Qualifying Oral Position Paper

by

Rinat Abdrashitov

Contents

Chapter 1

Introduction

Creating an animation of a realistic and expressive human face or fantasy creatures is one of greatest challenges in computer graphics. The face is an extremely complex biomechanical system that is very difficult to model because it can convey subtle emotions through tiny motions. The entertainment industry is the main driver for the development of advanced computer facial animation systems. In films and videogames, the face and the facial expressions become fundamental to convey emotions of a character. At core of character animation is believability. Animated characters like Gollum have been convincing enough to carry important scenes in close up within a live action film. As animated characters approach high level of realism in terms of their visual and motion details there comes a point when our perception of a character identifies that something isn't natural. The character appears too plastic, the motion lacks fluidity, or the lip-synchronization looks strange. At that point, we become aware that the character is not real. This phenomenon is known as the uncanny valley.

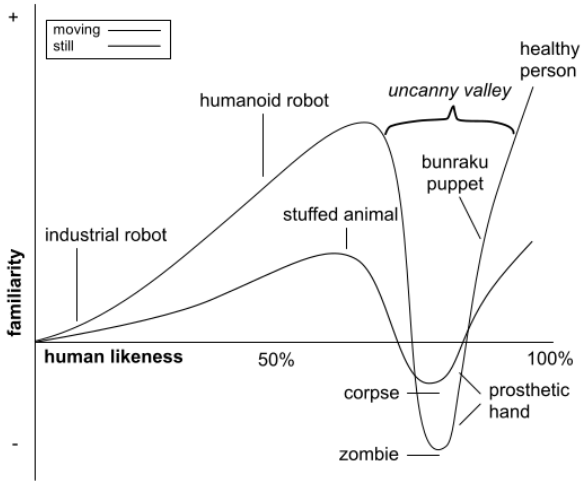

Figure 1.1: The uncanny valley.

In 1970, the Japanese roboticist Masahiro Mori coined the term the uncanny valley as a concept of robotics [Mori, 1970]. His hypothesis states that as a robot is made increasingly human-like in appearance and motion, our human response is increasingly empathetic until a point when there is a reversal and our response becomes strongly repulsive (see Figure 2.2).

Current workflow for producing facial animations can be represented as follows:

- 2D concept sketch

- Modelling (geometry)
- Rigging (control parameters)
- Animation (perfomance)
- Rendering (skin and hair shading)

In this document we look at the stages involved in producing facial animation (modelling, rigging, animation), ignoring rendering issues and concept development.

Chapter 2

Digital Facial model

2.1 Facial Expression Theory and Coding System

The psychologist with the greatest influence on expression in facial animation is Paul Ekman. The majority of papers on the subject mention his Facial Action Coding System (FACS) system or the six basic emotions: Anger, Disgust, Fear, Joy, Sadness and Surprise. The first version of FACS was described in 1976 [Ekman and Friesen, 1976], then published in 1978 [Friesen and Ekman, 1978], and revised in 2002 [Ekman, 2002]. The FACS system defines Action Units in several varieties: Single Action Units, Action Descriptors and Gross Behaviour Codes which are muscle actions that in combination can describe any visually distinguishable facial position a human can voluntarily make. In this text we concentrate on the Single Action Units and refer to them as AUs. Each AU represents an individual muscle action, or an action of a small group of muscles, a single recognizable facial expression. Example action units are the inner brow raiser, the outer brow raiser, and the lid tightener. According to Ekman, "FACS allows the description of all facial behavior we have observed, and every facial action we have attempted." There are a total of 46 action units as listed in Figure 2.1. Theoretically, it is possible for as many as 20 to combine in a single facial movement. A facial movement also may involve only one action unit. Not all action units can be combined, since some involve opposite actions. Also, some of the actions can conceal the presence of others.

The use of FACS is extensive in computer facial animation. This use exceeds Ekman's intended use which was to describe and score all voluntary, distinguishable, momentary facial positions for psychological study. Thus its use in computer facial animation has some limitations:

- FACS offers spatial motion descriptions but not temporal components, so co-articulation effects are lost [Deng and Noh, 2008].
- the jaw and lips have many positions that are not specified by FACS [Parke and Waters, 2008].

2.2 Parametarization

2.2.1 Blendshape parameterization

Blendshape facial animation is the most common choice for realistic or humanoid characters in the entertainment industry. The approach has been used for lead characters in movies such as The Curious

AU	FACS Name	Muscular Basis
1	inner-brow raiser	*frontalis, pars medialis*
2	outer-brow raiser	*frontalis, pars lateralis*
4	brow raiser	*depressor glabellae, depressor supercilli, corrugator*
5	upper-lid raiser	*lavator palpebrae superioris*
6	cheek raiser	*orbicularis oculi, pars orbitalis*
7	lid tightener	*orbicularis oculi, pars palebralis*
8	lips together	*orbicularis oris*
9	nose wrinkler	*levator labii superioris, alaeque nasi*
10	upper-lip raiser	*levator labii superioris, caput infraorbitalis*
11	nasolabial-furrow deepener	*zygomatic minor*
12	lip corner puller	*zygomatic major*
13	cheek puffer	*caninus*
14	dimpler	*buccinnator*
15	lip-corner depressor	*triangularis*
16	lower-lip depressor	*depressor labii*
17	chin raiser	*mentalis*
18	lip puckerer	*incisivii labii superioris, incisivii labii inferiouis*
20	lip stretcher	*risorius*
22	lip funneler	*orbicularis oris*
23	lip tightener	*orbicularis oris*
24	lip pressor	*orbicularis oris*
25	parting of lips	*depressor labii*, or relaxation of *mentalis* or *orbicularis oris*
26	jaw drop	*masetter*; relaxed *temporal* and internal *pterygoid*
27	mouth stretch	*pterygoids; digastric*
28	lip suck	*orbicularis oris*
38	nostril dilator	*nasalis, pars alaris*
39	nostril compressor	*nasalis, pars transversa* and *depressor septi nasi*
41	lid droop	relaxation of *levator palpebrae superioris*
42	eyelid slit	*orbicularis oculi*
43	eyes closed	relaxation of *levator palpebrae superioris*
44	squint	*orbicularis oculi, pars palpebralis*
45	blink	relax *levator palpebrae* and then contract *orbicularis oculi, pars palpebralis*
46	wink	*orbicularis oculi*

Figure 2.1: Single Action Units.

Case of Benjamin Button, King Kong, The Lord of the Rings and Avatar. A unique facial pose is generated as a linear combination of a number of facial expressions, the blendshape "targets". By varying the weights of the linear combination, a range of facial expressions can be expressed with simple computation. Ekman's FACS system is used as the guide to initial set of 46 blendshapes that are created to animate the character. The set of shapes can be extended or reduced as desired to refine the range of expressions that the character can produce. Game characters tend to have much smaller set than the characters created for movies. In comparison with other representations, blendshapes have two major advantages:

- Animators have intuitive controls over blendshapes targets and clear understanding of how weights affect resulting facial expression. Other linear models such as PCA do not provide this.
- Blendshapes force the animator to stay within the facial expression space, that is, arbitrary deformations are not possible. While this could be taken as a limitation, it helps ensure that the facial character is consistent even if animated by different individuals. It also allows for a division of responsibility between the character modeler and animator.

Even though the blendshape technique is conceptually simple, developing a blendshape face model is a challenging and time consuming process at present. To express a complete range of realistic expressions, digital modellers often have to create large libraries of blendshape targets. Even with the help of skilled modellers and riggers generating a reasonably detailed model and rig can take months of work, involving many iterations of refinement.

2.2.2 PCA parameterization.

To avoid the need to manually create every blendshape, Principal component analysis (PCA) of motion capture data (dense or marker) can be used to automatically produce a blendshape model of sorts. The advantages of a PCA model are that it is the most accurate blendshape model possible with a given number of blendshapes, the blendshapes are orthogonal which makes fitting particularly efficient. However, the model is quite poorly suited for human manipulation which is a big disadvantage. This is because the individual blendshape targets resulting from PCA tend to have widespread effects that are difficult to describe and remember. Changing one weight might affect multiple areas of the face that are otherwise isolated in a regular blendshape model. In a production environment, it is often crucial to have ability of human post-processing and editing of the animation so intuitive parameterization is necessary. As well, motion capture data is often not available by definition.

Figure 2.2: Example of blendshape derived from PCA analysis of the motion capture data. Its difficult to interpret and remember.

2.3 Mathematical definition of blendshapes

Consider a facial model composed of n blendshapes and p verticies. We represent the face model as a column vector $\mathbf{f}$ containing all the model vertex coordinates in some arbitrary but fixed order, such as xyzxyzxyz [Lewis et al., 2014]. Similarly, we denote the blendshape targets as vectors $\mathbf{b}_k$ so the blendshape model is

$$\mathbf{f} = \sum_{k=0}^{n} w_k \mathbf{b}_k \tag{2.1}$$

or using matrix notation

$$\mathbf{f} = \mathbf{B}\mathbf{w} \tag{2.2}$$

This formulation is called whole-face blendshape formulation because each column of B represents an entire facial expression, such as a smile. Another formulation is delta blendshape formulation, where the targets $\mathbf{b}_k$ are offsets from a neutral face with many zero entries. If $\mathbf{b}_0$ is a neuteral face then

$$\mathbf{f} = \mathbf{B}\mathbf{w} + \mathbf{b}_0 \tag{2.3}$$

In both cases, the weights are usually nonnegative and bounded to [0,1] range, although allowing mildly negative weights or weights greater than 1 is occasionally useful. The wholeface approach makes it easier to reproduce the chosen expressions and can be efficient for modeling speech if the basis includes a set of visemes. On the other hand, this approach makes it diffcult to perform local adjustments. Current industry practice most often often uses delta blendshapes modeled on facial muscles. Popular animation software Maya [Derakhshani, 2012] uses delta formulation.

As an individual weight in Eq. 2.3 varies from zero to one, the moving vertices on the face travel along a line. However, facial movement is inherently non-linear. To solve this problem and allow for more fidelity, production blendshape implementations such as that in Maya allow targets to be situated at intermediate weight values, giving piecewise linear interpolation.

Eq. 2.3 encodes facial expression but doesnt provide information about the head pose. Head pose estimation is often required in facial tracking. To fix that, rotation matrix $\mathbf{R}$ and translation vector $\mathbf{t}$ are added to arrive at the equation

$$\mathbf{f} = \mathbf{R}(\mathbf{B}\mathbf{w} + \mathbf{b}_0) + \mathbf{t} \tag{2.4}$$

Chapter 3

Facial modeling and rigging

3.1 Modelling

There are several approaches for creating blendshapes. A skilled digital artist can deform a base mesh into the different shapes needed to cover the desired range of expressions. Alternatively, the blendshapes can be directly scanned from a real actor or a sculpted model. A common template model can be registered to each scan in order to obtain vertex-wise correspondences across the blendshape targets. Below we outline one of the methods to register a generic template to scans [Achenbach et al., 2015]. Other methods include [Allen et al., 2003][Amberg et al., 2007][Schneider and Eisert, 2009][Cao et al., 2014].

First, high-resolution DSLR camera images are used to reconstruct a 3D point cloud of the face doing different expressions. Data includes point position, normal vectors and RGB colors. Generic template model can be obtained from the FaceWarehouse database [Cao et al., 2014] Inital alignment is perfomed by fitting a template to facial landmarks (automatically detected from images) using Blanz and Vetter's morphable model [Blanz and Vetter, 1999]. After initialization, the deformable registration updates the template vertex positions X, such that the template model better fits the scanner points P. This is achieved by minimizing an objective function that consists of a fitting and a regularization term:

$$E(X) = E_{fit}(X, P) + \lambda E_{reg}(X, X^{'}) \tag{3.1}$$

The fitting energy E_{fit} penalizes the distance between the template X and the point-cloud P, and the regularization energy E_{reg} penalizes the distortion from the undeformed state $X^{'}$ to the deformed state X. Non-rigid ICP is used alternatingly to compute correspondences and minimize the enegery, starting with a rather stiff surface $\lambda = 1$ that is subsequently softened until $\lambda = 10^{-7}$ to allow for more and more accurate fits.

In more detail, E_{fit} penalizes squared deviation between points on the template and scanned model in a point-to-point or point-to-plane manner, or a linear combination thereof. E_{reg} is responsible for ensuring physical validity of the deformed model by penalizing unwanted types of deformations, typically by trying to keep the surface locally rigid. Their regularization energy minimizes a discrete bending model by penalizing the Laplacian of the deformation.

3.2 Rigging

A face model can be completely rigged using only blendshapes. In fact in Maya it only takes few clicks to setup a rig using only blendshapes. However, a large number of them is necessary to provide control over every region of the face and every subtle movement. For example facial animation of Gollum, in the film The Lord of the Rings: The Two Towers, requires 675 blendshapes. Despite that, blendshapes is still a common technique used by artists in production [Osipa, 2010].

However, the jaw and neck are sometimes handled by alternate approaches. For example, since the motion of the jaw has a clear rotational component, the jaw-open target is often augmented by linear blend skinning. The eyelids are another area that is sometimes handled by alternate rigging approaches, again due to the rotational motion [Orvalho et al., 2012]. So its common to see hybrid rigs (blendshapes + LBS).

Example-Based Facial Rigging

[Li et al., 2010] addresses the problem of large number of blendshapes that need to be created to achieve high quality facial animation. Their goal is to produce a full set of blendshapes from a user provided handcrafted character or scanned 3D model in neutral expression. They automatically create optimal blendshapes from a set of example(training) poses and generic template blendshapes. Generic blendshape model is given as a set of meshes $\mathbf{A} = \mathbf{a}_0, ..., \mathbf{a}_n$, where $\mathbf{a}_0$ is the neuteral pose and the $\mathbf{a}_i$, $i > 0$ are delta blendshapes. They deform the generic neuteral face $\mathbf{a}_0$ to the example poses using the non-rigid registration. This produces a set $\mathbf{S} = \mathbf{s}_1, ..., \mathbf{s}_m$ of complete meshes with connectivity of the prior model and shape of the example poses. Their goal is to compute a new blendshape model $\mathbf{B} = \mathbf{b}_0, ..., \mathbf{b}_n$ that matches the geometry and motion of the target actor. In other words they want to find blendshapes $\mathbf{B}$ and corresponding weights $\mathbf{w}_j$ for each $s_j \in \mathbf{S}$ such that the training poses are faithfully reproduced, i.e., $\mathbf{s}_j = \mathbf{B}\mathbf{w}_j + \mathbf{b}_0$. This bi-linear problem is solved iteratively by alternating between two steps: keeping the blending weights $\mathbf{w}_j$ fixed and optimizing for the blendshapes, keeping blendshapes fixed and solving for the optimal weights. To approximate weights $\mathbf{w}_j$ that provide initial values for the first step of the optimization user selects appropriate blending weights on the template to model a pose similar to the training pose $\mathbf{s}_j$. Those weights dont need to be very accurate and typically not more than 4 poses is required.

Chapter 4

Animation and Interaction techniques

4.1 Keyframe animation

Blendshape models have traditionally been animated using keyframe animation of the weights. Commercial packages such as Maya provide spline interpolation of the weights and allow the tangents to be set at keyframes. Professional animation requires a keyframe roughly every three frames. This is an extremely laborious process that requires years to master.

4.2 Direct manipulation

Animators must memorize up tp 100 commonly used sliders to efficient at using a blendshape model. Even quickly locating a desired slider isn't immediate. Professional 3D animation requires about one hour of labor to produce one second of animation. To accelerate this work [Lewis and Anjyo, 2010] proposed a direct-manipulation interface of blendshape rigs that is similar in its idea to inverse kinematics for body animation.

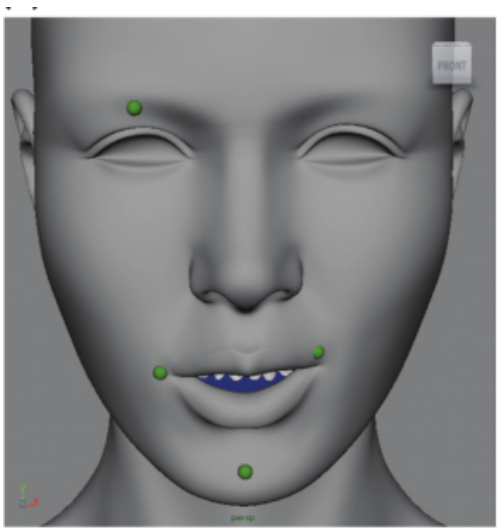

Figure 4.1: Using direct manipulators(green spheres) to pose the face.

Their user interface consistes of manipulators (represented as a spheres in the UI Fig. 4.1) indicating the desired location of a particular corresponding vertex. The selected manipulator (the active manipulator) is the one being moved; the other manipulators serve as constraints. The animator simply selects manipulators and drags them to desired locations and the system must determine the other vertices' positions. Their solution is based on the idea of finding the defrormation that causes the smallest change in blendshape weights.

They solve an optimization problem with the following objective

$$E(\mathbf{w}) = ||\mathbf{B}\mathbf{w} - \mathbf{m}||^2 + \alpha||\mathbf{w} - \mathbf{w}_0||^2 + \gamma||\mathbf{w}||^2 \tag{4.1}$$

where $\mathbf{m}$ is the manipulator location, $\mathbf{B}$ is the blendshape matrix (sliced to only have rows corresponding to $\mathbf{m}$), $\mathbf{w}_0$ is the previous weight vector. First term pushes the face towards manipulators, second term ensures that blendshape weights dont change too much and last term is weight decay regularization term that biases the results away from extreme poses.

4.3 Perfomance-driven animation

Lance William's paper "Performance-Driven Facial Animation" [Williams, 1990] introduced the term to the computer graphics community. In performance-driven facial animation, the motion of a human actor is used to drive the target face model. While keyframe animation is mostly used in films with stylized characters, performance-driven animation is commonly used for visual-effects movies. Because blendshapes are the common choice for realistic facial models, blendshapes and performance-driven animation are frequently used together. Main advantages of performance-driven animations compared to keyframe animaiton are:

- Much easier to produce a facial expression than to adjust multiple sliders.
- Keyframe animation cannot consistently capture the subtleties of human motion.
- Getting consistent results is time consuming and expensive.

The literature on face tracking is extensive so we will concentrate on performance capture methods that drive a blendshape rig. Techniques that drive a mesh or a skeleton-based rig with linear blend skinning will not be covered. We separate methods by their input: methods that take 3D position of face markers, 2D RGB images from simple consumer cameras, depth maps from RGBD cameras.

4.3.1 3D motion capture

Overview

3D motion capture assumes that the input is relative positions of reference points (3D markers) on the actor's face captured using multi-camera rigs or laser marker system.

For a particular motion capture frame f, the actor's performance is given as a 3M-dimensional vector of M stacked marker positions $\mathbf{a}_f = (\mathbf{a}_f^1, ..., \mathbf{a}_f^M)^T$. The goal of any blendshape retargeting system is to compute the time-varying weights that reproduce the facial expressions on the target face rig for a given actor's performance capture. This requires a set of personalized sparse actor blendshapes $\mathbf{p}_k = (p_k^1, ..., p_k^M)^T$ where $\mathbf{p}_k$ is a subset of verticies in $\mathbf{b}_k$ (full geometry of a blendshape). For each captured frame f, the blendshape weights $\mathbf{w}$ can be computed by minimizing the squared

Figure 4.2: Mocap of actress Ellen Page.

distance between the markers $\mathbf{a}_f$ and a weighted combination of the actor's sparse blendshapes $\mathbf{p}_k$:

$$E_{fit}(\mathbf{w}) = ||\mathbf{a}_f - \sum_{k=1}^{K} w_k \mathbf{p}_k||^2 \tag{4.2}$$

In order to resolve ambiguities, prevent overfitting, or penalize artifacts, the blendshape fitting process is typically regularized through additional energy term $E_{reg}(\mathbf{w})$. Typical choices for the energy $E_{reg}(\mathbf{w})$ are (weighted combinations of) L2 regularization $||\mathbf{w}||^2$ to penalize large weights [Lewis and Anjyo, 2010], L1 regularization $||\mathbf{w}||_1$ for inducing sparsity [Bouaziz et al., 2013], and penalization of temporal changes $||\mathbf{w}_{f-1} - \mathbf{w}_f||^2$ to remove jitter [Lewis and Anjyo, 2010]. Finally typical retargetting problem can be casted as a minimization problem of the enegry:

$$E_{retarget}(\mathbf{w}) = E_{fit}(\mathbf{w}) + E_{reg}(\mathbf{w}) \tag{4.3}$$

If we have a set of blendshapes for the target character that correspond to the same expressions of the actor's blendshapes, then we can simply reuse the same weights $\mathbf{w}$. However there are few problems with this approach:

- We assume that source blendshape are available. Unfortunately, this rarely occurs in practice. Often the blendshapes are not available for the source or they do not correspond to the target blendshapes. Although a custom model representing the source performance could be manually constructed, this can take many months of manual effort to do well.
- Simple linear mapping of weights doesn't always produce desired effects for characters with different facial proportions (i.e., stylized cartoon models)

Below we describe techniquess that make the manual construction of a source blendshape model unnecessary and could provide decent results for retargeting to stylized characters.

Animating Blendshape Faces by Cross-Mapping Motion Capture Data

Method of [Deng et al., 2006] doesnt require source blendshape rig in order to transfer its motion to new 3D faces. They present a semi-automatic technique to directly animate target blendshape face models by mapping facial motion capture data spaces to 3D blendshape face spaces.

They start by capturing small corpus of motion capture data consisting of actor speaking in a normal speed with full intensity of emotions (neutral, happiness, anger, and sadness). At the same time, they use another video camera to record the front of the actors face and audio. Using speech recognition software they perform phoneme-alignment on accompanying audio in order to align each phoneme with its corresponding motion capture segments.

Based on the above phoneme-alignment results, they manually select one reference motion capture frame C_f from a phoneme-associated motion capture segment, for each possible combination of visemes and expressions. Furthermore, they select addtional frames for extreme expressive poses. These selected motion capture frames C_f with their aligned video frames V_f are referred as reference mocap-video pairs, $\{C_f, V_f\}$. The motions of all markers in one frame are packed into a motion vector and PCA is applied onto all the motion vectors to reduce dimensionality using the first 4 principal components. As a result reduced space coordinates CF_f are obtained. Given a reference face image V_f, they manually tune blendshape weights $\mathbf{w}_f$ on the target rig to perceptually match the animated faces and the reference face images. Now every $\{C_f, V_f\}$ pair has a corresponding reference mocap-weight pairs $\{CF_f, w_f\}$.

They use scattered data interpolation algorithm(RBF) to describe the calculation of blendshape weights for an input motion capture frame:

$$F(\mathbf{x}) = \sum_{k=1}^{N} w_{rbf}\phi_k(\mathbf{x}) \tag{4.4}$$

Where $F(\mathbf{x})$ maps PCA coordinates of mocap landmarks to blendshape weights. Gaussian function is used as the basis function, $\phi_k(\mathbf{x}) = exp(\frac{-||x-CF_k||^2}{2\sigma_k^2})$, N is the number of the training set.The above regression function output $F(\mathbf{x})$ is created for each blendshape weight. We need to learn w_{rbf} from training set $\{CF_f, w_f\}$.

The above regression function output $F(\mathbf{x})$ is created for each blendshape weight. Taking the regression function, $F_j(\mathbf{x})$, for the jth blendshape weight, we want to minimize the following cost function to learn w_{rbf} weights.

$$C(\mathbf{x}) = \sum_{k=1}^{N}(w_k^j - F_j(CF_k))^2 + \lambda\sum_{k=1}^{N} w_{rbf}^2 \tag{4.5}$$

Now given any new motion capture frame MC_f for frame f the blendshape weights for the target rig are calculated by: 1. projecting MC_f into PCA space ($\mathbf{x}_f = \text{PCA}(C_f)$) 2. calculating each jth target blendshape weight $w_k^j = F_j(\mathbf{x}_f)$.

Facial Retargeting with Automatic Range of Motion Alignment

Recent paper by [Ribera et al., 2017] automatically finds source blendshapes and unlike [Deng et al., 2006] doesn't need manually created examples of matching expressions of source and target character. Their method works well when characters have different facial proportions and can automatically retarget facial animations from a real actor to stylized characters.

They utilize a small training data set $\mathbf{a}_1, ..., \mathbf{a}_n$ (3D landmark data) to automatically find an initial guess for source blendshapes based on a similarity between target blendshapes and individual frames. Given an individual frame $\mathbf{a}_f$ and $\mathbf{s}_k = \{s_k^1, .., s_k^m\}$ which is the sparse representation of target blendshape (same as $\mathbf{p}_k$ but for target shapes see Figure.4.3) the similarity between $\mathbf{a}_i$ and $\mathbf{s}_k$ is defined as:

$$c_{k,f} = \frac{\mathbf{a}_f \cdot \mathbf{s}_k}{||\mathbf{a}_f||||\mathbf{s}_k||} \tag{4.6}$$

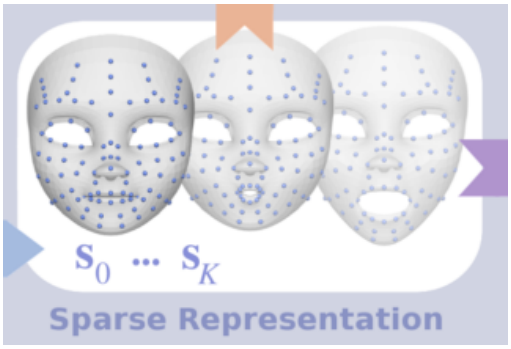

Figure 4.3: Sparse markers representing target blendshapes.

This simple dot-product formulation of Equation 4.6 is effective, because:

- Displacements of blendshapes and actor's expressions in similar directions have high correlation,
- Locality of blendshapes is considered, because vertices that do not move in blendshape s_k cancel out the contributions of the corresponding expression a_f .

Using this similarity metric they can extract individual frames from the training sequence to get the best initial guess $\mathbf{g}_k$ for actor specific blendshapes ($\mathbf{p}_k$)

They further refine ($\mathbf{p}_k$) via minimzation of the following energy

$$E_{Align}(\mathbf{p}) = E_{Match}(\mathbf{p}) + \alpha E_{Mesh}(\mathbf{p}) + \beta E_{CEG}(\mathbf{p}) \tag{4.7}$$

Where E_{Match} is inspired by manifold alignment techniques and fits the personalized blendshapes $\mathbf{p}_k$ to the actor's expressions $\mathbf{a}_f$, E_{Mesh} ensures that the actor-specific blendshapes $\mathbf{p}_k$ do not deviate too much from an initial guess $\mathbf{g}_k$, E_{CEG} preserves relations between individual blendshapes.

4.3.2 Model-based tracking of video

Overview

Instead of manually placing 3d markers on actors faces and perfoming mocap camera setup there are effective methods that only require ordinary web camera. Those methods might not produce the highest quality of animation but are much easier to setup and deploy.

Vision-based Control of 3D Facial Animation

[Chai et al., 2003] notices that the control parameters derived from a 2d cameras are noisy and frequently contain errors. So instead they propose to reuse and modify existing motion capture database to synthesise new animation by mapping low-quality 2D visual signals to high-quality 3D motion data. This idea was unique at the time and havent been explored since.
The input to their system consists of:

- a single video stream recording the user's facial movement
- a preprocessed motion capture database (each frame contains 76 3D marker positions)
- a 3D source surface scanned from the motion capture subject
- and a 3D avatar surface model to be animated

Video stream is used to track the 6 DOFs of head motion (yaw, pitch, roll and 3D position) and 19 2D expression features on the face. They convert that data into 15 control parameters $\mathbf{z} = \{z^1, ..., z^{15}\}$ that correspond in a meaningful and intuitive way with the expression movement qualities they control. For example some mouth control parameters are defined as distance between the lower and upper lips, the distance between the left and right corners of the mouth and so on. Nose, eye and eyebrow controls are defined in a similar way.

Given the definition for control parameters they pre-process mocap database to extract same 15 control parameters for each frame. So each set of 76 3D marker positions $\mathbf{x}_f = \{x_f^1, ..., x_f^{76}\}$ is associated with 15 control parameters $\mathbf{z}_f$.

Retargeting is done by finding the K closest examples in the motion capture database using the control parameters from the vision-based interface and then linearly interpolates the corresponding high-quality motion examples in the database with a local regression technique. Because the mapping from the control parameters space to the motion data space is not one to one, the query control signal is based on multiple frames rather than a single frame

Main limitation of the their system is that sometimes it loses details of the lip movement. This problem arises because the motion capture database does not include sufficient samples related to speaking. Alternatively, the amount of tracking data in the vision-based interface might not be enough to capture

the subtle movement of the lips. Their claim that with a larger database and more tracking features around the mouth, this limitation might be eliminated. At the time their system did not consider speech as an input.

3D Shape Regression for Real-time Facial Animation
[Cao et al., 2013] presented a system based on 3D shape regression that uses simple web-camera as an input. Their pipeline is divided into two stages: preprocessing stage and runtime stage.
Preprocessing stage:
They begin with capturing a pre-defined sequence of setup images consisting of the user's face under a set of different facial configurations. These configurations are separated into two classes. The first is due to rigid motion, and is captured for 15 different head poses. These poses consist of head rotations over a sequence of angles with a neutral expression. The second class is non-rigid motion and consists of 15 different expressions each at three yaw angles. Finally large expressions (smile, brow raise, disgust and so on) that widely vary among different people are captured. These labeled images are used to generate user-specific blendshapes. Additionally training data for the regressor is produced by fitting user blendshape model into setup images of the user with different head poses and facial expressions. From a training set consisting of the labeled 2D images and corresponding 3D facial shapes, they learn a shape regression model for automatic 3D facial feature alignment.
Runtime stage:
At run time, this 3D shape regressor is used in tracking the 3D positions of facial landmarks from a 2D video stream. The head's rigid transformation and facial expression parameters are calculated from the 3D landmark positions, and they are then transferred to a digital avatar to generate the corresponding animation.

4.3.3 Depth-camera tracking

Overview
With the popularity and affordability of low-cost commercial depth cameras (e.g., Microsoft's Kinect, iPhone X if $1000 can be considered as low cost), number of new techniques have been developed for performance driven facial animation that use depth information. Advantage of this devices include:

- Ease of deployment similar to regular web-cameras
- User is not required to wear any physical markers or specialized makeup
- Much more robust to different lighting conditions

The Kinect simultaneously captures 2D RGB image and a 3D depth map at 30 frames per second, using invisible infrared projection. However, data is quite noisy and the quality suffers compared to state-of-the-art performance capture systems based on markers and/or active lighting.

Realtime Performance-Based Facial Animation
[Weise et al., 2011] introduced a novel face tracking algorithm that combines geometry and texture registration with dynamic blendshape priors generated from existing face animation sequences.
Their pipeline consists of two main stages: constructing facial expression model and realtime tracking.
Facial expression model

First example expressions are captured with the Kinect and reconstructed using a morphable model [Blanz and Vetter, 1999] and further refined by warping using non-rigid registration method [Li et al., 2009].

To generate the full set of blendshapes of the user they employ example-based facial rigging as proposed by [Li et al., 2010] (explained in rigging section 3.2). To recap, the method takes as input a generic blendshape model, the reconstructed example expressions, and approximate blendshape weights that specify the linear combination of blendshapes for each example expression. Since the user is asked to perform a fixed set of expressions, these weights are manually determined once and kept constant for all users. Still the need for manual weight approximation is a major drawback. The user-specific blendshape model defines a compact parameter space suitable for realtime tracking.

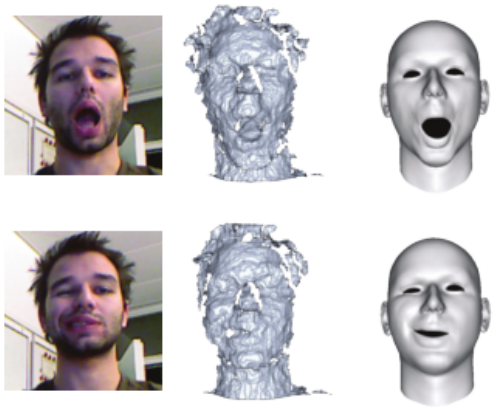

Figure 4.4: The user's facial expressions are reconstructed and mapped to different target characters in realtime.

Real time tracking Now the system needs to perform retargeting using depth and color information for each frame. As stated in Eq. 2.4 they first need to decouple the rigid from the non-rigid motion and directly estimate the rigid transform of the user's face before performing the optimization of blendshape weights. To obtain rigid parameters they align the reconstructed mesh of the previous frame with the acquired depth map of the current frame using ICP with point-plane constraints. Given the rigid pose, they estimate the blendshape weights that capture the dynamics of the facial expression. To prevent unrealistic face poses they regularize the blendshape weights with a dynamic expression prior computed from a set of existing blendshape animations $A = a_1, ..., a_l$. Each animation a_j is a sequence of blendshape weight vectors that sample a continuous path in the blendshape space. In order to estimate blendshape weights $\mathbf{w}$ for each frame they solve maximum a posterior problem defined as:

$$\arg\min_{\mathbf{w}}(-\ln p(\mathbf{G}|\mathbf{w}) - \ln p(\mathbf{I}|\mathbf{w}) - \ln p(\mathbf{w}, W_n)) \tag{4.8}$$

where G is a depth map, I is a texture, $W_n = \{w_{n-1}, ..., w_{i-n}\}$ is a set of weights prior to w. First term captures the alignment of the blendshape model with the acquired depth map, second term models texture registration and the last term is a prior term that ensure temporal coherence.

Realtime Facial Animation with On-the-fly Correctives

[Li et al., 2013] notices that the use of example-based blendshapes in combination with data-driven animation priors ([Weise et al., 2011]) often result in poor tracking accuracy around mouth and eye regions. While this capture method is sufficient to produce plausible results in the context of puppeteering, it fails to accurately capture the lip movements and high-frequency geometric details needed to convey nuanced emotions and micro-expression.

They instead propose a real-time facial animation framework where an adaptive PCA model, based on correctives, rapidly adapts to the expressions of the performing actor during the tracking. They begin

with the construction of a 3D model from a face scan in neutral expression. Similar to [Weise et al., 2011], they first aggregate multiple input depth map frames using a rigid alignment based on the fast iterative closest point method (ICP). They then warp the model of a statistically average face onto the integrated scan using a linear fit of PCA modes [Blanz and Vetter, 1999]. To capture details that are not present in the PCA model, they shrink-wrap the resulting model onto the input scan using the non-rigid ICP algorithm. For both deformable alignment steps, PCA and non-rigid ICP, they use point-to-plane constraints on the input scans and point-to-point 2D feature constraints (lips, eyes, and eyebrows).

Once the neutral face model of the actor is reconstructed, they automatically generate a set of initial blendshapes from a collection of 23 FACS inspired generic expressions using the deformation transfer algorithm of [Sumner and Popović, 2004]. The initial blendshapes are personalized but crude approximations of the actor's real expressions.

After creating the initial blendshapes of the actor, they begin tracking the actor's face. They first solve for a global rigid transformation using fast rigid ICP as before; they then perform an initial blendshape fit for every input frame using both 3D point constraints on the input scans and 2D facial features. The fitting is refined using a Laplacian deformation algorithm with the same constraints, followed by a projection onto an adaptive PCA space since the input data are noisy and incomplete. The adaptive PCA space is an orthonormal basis and consists of A anchor shapes and K additional corrective shapes.

They initialize the anchor shapes with $A = 23$ orthonormalized vectors from the initial blendshapes and learn K corrective shapes to improve the fitting accuracy over time. To train the correctives, they first collect new expression samples that fall outside of the currently used adaptive PCA space. These samples are obtained by warping the result of the initial blendshape fit to fit the current input depth map and 2D facial features using again a per-vertex Laplacian deformation algorithm. These samples are used to refine the corrective shapes using the incremental PCA technique.

Chapter 5

Challenges and Conclusion

In this document, we give overview of the facial animation pipeline. We review established methods as well as latest research on modelling, rigging and animating of faces. Our report is not exhaustive and a number of contributions in facial animation have been omitted. Mostly those that do not utilize blendshape models.
Here I am outlining few interesting challenges in the domain of facial animation:

- Current perfomance driven retargeting methods mostly used for characters with similar facial proportions (i.e., realistic models). Retargeting human facial motion to highly stylized or non-human looking character stil remains a challenging problem.
- Although motion capture data is a reliable way to capture the detail and nuance of live motion, reuse and modification for a different purpose remains a challenging task. Synthesis of the new animations from existing data hasn't been explored.
- Providing the user with an intuitive interface to control a broad range of facial expressions is difficult because character expressions are high dimensional but most available input devices are not. Intuitive control of individual degrees of freedom is currently not possible for interactive environments unless the user can use his or her own face to act out the motion using online motion capture.

Bibliography

[Achenbach et al., 2015] Achenbach, J., Zell, E., and Botsch, M. (2015). Accurate face reconstruction through anisotropic fitting and eye correction.

[Allen et al., 2003] Allen, B., Curless, B., and Popović, Z. (2003). The space of human body shapes: reconstruction and parameterization from range scans. *ACM transactions on graphics (TOG)*, 22(3):587–594.

[Amberg et al., 2007] Amberg, B., Romdhani, S., and Vetter, T. (2007). Optimal step nonrigid icp algorithms for surface registration. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE.

[Blanz and Vetter, 1999] Blanz, V. and Vetter, T. (1999). A morphable model for the synthesis of 3d faces. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 187–194. ACM Press/Addison-Wesley Publishing Co.

[Bouaziz et al., 2013] Bouaziz, S., Wang, Y., and Pauly, M. (2013). Online modeling for realtime facial animation. *ACM Transactions on Graphics (TOG)*, 32(4):40.

[Cao et al., 2013] Cao, C., Weng, Y., Lin, S., and Zhou, K. (2013). 3d shape regression for real-time facial animation. *ACM Transactions on Graphics (TOG)*, 32(4):41.

[Cao et al., 2014] Cao, C., Weng, Y., Zhou, S., Tong, Y., and Zhou, K. (2014). Facewarehouse: A 3d facial expression database for visual computing. *IEEE Transactions on Visualization and Computer Graphics*, 20(3):413–425.

[Chai et al., 2003] Chai, J.-x., Xiao, J., and Hodgins, J. (2003). Vision-based control of 3d facial animation. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 193–206. Eurographics Association.

[Deng et al., 2006] Deng, Z., Chiang, P.-Y., Fox, P., and Neumann, U. (2006). Animating blendshape faces by cross-mapping motion capture data. In *Proceedings of the 2006 symposium on Interactive 3D graphics and games*, pages 43–48. ACM.

[Deng and Noh, 2008] Deng, Z. and Noh, J. (2008). Computer facial animation: A survey. In *Data-driven 3D facial animation*, pages 1–28. Springer.

[Derakhshani, 2012] Derakhshani, D. (2012). *Introducing Autodesk Maya 2013*. John Wiley & Sons.

[Ekman, 2002] Ekman, P. (2002). Facial action coding system (facs). *A human face*.

[Ekman and Friesen, 1976] Ekman, P. and Friesen, W. V. (1976). Measuring facial movement. *Environmental psychology and nonverbal behavior*, 1(1):56–75.

[Friesen and Ekman, 1978] Friesen, E. and Ekman, P. (1978). Facial action coding system: a technique for the measurement of facial movement. *Palo Alto*.

[Lewis and Anjyo, 2010] Lewis, J. and Anjyo, K.-i. (2010). Direct manipulation blendshapes. *IEEE Computer Graphics and Applications*, 30(4):42–50.

[Lewis et al., 2014] Lewis, J. P., Anjyo, K., Rhee, T., Zhang, M., Pighin, F. H., and Deng, Z. (2014). Practice and theory of blendshape facial models. *Eurographics (State of the Art Reports)*, 1(8).

[Li et al., 2009] Li, H., Adams, B., Guibas, L. J., and Pauly, M. (2009). Robust single-view geometry and motion reconstruction. In *ACM Transactions on Graphics (TOG)*, volume 28, page 175. ACM.

[Li et al., 2010] Li, H., Weise, T., and Pauly, M. (2010). Example-based facial rigging. In *Acm transactions on graphics (tog)*, volume 29, page 32. ACM.

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

[Mori, 1970] Mori, M. (1970). The uncanny valley. *Energy*, 7(4):33–35.

[Orvalho et al., 2012] Orvalho, V., Bastos, P., Parke, F. I., Oliveira, B., and Alvarez, X. (2012). A facial rigging survey. In *Eurographics (STARs)*, pages 183–204.

[Osipa, 2010] Osipa, J. (2010). *Stop staring: facial modeling and animation done right*. John Wiley & Sons.

[Parke and Waters, 2008] Parke, F. I. and Waters, K. (2008). *Computer facial animation*. CRC Press.

[Ribera et al., 2017] Ribera, R. B. i., Zell, E., Lewis, J. P., Noh, J., and Botsch, M. (2017). Facial retargeting with automatic range of motion alignment. *ACM Trans. Graph.*, 36(4):154:1–154:12.

[Schneider and Eisert, 2009] Schneider, D. C. and Eisert, P. (2009). Fast nonrigid mesh registration with a data-driven deformation prior. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 304–311. IEEE.

[Sumner and Popović, 2004] Sumner, R. W. and Popović, J. (2004). Deformation transfer for triangle meshes. In *ACM Transactions on Graphics (TOG)*, volume 23, pages 399–405. ACM.

[Weise et al., 2011] Weise, T., Bouaziz, S., Li, H., and Pauly, M. (2011). Realtime performance-based facial animation. In *ACM transactions on graphics (TOG)*, volume 30, page 77. ACM.

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