishing Co., pp. 187–194.

[22] Borges, A. F. Relaxed skin tension lines (rstl) versus other skin lines. *Plastic and reconstructive surgery 73*, 1 (1984), 144–150.

[23] Botsch, M., Kobbelt, L., Pauly, M., Alliez, P., and Levy, B. *Polygon Mesh Processing*. AK Peters, 2010.

[24] Bouaziz, S., Martin, S., Liu, T., Kavan, L., and Pauly, M. Projective Dynamics: Fusing constraint projections for fast simulation. *ACM Trans. Graph. 33*, 4 (July 2014), 154:1–154:11.

[25] Bouaziz, S., and Pauly, M. Semi-supervised facial animation retargeting. Tech. rep., 2014.

[26] Bouaziz, S., Wang, Y., and Pauly, M. Online modeling for realtime facial animation. *ACM Trans. Graph. 32*, 4 (2013), 40:1–40:10.

[27] Bridson, R., Fedkiw, R., and 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.

[28] Buck, I., Finkelstein, A., Jacobs, C., Klein, A., Salesin, D. H., Seims, J., Szeliski, R., and Toyama, K. Performance-driven hand-drawn animation. In *Proceedings of the 1st international symposium on Non-photorealistic animation and rendering* (New York, NY, USA, 2000), NPAR '00, ACM, pp. 101–108.

[29] Bui, T. D., Heylen, D., Poel, M., and Nijholt, A. Exporting vector muscles for facial animation. In *Proceedings of the 3rd International Conference on Smart Graphics* (Berlin, Heidelberg, 2003), SG'03, Springer-Verlag, pp. 251–260.

[30] Byun, M., and Badler, N. I. FacEMOTE: Qualitative parametric modifiers for facial animations. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2002), SCA '02, ACM, pp. 65–71.

[31] Cao, C., Bradley, D., Zhou, K., and Beeler, T. Real-time high-fidelity facial performance capture. *ACM Trans. Graph. 34*, 4 (July 2015), 46:1–46:9.

[32] Cao, C., Hou, Q., and Zhou, K. Displaced dynamic expression regression for real-time facial tracking and animation. *ACM Trans. Graph. 33*, 4 (July 2014), 43:1–43:10.

[33] Cao, C., Weng, Y., Lin, S., and Zhou, K. 3D shape regression for real-time facial animation. *ACM Trans. Graph. 32*, 4 (2013), 41:1–41:10.

[34] Cao, C., Weng, Y., Zhou, S., Tong, Y., and Zhou, K. FaceWarehouse : a 3D facial expression database for visual computing. *IEEE Transactions on Visualization and Computer Graphics 20*, 3 (2014), 413–425.

[35] Cao, C., Wu, H., Weng, Y., Shao, T., and Kun, Z. Real-time facial animation with image-based dynamic avatars. *ACM Trans. Graph. 35*, 4 (2016), 126:1–126:12.

[36] Cao, Y., Faloutsos, P., Kohler, E., and Pighin, F. Real-time speech motion synthesis from recorded motions. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2004), SCA '04, Eurographics Association, pp. 345–353.

[37] Casas, D., Feng, A., Alexander, O., Fyffe, G., Debevec, P., Ichikari, R., Li, H., Olszewski, K., Suma, E., and Shapiro, A. Rapid photorealistic blendshape modeling from RGB-D sensors. In *Proceedings of the 29th International Conference on Computer Animation and Social Agents* (New York, NY, USA, 2016), CASA '16, ACM, pp. 121–129.

[38] Chabanas, M., Luboz, V., and Payan, Y. Patient specific finite element model of the face soft tissues for computer-assisted maxillofacial surgery. *Medical Image Analysis 7*, 2 (2003), 131 – 151.

[39] Chai, J.-x., Xiao, J., and Hodgins, J. Vision-based control of 3d facial animation. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2003), SCA '03, Eurographics Association, pp. 193–206.

[40] Chang, E., and Jenkins, O. C. Sketching articulation and pose for facial animation. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2006), SCA '06, Eurographics Association, pp. 271–280.

[41] Chang, Y.-J., and Ezzat, T. Transferable videorealistic speech animation. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2005), SCA '05, ACM, pp. 143–151.

[42] Chen, Y., Davis, T. A., Hager, W. W., and Rajamanickam, S. Algorithm 887: CHOLMOD, Supernodal Sparse Cholesky Factorization and Update/Downdate. *ACM Trans. Math. Softw. 35*, 3 (Oct. 2008), 22:1–22:14.

[43] Choe, B., and Ko, H.-S. Analysis and synthesis of facial expressions with hand-generated muscle actuation basis. In *ACM SIGGRAPH 2005 Courses* (New York, NY, USA, 2005), SIGGRAPH '05, ACM.

[44] Choe, B., Lee, H., and Ko, H.-S. Performance-driven muscle-based facial animation. *The Journal of Visualization and Computer Animation 12*, 2 (2001), 67–79.

[45] Chuang, E., and Bregler, C. Performance driven facial animation using blendshape interpolation. Tech. Rep. Technical Report CS-TR-2002-02, Stanford University, 2002.

[46] Chuang, E., and Bregler, C. Mood swings: expressive speech animation. *ACM Trans. Graph. 24*, 2 (2005), 331–347.

[47] Cong, M., Bao, M., E., J. L., Bhat, K. S., and Fedkiw, R. Fully automatic generation of anatomical face simulation models. In *Proceedings of the 14th ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (New York, NY, USA, 2015), SCA '15, ACM, pp. 175–183.

[48] Cong, M., Bhat, K. S., and Fedkiw, R. Art-directed muscle simulation for high-end facial animation. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2016), SCA '16, Eurographics Association, pp. 119–127.

[49] Costigan, T., Prasad, M., and McDonnell, R. Facial retargeting using neural networks. In *Proceedings of the Seventh International Conference on Motion in Games* (New York, NY, USA, 2014), MIG '14, ACM, pp. 31–38.

[50] DeCarlo, D., Metaxas, D., and Stone, M. An anthropometric face model using variational techniques. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1998), SIGGRAPH '98, ACM, pp. 67–74.

[51] Deng, Z., Chiang, P.-Y., Fox, P., and Neumann, U. Animating blendshape faces by cross-mapping motion capture data. In *Proceedings of the 2006 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2006), I3D '06, ACM, pp. 43–48.

[52] Deng, Z., and Ma, X. Perceptually Guided Expressive Facial Animation. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2008), SCA '08, Eurographics Association, pp. 67–76.

[53] Diamant, R., and Simantov, J. Uncharted 2 character pipeline: An in depth look at the creation of u2's characters. In *GDC* (2010).

[54] Dutreve, L., Meyer, A., and Bouakaz, S. Feature Points Based Facial Animation Retargeting. In *Proceedings of the 2008 ACM Symposium on Virtual Reality Software and Technology* (New York, NY, USA, 2008), VRST '08, ACM, pp. 197–200.

[55] DYNAMIXYZ. Performer2, facial markerless and marker performance capture, 2013. http://www.dynamixyz.com.

[56] Edwards, P., Landreth, C., Fiume, E., and Singh, K. JALI: An animator-centric viseme model for expressive lip synchronization. *ACM Trans. Graph. 35*, 4 (July 2016), 127:1–127:11.

[57] Ekman, P., and Friesen, W. V. *Manual for the facial action coding system.* Consulting Psychologists Press, Palo Alto CA, 1977.

[58] Ekman, P., and Friesen, W. V. Facial action coding system: A technique for the measurement of facial movement. *CA: Consulting Psychologists Press. Ellsworth, PC, & Smith, CA (1988). From appraisal to emotion: Differences among unpleasant feelings. Motivation and Emotion 12* (1978), 271–302.

[59] Essa, I., Basu, S., Darrell, T., and 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–.

[60] Evans, C., Martinsson, L., and Herfort, S. Building an empire: Asset production in ryse. In *ACM SIGGRAPH 2014 Courses* (New York, NY, USA, 2014), SIGGRAPH '14, ACM, pp. 18:1–18:49.

[61] Fanelli, G., and Fratarcangeli, M. A non-invasive approach for driving virtual talking heads from real facial movements. In *3DTV Conference, 2007* (2007), pp. 1–4.

[62] Feng, W.-W., Kim, B.-U., and Yu, Y. Real-time data driven deformation using kernel canonical correlation analysis. In *ACM SIGGRAPH 2008 Papers* (New York, NY, USA, 2008), SIGGRAPH '08, ACM, pp. 91:1–91:9.

[63] Fratarcangeli, M. Physically based synthesis of animatable face models. In *Proceedings of the 2nd International Workshop on Virtual Reality and Physical Simulation (VRIPHYS05)* (2005).

[64] Fratarcangeli, M., Tibaldo, V., and Pellacini, F. Vivace: A practical Gauss-seidel method for stable soft body dynamics. *ACM Trans. Graph. 35*, 6 (Nov. 2016), 214:1–214:9.

[65] Fried, O., Shechtman, E., Goldman, D. B., and Finkelstein, A. Perspective-aware manipulation of portrait photos. *ACM Trans. Graph. 35*, 4 (July 2016), 128:1–128:10.

[66] Gao, Y., Zhao, Q., Hao, A., Sezgin, T. M., and Dodgson, N. A. Automatic construction of 3d animatable facial avatars. *Comput. Animat. Virtual Worlds 21* (2010), 343–354.

[67] Garrido, P., Zollhöfer, M., Casas, D., Valgaerts, L., Varanasi, K., Pérez, P., and Theobalt, C. Reconstruction of personalized 3d face rigs from monocular video. *ACM Trans. Graph. 35*, 3 (May 2016), 28:1–28:15.

[68] Gérard, J.-M., Ohayon, J., Luboz, V., Perrier, P., and Payan, Y. Non linear elastic properties of the lingual and facial tissues assessed by indentation technique. application to the biomechanics of speech production. *Medical Engineering & Physics 27 (10)* (2005), 884–892.

[69] Hahn, F., Martin, S., Thomaszewski, B., Sumner, R., Coros, S., and Gross, M. Rig-space physics. *ACM Trans. Graph. 31*, 4 (2012), 72:1–72:8.

[70] Harmon, D., Vouga, E., Tamstorf, R., and Grinspun, E. Robust treatment of simultaneous collisions. In *ACM SIGGRAPH 2008 Papers* (New York, NY, USA, 2008), SIGGRAPH '08, ACM, pp. 23:1–23:4.

[71] Havaldar, P. Sony pictures imageworks. In *ACM SIGGRAPH 2006 Courses* (New York, NY, USA, 2006), SIGGRAPH '06, ACM.

[72] Huang, D., and De la Torre, F. Bilinear kernel reduced rank regression for facial expression synthesis. In *Proceedings of the 11th European Conference on Computer Vision: Part II* (Berlin, Heidelberg, 2010), ECCV'10, Springer-Verlag, pp. 364–377.

[73] Hueber, T., Badin, P., Bailly, G., Ben Youssef, A., Elisei, F., Denby, B., and Chollet, G. Statistical mapping between articulatory and acoustic data. application to silent speech interface and visual articulatory feedback. In *1st International Workshop on Performative Speech and Singing Synthesis [P3S]* (Vancouver, Canada, Mar 2011).

[74] Hunty, M. d. l., Asthana, A., and Goecke, R. Linear facial expression transfer with active appearance models. In *Proceedings of the 2010 20th International Conference on Pattern Recognition* (Washington, DC, USA, 2010), ICPR '10, IEEE Computer Society, pp. 3789–3792.

[75] Ichim, A. E., Bouaziz, S., and Pauly, M. Dynamic 3d avatar creation from hand-held video input. *ACM Trans. Graph. 34*, 4 (July 2015), 45:1–45:14.

[76] Ichim, A.-E., Kadleček, P., Kavan, L., and Pauly, M. Phace: Physics-based face modeling and animation. *ACM Transactions on Graphics (TOG) 36*, 4 (2017).

[77] Ichim, A.-E., Kavan, L., Nimier-David, M., and Pauly, M. Building and animating user-specific volumetric face rigs. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2016), SCA '16, Eurographics Association, pp. 107–117.

[78] Jacobson, A., Deng, Z., Kavan, L., and Lewis, J. P. Skinning: Real-time shape deformation (full text not available). In *ACM SIGGRAPH 2014 Courses* (New York, NY, USA, 2014), SIGGRAPH '14, ACM, pp. 24:1–24:1.

[79] Jolliffe, I. T. *Principal Component Analysis*. Springer, New York, NY, USA, 1986.

[80] Joshi, P., Tien, W. C., Desbrun, M., and Pighin, F. Learning controls for blend shape based realistic facial animation. In *ACM SIGGRAPH 2005 Courses* (New York, NY, USA, 2005), SIGGRAPH '05, ACM.

[81] Kalra, P., Mangili, A., Thalmann, N. M., and Thalmann, D. Simulation of facial muscle actions based on rational free form deformations. *Computer Graphics Forum 11*, 3 (1992), 59–69.

[82] Kazemi, V., and Sullivan, J. One millisecond face alignment with an ensemble of regression trees. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition* (Washington, DC, USA, 2014), CVPR '14, IEEE Computer Society, pp. 1867–1874.

[83] Keeve, E., Girod, S., Pfeifle, P., and 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. 21–28.

[84] Kholgade, N., Matthews, I., and 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.

[85] Kim, H.-J., Öztireli, A. C., Shin, I.-K., Gross, M., and Choi, S.-M. Interactive generation of realistic facial wrinkles from sketchy drawings. *Computer Graphics Forum 34*, 2 (2015), 179–191.

[86] Kim, P. H., Seol, Y., Song, J., and Noh, J. Facial retargeting by adding supplemental blendshapes. In *Pacific Graphics Short Papers* (2011), The Eurographics Association, pp. 89–92.

[87] Kim, T., and James, D. L. Skipping steps in deformable simulation with online model reduction. In *ACM SIGGRAPH Asia 2009 Papers* (New York, NY, USA, 2009), SIGGRAPH Asia '09, ACM, pp. 123:1–123:9.

[88] Koch, R. M., Gross, M. H., and Bosshard, A. A. Emotion editing using finite elements. *Computer Graphics Forum 17*, 3 (1998), 295–302.

[89] Koch, R. M., Gross, M. H., Carls, F. R., von Büren, D. F., Fankhauser, G., and 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.

[90] Kozlov, Y., Bradley, D., Bächer, M., Thomaszewski, B., Beeler, T., and Gross, M. Enriching facial blendshape rigs with physical simulation. *Computer Graphics Forum (Proc. Eurographics) 36*, 2 (2017), 75–84.

[91] Kuratate, T., Vatikiotis-Bateson, E., and Yehia, H. C. Cross-subject face animation driven by facial motion mapping. In *ISPE International Conference on Concurrent Engineering : Research and Applications* (2003), Swets & Zeitlinger, pp. 971–979.

[92] Kähler, K., Haber, J., and Seidel, H.-P. Geometry-based muscle modeling for facial animation. In *No Description on Graphics Interface 2001* (Toronto, Ont., Canada, Canada, 2001), GRIN'01, Canadian Information Processing Society, pp. 37–46.

[93] Kähler, K., Haber, J., and Seidel, H.-P. Reanimating the dead: Reconstruction of expressive faces from skull data. In *ACM SIGGRAPH 2003 Papers* (New York, NY, USA, 2003), SIGGRAPH '03, ACM, pp. 554–561.

[94] Kähler, K., Haber, J., Yamauchi, H., and Seidel, H.-P. Head shop: Generating animated head models with anatomical structure. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2002), SCA '02, ACM, pp. 55–63.

[95] Lau, M., Chai, J., Xu, Y.-Q., and Shum, H.-Y. Face poser: interactive modeling of 3d facial expressions using model priors. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2007), SCA '07, Eurographics Association, pp. 161–170.

[96] Lavagetto, F., and Pockaj, R. The facial animation engine: Toward a high-level interface for the design of MPEG-4 compliant animated faces. *IEEE Trans. Cir. and Sys. for Video Technol. 9*, 2 (1999), 277–289.

[97] Lewis, J., Mo, Z., Anjyo, K., and Rhee, T. Probable and improbable faces. In *Mathematical Progress in Expressive Image Synthesis I*. Springer, 2014, pp. 21–30.

[98] Lewis, J. P., Anjyo, K., Rhee, T., Zhang, M., Pighin, F., and Deng, Z. Practice and Theory of Blendshape Facial Models. In *Eurographics 2014 - State of the Art Reports* (2014), S. Lefebvre and M. Spagnuolo, Eds., The Eurographics Association.

[99] Lewis, J. P., and Anjyo, K.-i. Direct manipulation blendshapes. *IEEE Comput. Graph. Appl. 30*, 4 (2010), 42–50.

[100] Lewis, J. P., Cordner, M., and Fong, N. Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2000), SIGGRAPH '00, ACM Press/Addison-Wesley Publishing Co., pp. 165–172.

[101] Lewis, J. P., Mooser, J., Deng, Z., and Neumann, U. Reducing blendshape interference by selected motion attenuation. In *Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2005), I3D '05, ACM, pp. 25–29.

[102] Li, B., Zhang, Q., Zhou, D., and Wei, X. Facial animation based on feature points. *TELKOMNIKA Indonesian Journal of Electrical Engineering 11*, 3 (2013), 1697–1706.

[103] Li, H., Trutoiu, L., Olszewski, K., Wei, L., Trutna, T., Hsieh, P.-L., Nicholls, A., and Ma, C. Facial performance sensing head-mounted display. *ACM Trans. Graph. 34*, 4 (2015), 47:1–47:9.

[104] Li, H., Weise, T., and Pauly, M. Example-based facial rigging. In *ACM SIGGRAPH 2010 papers* (New York, NY, USA, 2010), SIGGRAPH '10, ACM, pp. 32:1–32:6.

[105] Li, H., Yu, J., Ye, Y., and Bregler, C. Realtime facial animation with on-the-fly correctives. *ACM Trans. Graph. 32*, 4 (2013), 42:1–42:10.

[106] Li, J., Xu, W., Cheng, Z., Xu, K., and Klein, R. Lightweight wrinkle synthesis for 3d facial modeling and animation. *Computer Aided Design 58* (2015), 117–122.

[107] Li, K., Xu, F., Wang, J., Dai, Q., and Liu, Y. A data-driven approach for facial expression synthesis in video. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2012), pp. 57–64.

[108] Li, Q., and Deng, Z. Orthogonal-blendshape-based editing system for facial motion capture data. *IEEE Comput. Graph. Appl. 28*, 6 (2008), 76–82.

[109] Li, Y., Xu, H., and Barbič, J. Enriching triangle mesh animations with physically based simulation. Tech. rep., University of Southern California, 2015.

[110] Liu, K.-Y., Ma, W.-C., Chang, C.-F., Wang, C.-C., and Debevec, P. A framework for locally retargeting and rendering facial performance. *Comput. Animat. Virtual Worlds 22*, 2-3 (2011), 159–167.

[111] Liu, T., Bargteil, A. W., O'Brien, J. F., and Kavan, L. Fast simulation of mass-spring systems. *ACM Transactions on Graphics 32*, 6 (Nov. 2013), 209:1–7. Proceedings of ACM SIGGRAPH Asia 2013, Hong Kong.

[112] Liu, T., Bouaziz, S., and Kavan, L. Quasi-newton methods for real-time simulation of hyperelastic materials. *ACM Trans. Graph. 36*, 3 (2017), 23:1–23:16.

[113] Liu, X., Mao, T., Xia, S., Yu, Y., and Wang, Z. Facial animation by optimized blendshapes from motion capture data. *Comput. Animat. Virtual Worlds 19*, 3-4 (2008), 235–245.

[114] Liu, X., Xia, S., Fan, Y., and Wang, Z. Exploring non-linear relationship of blendshape facial animation. *Computer Graphics Forum 30*, 6 (2011), 1655–1666.

[115] Liu, Y., Xu, F., Chai, J., Tong, X., Wang, L., and Huo, Q. Video-audio driven real-time facial animation. *ACM Trans. Graph. 34*, 6 (Oct. 2015), 182:1–182:10.

[116] Ma, W.-C., Barbati, M., and Lewis, J. P. A facial composite editor for blendshape characters. In *Proceedings of the Digital Production Symposium* (New York, NY, USA, 2012), DigiPro '12, ACM, pp. 21–26.

[117] Ma, W.-C., Fyffe, G., and 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.

[118] Ma, W.-C., Jones, A., Chiang, J.-Y., Hawkins, T., Frederiksen, S., Peers, P., Vukovic, M., Ouhyoung, M., and 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.

[119] Ma, W.-C., Wang, Y.-H., Fyffe, G., Barbič, J., Chen, B.-Y., and Debevec, P. A blendshape model that incorporates physical interaction. In *SIGGRAPH Asia 2011 Posters* (New York, NY, USA, 2011), SA '11, ACM, pp. 35:1–35:1.

[120] Ma, X., Le, B. H., and Deng, Z. Style learning and transferring for facial animation editing. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2009), SCA '09, ACM, pp. 123–132.

[121] Macedo, I., Brazil, E., and Velho, L. Expression transfer between photographs through multilinear AAM's. In *Computer Graphics and Image Processing, 2006. SIBGRAPI '06. 19th Brazilian Symposium on* (2006), pp. 239–246.

[122] Magnenat-Thalmann, N., Primeau, E., and Thalmann, D. Abstract muscle action procedures for human face animation. *The Visual Computer 3*, 5 (1988), 290–297.

[123] Martin, S., Thomaszewski, B., Grinspun, E., and Gross, M. Example-based elastic materials. In *ACM SIGGRAPH 2011 Papers* (New York, NY, USA, 2011), SIGGRAPH '11, ACM, pp. 72:1–72:8.

[124] McAdams, A., Selle, A., Tamstorf, R., Teran, J., Sifakis, E., and Studios, W. D. A. Computing the singular value decomposition of 3×3 matrices with minimal branching and elementary floating point operations. Tech. rep., 2011.

[125] Meyer, M., and Anderson, J. Key point subspace acceleration and soft caching. *ACM Trans. Graph. 26*, 3 (July 2007).

[126] Morishima, S., Ishikawa, T., and Terzopoulos, D. Facial muscle parameter decision from 2d frontal image. In *Proceedings of the 14th International Conference on Pattern Recognition-Volume 1 - Volume 1* (Washington, DC, USA, 1998), ICPR '98, IEEE Computer Society, pp. 160–162.

[127] MPEG-4 SNHC. Information technology-generic coding of audio-visual objects part 2: Visual. *ISO/IEC 14996-2, Final draft of international standard, ISO/IEC JTCI/SC29/WG11 N2501* (1998).

[128] Müller, M., Bender, J., Chentanez, N., and Macklin, M. A robust method to extract the rotational part of deformations. In *Proceedings of the 9th International Conference on Motion in Games* (New York, NY, USA, 2016), MIG '16, ACM, pp. 55–60.

[129] Müller, M., Heidelberger, B., Hennix, M., and Ratcliff, J. Position based dynamics. *J. Vis. Comun. Image Represent. 18*, 2 (Apr. 2007), 109–118.

[130] Na, K., and Jung, M. Hierarchical retargetting of fine facial motions. *Computer Graphics Forum 23*, 3 (2004), 687–695.

[131] Nazari, M. A., Perrier, P., and Payan, Y. The distributed lambda (λ) model (DLM): A 3-D, finite-element muscle model based on feldman's λ model; assessment of orofacial gestures. *Journal of speech, language, and hearing research 56*, 6 (2013), S1909–S1923.

[132] Neumann, T., Varanasi, K., Wenger, S., Wacker, M., Magnor, M., and Theobalt, C. Sparse localized deformation components. *ACM Trans. Graph. 32*, 6 (2013), 179:1–179:10.

[133] Noh, J.-y., Fidaleo, D., and Neumann, U. Animated deformations with radial basis functions. In *Proceedings of the ACM symposium on Virtual reality software and technology* (New York, NY, USA, 2000), VRST '00, ACM, pp. 166–174.

[134] Noh, J.-y., and Neumann, U. Expression cloning. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), SIGGRAPH '01, ACM, pp. 277–288.

[135] Olszewski, K., Lim, J. J., Saito, S., and Li, H. High-fidelity facial and speech animation for vr hmds. *ACM Trans. Graph. 35*, 6 (Nov. 2016), 221:1–221:14.

[136] Orvalho, V. C., Zacur, E., and Susin, A. Transferring the rig and animations from a character to different face models. *Computer Graphics Forum 27*, 8 (2008), 1997–2012.

[137] Osipa, J. *Stop Staring: Facial Modeling and Animation Done Right*, 3rd ed. SYBEX Inc., Alameda, CA, USA, 2010.

[138] Papadopoulo, T., and Lourakis, M. I. Estimating the jacobian of the singular value decomposition: Theory and applications. In *European Conference on Computer Vision* (2000), Springer, pp. 554–570.

[139] Parke, F. I. Computer generated animation of faces. In *Proceedings of the ACM Annual Conference - Volume 1* (New York, NY, USA, 1972), ACM '72, ACM, pp. 451–457.

[140] Pighin, F., Hecker, J., Lischinski, D., Szeliski, R., and Salesin, D. H. Synthesizing realistic facial expressions from photographs. In *SIGGRAPH '98* (New York, NY, USA, 1998), SIGGRAPH '98, ACM, pp. 75–84.

[141] Pighin, F., Szeliski, R., and Salesin, D. Resynthesizing facial animation through 3d model-based tracking. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on* (1999), vol. 1, pp. 143–150 vol.1.

[142] Pitermann, M., and Munhall, K. G. An inverse dynamics approach to face animation. *The Journal of the Acoustical Society of America 110*, 3 (2001), 1570–1580.

[143] Platt, S. M., and 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.

[144] Pyun, H., Kim, Y., Chae, W., Kang, H. W., and Shin, S. Y. An example-based approach for facial expression cloning. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2003), SCA '03, Eurographics Association, pp. 167–176.

[145] Ravikumar, S., Davidson, C., Kit, D., Campbell, N., Benedetti, L., and Cosker, D. Reading between the dots: Combining 3d markers and facs classification for high-quality blendshape facial animation. In *Proceedings of Graphics Interface 2016* (2016), GI 2016, Canadian Human-Computer Communications Society / Société canadienne du dialogue humain-machine, pp. 143–151.

[146] Rhee, T., Hwang, Y., Kim, J. D., and Kim, C. Real-time facial animation from live video tracking. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2011), SCA '11, ACM, pp. 215–224.

[147] Rivers, A. R., and James, D. L. Fastlsm: Fast lattice shape matching for robust real-time deformation. *ACM Trans. Graph. 26*, 3 (July 2007).

[148] Sagar, M. Facial performance capture and expressive translation for king kong. In *ACM SIGGRAPH 2006 Sketches* (New York, NY, USA, 2006), SIGGRAPH '06, ACM.

[149] Saito, J. Smooth contact-aware facial blendshapes transfer. In *Proceedings of the Symposium on Digital Production* (New York, NY, USA, 2013), DigiPro '13, ACM, pp. 7–12.

[150] Saito, S., Li, T., and Li, H. Real-time facial segmentation and performance capture from rgb input. In *European Conference on Computer Vision* (2016), Springer, pp. 244–261.

[151] Saragih, J. M., Lucey, S., and Cohn, J. Real-time avatar animation from a single image. In *Automatic Face Gesture Recognition and Workshops (FG 2011), 2011 IEEE International Conference on* (2011), pp. 117–124.

[152] Seo, J., Irving, G., Lewis, J. P., and Noh, J. Compression and direct manipulation of complex blendshape models. In *Proceedings of the 2011 SIGGRAPH Asia Conference* (New York, NY, USA, 2011), SA '11, ACM, pp. 164:1–164:10.

[153] Seol, Y., Lewis, J., Seo, J., Choi, B., Anjyo, K., and Noh, J. Spacetime expression cloning for blendshapes. *ACM Trans. Graph. 31*, 2 (2012), 14:1–14:12.

[154] Seol, Y., and Lewis, J. P. Tuning facial animation in a mocap pipeline. In *ACM SIGGRAPH 2014 Talks* (New York, NY, USA, 2014), SIGGRAPH '14, ACM, pp. 13:1–13:1.

[155] Seol, Y., Seo, J., Kim, P. H., Lewis, J. P., and Noh, J. Artist friendly facial animation retargeting. In *Proceedings of the 2011 SIGGRAPH Asia Conference* (New York, NY, USA, 2011), SA '11, ACM, pp. 162:1–162:10.

[156] Seol, Y., Seo, J., Kim, P. H., Lewis, J. P., and Noh, J. Weighted pose space editing for facial animation. *Vis. Comput. 28*, 3 (2012), 319–327.

[157] Sifakis, E., and Barbič, J. FEM simulation of 3d deformable solids: A practitioner's guide to theory, discretization and model reduction. In *ACM SIGGRAPH 2012 Courses* (New York, NY, USA, 2012), SIGGRAPH '12, ACM, pp. 20:1–20:50.

[158] Sifakis, E., Neverov, I., and Fedkiw, R. Automatic determination of facial muscle activations from sparse motion capture marker data. In *ACM SIGGRAPH 2005 Papers* (New York, NY, USA, 2005), SIGGRAPH '05, ACM, pp. 417–425.

[159] Sifakis, E., Selle, A., Robinson-Mosher, A., and Fedkiw, R. Simulating speech with a physics-based facial muscle model. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2006), SCA '06, Eurographics Association, pp. 261–270.

[160] Sobotta, J. *Sobotta's Atlas and Text-book of Human Anatomy.* 1909.

[161] Stoiber, N., Aubault, O., Seguier, R., and Breton, G. The mimic game: Real-time recognition and imitation of emotional facial expressions. In *ACM SIGGRAPH 2010 Talks* (New York, NY, USA, 2010), SIGGRAPH '10, ACM, pp. 38:1–38:1.

[162] Stoiber, N., Seguier, R., and Breton, G. Automatic design of a control interface for a synthetic face. In *Proceedings of the 14th International Conference on Intelligent User Interfaces* (New York, NY, USA, 2009), IUI '09, ACM, pp. 207–216.

[163] Sumner, R. W., and Popović, J. Deformation transfer for triangle meshes. *ACM Trans. Graph. 23*, 3 (2004), 399–405.

[164] Tena, J. R., De la Torre, F., and Matthews, I. Interactive region-based linear 3d face models. In *ACM SIGGRAPH 2011 papers* (New York, NY, USA, 2011), SIGGRAPH '11, ACM, pp. 76:1–76:10.

[165] Terzopoulos, D., and Waters, K. Analysis and synthesis of facial image sequences using physical and anatomical models. *IEEE Trans. Pattern Anal. Mach. Intell. 15*, 6 (1993), 569–579.

[166] Teschner, M., Girod, S., and Girod, B. Direct computation of nonlinear soft-tissue deformation.

[167] Teschner, M., Heidelberger, B., Mueller, M., Pomeranets, D., and Gross, M. Optimized spatial hashing for collision detection of deformable objects. *Proceedings of Vision, Modeling, Visualization (VMV 2003)* (2003), 47–54.

[168] Thies, J., Zollhöfer, M., Niessner, M., Valgaerts, L., Stamminger, M., and Theobalt, C. Real-time expression transfer for facial reenactment. *ACM Trans. Graph. 34*, 6 (Oct. 2015), 183:1–183:14.

[169] Thies, J., Zollhöfer, M., Stamminger, M., Theobalt, C., and Niessner, M. Face2face: Real-time face capture and reenactment of RGB videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 2387–2395.

[170] Thies, J., Zollöfer, M., Stamminger, M., Theobalt, C., and Niessner, M. FaceVR: Real-time facial reenactment and eye gaze control in virtual reality. *arXiv preprint arXiv:1610.03151* (2016).

[171] Vlasic, D., Brand, M., Pfister, H., and Popović, J. Face transfer with multilinear models. In *ACM SIGGRAPH 2005 Papers* (New York, NY, USA, 2005), SIGGRAPH '05, ACM, pp. 426–433.

[172] von der Pahlen, J., Jimenez, J., Danvoye, E., Debevec, P., Fyffe, G., and Alexander, O. Digital ira and beyond: Creating real-time photoreal digital actors. In *ACM SIGGRAPH 2014 Courses* (New York, NY, USA, 2014), SIGGRAPH '14, ACM, pp. 1:1–1:384.

[173] Wang, C., Shi, F., Xia, S., and Chai, J. Realtime 3d eye gaze animation using a single RGB camera. *ACM Trans. Graph. 35*, 4 (July 2016), 118:1–118:14.

[174] 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.

[175] Wang, H., and Ahuja, N. Facial expression decomposition. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on* (2003), pp. 958–965 vol.2.

[176] Wang, H., and Yang, Y. Descent methods for elastic body simulation on the gpu. *ACM Trans. Graph. 35*, 6 (Nov. 2016), 212:1–212:10.

[177] Wang, Y., Huang, X., Lee, C.-S., Zhang, S., Li, Z., Samaras, D., Metaxas, D., Elgammal, A., and Huang, P. High resolution acquisition, learning and transfer of dynamic 3-d facial expressions. *Computer Graphics Forum 23*, 3 (2004), 677–686.

[178] 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.

[179] Waters, K., and Terzopoulos, D. The computer synthesis of expressive faces. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences 335*, 1273 (1992), 87–93.

[180] Weber, R., Barrielle, V., Soladié, C., and Séguier, R. High-level geometry-based features of video modality for emotion prediction. In *Proceedings of the 6th International Workshop on Audio/Visual Emotion Challenge* (New York, NY, USA, 2016), AVEC '16, ACM, pp. 51–58.

[181] Weise, T., Bouaziz, S., Li, H., and Pauly, M. Realtime performance-based facial animation. *ACM Trans. Graph. 30*, 4 (July 2011), 77:1–77:10.

[182] Weise, T., Li, H., Van Gool, L., and Pauly, M. Face/off: Live facial puppetry. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2009), SCA '09, ACM, pp. 7–16.

[183] Weiss, J. A., Maker, B. N., and Govindjee, S. Finite element implementation of incompressible, transversely isotropic hyperelasticity. *Computer methods in applied mechanics and engineering 135*, 1 (1996), 107–128.

[184] Weng, Y., Cao, C., Hou, Q., and Zhou, K. Real-time facial animation on mobile devices. *Graphical Models 76*, 3 (2014), 172–179.

[185] Williams, L. Performance-driven facial animation. In *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1990), SIGGRAPH '90, ACM, pp. 235–242.

[186] Xu, F., Chai, J., Liu, Y., and Tong, X. Controllable high-fidelity facial performance transfer. *ACM Trans. Graph. 33*, 4 (July 2014), 42:1–42:11.

[187] Yang, F., Wang, J., Shechtman, E., Bourdev, L., and Metaxas, D. Expression flow for 3d-aware face component transfer. In *ACM SIGGRAPH 2011 Papers* (New York, NY, USA, 2011), SIGGRAPH '11, ACM, pp. 60:1–60:10.

[188] Yu, J., Jiang, C., Li, R., Luo, C. W., and Wang, Z. F. Real-time 3d facial animation: From appearance to internal articulators. *IEEE Transactions on Circuits and Systems for Video Technology* (2016).

[189] Zeng, M., Liang, L., Liu, X., and Bao, H. Video-driven state-aware facial animation. *Comput. Animat. Virtual Worlds 23*, 3-4 (2012), 167–178.

[190] Zhang, L., Snavely, N., Curless, B., and Seitz, S. M. Spacetime faces: High resolution capture for modeling and animation. In *ACM SIGGRAPH 2004 Papers* (New York, NY, USA, 2004), SIGGRAPH '04, ACM, pp. 548–558.

[191] Zhang, Q., Liu, Z., Guo, B., and Shum, H. Geometry-driven photorealistic facial expression synthesis. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2003), SCA '03, Eurographics Association, pp. 177–186.

[192] Zhang, Q., Liu, Z., Quo, G., Terzopoulos, D., and Shum, H.-Y. Geometry-driven photorealistic facial expression synthesis. *Visualization and Computer Graphics, IEEE Transactions on 12*, 1 (2006), 48–60.

[193] Zhang, W., Wang, Q., and Tang, X. Performance driven face animation via non-rigid 3d tracking. In *Proceedings of the 17th ACM International Conference on Multimedia* (New York, NY, USA, 2009), MM '09, ACM, pp. 1027–1028.

Abstract

Generating convincing synthetic facial animation is a crucial step in the creation of content for a wide variety of media: movies, video games, virtual reality, and video-conferencing. However, producing convincing results is challenging, since humans are experts in analyzing facial expressions and will hence detect any artifact. The dominant paradigm for the production of high-quality facial animation is the blendshapes paradigm, where facial expressions are decomposed as a linear combination of more basic expressions. However, this technique requires large amounts of work to reach the desired quality, and is thus only capable of producing high-quality animation for large budget movies. Producing high-quality facial animation is possible using physical simulation, but this requires the costly acquisition of medical imaging data, which reduces considerably the availability of the technique. We propose to merge the blendshapes paradigm and the physical simulation paradigm, to build upon the ubiquity of blendshapes while benefiting from physical simulation for complex effects. We therefore introduce *blendforces*, a paradigm where blendshapes are interpreted as a basis for approximating the forces emanating from the facial muscles. We show that, combined with an appropriate face physical system, these blendforces can be used to produce convincing facial animation, with natural skin dynamics, handling of lips contacts, sticky lips, inertial effects and handling of gravity. We encompass this framework within a practical realtime performance capture setup, where we produce realtime facial animation with physical effects from a simple RGB camera feed. To the best of our knowledge, this constitutes the first instance of realtime physical simulation applied to the challenging task of facial animation.

Résumé

La génération d'animation faciale de synthèse convaincante constitue une étape cruciale pour la création de contenus dédiés à une grande variété de média : films, jeux vidéo, réalité virtuelle, et vidéoconférence. Cependant, chaque personne étant un expert dans l'analyse des expressions faciales, le moindre défaut se remarque aisément, et il est donc ardu de produire des animations convaincantes. Le paradigme dominant pour la création d'animations faciales de haute-qualité est la méthode des blendshapes, où les expressions faciales sont décomposées comme la combinaison linéaire d'expressions plus basiques. Toutefois, cette technique requiert une grande quantité de travail manuel pour produire la qualité requise, et ne peut donc produire des animations de grande qualité que pour les films à grand budget. La production d'animations faciales de haute qualité est possible à l'aide de la simulation physique, mais ce processus requiert l'acquisition coûteuse de données obtenues à partir de scanners médicaux, ce qui réduit considérablement son champ d'application. Nous proposons de réunir le paradigme des blendshapes et celui de la simulation physique, afin de profiter de l'ubiquité des blendshapes tout en bénéficiant de la simulation physique pour produire des effets complexes. Nous introduisons donc *blendforces*, un paradigme dans lequel les blendshapes sont interprétées comme une base pour approximer les forces issues des muscles faciaux. Nous montrons que, combinées à un système physique approprié, ces blendforces peuvent être utilisées pour produire des animations faciales convaincantes, présentant une dynamique de mouvements de peau réaliste, gérant les contacts entre les lèvres, simulant le collage des lèvres, et intégrant les effets des forces inertielles et de la gravité. Nous intégrons ce formalisme dans une application pratique : la production en temps réel d'animation faciale avec effets physiques basée sur des mouvements capturés à l'aide d'une simple caméra RGB. À notre connaissance, il s'agit de la première démonstration de simulation physique temps-réel appliquée au cas délicat de la simulation d'animations faciales.