Performance-driven Facial Animation

Garoe Dorta-Perez, Ieva Kazlauskaite, Richard Shaw

CM50247 - Visual Effects

Unit Leader: Dr Darren Cosker

University of Bath

May 2015

Signature of Author . . . . . .

Garoe Dorta-Perez, Ieva Kazlauskaite, Richard Shaw

Chapter 1

Introduction

"*Birds are the most accomplished aeronauts the world has ever seen. They fly high and low, at great speed, and very slowly. And always with extraordinary precision and control.*" [**?**]

A talented painter or sculptor is able to imagine and reproduce the subtle details of a human face. Hours of training and endless manual adjustments are required before an arbitrary shape resembles a facial expression. Furthermore, a human eye is trained to notice subtle changes in the expression of another human being, which also applies to animated humans, thus even the smallest discrepancies in an animated model are easily detected. Though artists are able to produce good quality and appealing results, the limitations in budget and time so prominent in the entertainment industry motivate the development of more automated, faster and cheaper models.

The naive approach that allows for faster performance is the keyframe animation; though it is still based on manual input, keyframe animation requires fewer frames as the intermediate expressions are interpolated over. However, this approach is still very laborious and time consuming. With the emergence of software and systems for motion capture, development of mathods for performance-driven animation has received significant attention from both the industry as well as the academia.

Chapter 2

Previous Work

2.1 Data Capture

2.2 Sparse Reconstruction

2.3 Facial Animation

The first attempts at using performance-driven facial animation in academia date back to late 20^{th} century while in industry was first used for guidance rather than production, i.e. the facial motion of Gollum in the *Lord of the Rings* was based on the motion capture data of the performance of the role actor Andy Serkis [PSS02]. Another notable example is the pipeline used in the making of *the Polar Express*; the motion capture data was used to construct a multilayer facial rig that was used by an artist to create the final animation [Ben05]. Though many other examples exist, often the exact technology and methods used in production are not disclosed.

In the last two decades, nearly two hundred academic papers that include words *face*, *animation* and *motion capture* were published in computer graphics and vision journals [Sco]. The increase in the quality of the results is explained by the advancements in both the capturing technology and the computational methods. We shall concentrate on the latter; for a brief discussion of the capturing methods see Sec. 3.2.

The work of Guenter et al. provides the first detailed and unified system for capturing and reconstructing facial performance [GGW*98]. Their work is mostly focused on motion tracking and production of labelled three-dimensional motion of the dots on the actors face. Once a sequence of moving dots is acquired, the authors scan the actors face to get a polygonal mesh that corresponds to the geometry of the face. The motion of the sparse dots is described in terms of offsets from the neutral position. Then the vertices in the mesh are moved by

calculating a linear combination of the offsets of the nearest dots, i.e. in each frame a weighted sum of the change in the dot position is calculated and added to the neutral position of the given vertex. The weights are chosen to be zero everywhere except in the one-ring neighbourhood where the weights depend on the distance of the vertex to the dots; the weights must sum to one. An additional stationary ring of dots is added on the edge of the face to ensure there is no motion outside the face. The algorithm suffers from noise introduced during in the tracking, the reconstruction and the three-dimensional scanning. Moreover, the choice of the nearest dots for each vertex in the mesh is not trivial due to the irregular distribution of the tracked dots. The method that assigns the blends has a number of manually adjusted parameters and does not guarantee to find the best set of reference dots. Additionally, the areas around the eyes and lips require a special treatment; the dots above and below the problematic areas are marked and are not allowed to be blended. The authors point out that a number of artefacts are visible in the resulting animation; some of these flaws attributed to the poor quality of the facial scan, and the fact that the reconstruction method is not robust to jitter and incorrect placement of the tracked data.

At around the same time Pinghin et al. proposed a method for synthesizing facial expressions form photographs [PHL*98]. The authors concentrate their attention on the capture and reconstruction part of the process. First, a three-dimensional model of the face pulling a number of different expressions is constructed, then the animation is created by morphing these meshes and producing the intermediate animation. The meshes are topologically consistent and the features are marked manually resulting in matching feature sets. Then linear interpolation between corresponding vertices in the different meshes is used to get a smooth morphing. Compared to the method proposed by Guenter et al., the approach of Pinghin et al. puts more emphasis on the capture, reconstruction, and creation of good quality meshes for different expressions resulting in an almost trivial animation production process.

Noh and Neumann addressed the problem of having to create a new model for a new animation by exploiting an existing database of high quality animations [NN01]. Given a new target model, dense correspondence between the source model from the database and the target is estimated using the a set of manually selected matching points. The volume morphing is performed using a weighted linear combination of Radial Basis Functions (RBF). However, some manual input is required when matching the features; for this authors use a set of heuristic rules. Then a cylindrical projection is used to transfer the vertices from the source onto the target model, thus the source vertices are embedded in the target surface. The animation is created by applying the adjusted target motion vector onto the source. The adjustments involve scaling, and change of direction according to the curvature of the source surface. Moreover, if the two models have very different geometries, then small neighbourhoods are used to determine the local changes that were imposed during the morphing of the source surface. As in most facial models, special attention is paid to the area around the mouth; the edges of the lips in the two models are aligned, and the motion transformations take into account the motion of both lips to ensure that the duplicated vertices move consistently. The presented approach has a number of limitations, the model requires manual matching, the eyelids, teeth and the tongue are not incorporated in the model, and it is only able to reproduce the motion of the source model and cannot produce novel movements.

Building on previous work on synthesis of facial expressions, by Joshi et al. address the problem of automating the creation of blend shapes [JTDP03]. The proposed method segments the face into characteristic parts that can then be modified independently to make a desired expression thus simplifying the process of creating new blend shapes. The authors argue that one of the advantages of controlling small regions is the increased number of expressions that may be produced using the blend shapes. Each frame in the keyframe animation is produced by calculating the linear combination of blend shapes in each region. A linear elasticity model is used to deform the surfaces. A related method, proposed by Pyun et al. generates a new facial expression by combining emotional and verbal key expressions [PKC*03]. The main limitation of this model is that a set of key expressions have to be designed by an artist.

In some situations the sets of blend shapes is only available for the target model. Choe and Ko propose a method that constructs a set of source key shapes given the target key shapes [CK05]. First, a mapping between the source and the target coordinate systems is constructed. Then the method iteratively refines the set of shapes using the captured source data. The authors build on their previous work, and use a muscle actuation basis as opposed to the surface-based key shapes. The elements in the actuation basis are linearly independent and they span the corresponding space, forming a complete basis for the facial expressions [CLK01]. However, the actuation basis is much harder to construct in comparison to the surface-based set of shapes and the relation between the motion of facial muscles and the resulting motion of the tissues is not straight-forward.

Vlasic et al. proposed an novel approach to facial animation that is based on a multilinear model [VBPP05]. The work offers an alternative approach to blend shape models, that involves creating a large dataset that encodes various features, for instance the identities, expressions and location of vertices. Each dimension of the data tensor corresponds to a unique feature, and due to the independence of modes, each feature may be varied without affecting the others. Such model can be extended to include an arbitrary number of features, and may be used for motion retargeting or actor replacement, when a database of three-dimensional scans is available. Moreover, the probabilistic interpretation of the model is related to probabilistic principal component analysis, and is able to deal with missing data.

A number of dimensionality reduction techniques have been used to produce facial animation; for example Blanz and Vetter apply principal component analysis (PCA) to create a controllable low dimensional model [BV99]. Deng et al. use PCA to find a correspondence between the motion capture data are the manually tuned blend shape models [DCFN06]. An extension to nonlinear dimensionality reduction techniques was presented by Wang et al. [WHL*04]. The proposed method separates the time-varying facial expression and the individual style associated with the performer. Then the locally linear embedding (LLE) framework is used to find a nonlinear mapping from the high dimensional space onto a low dimensional manifold. The LLE method is based on an assumption that small neighbourhoods around each data point may be treated as linear patches. Using the resulting generative model, new expressions or an animation may be produced by sampling from the low dimensional manifold. Dimensionality reduction is used when no blend shape model is available, or when different features of the data need to be extracted.

Most of the previously described methods are unable to capture and reproduce the fine details on the actor's face; Bicket et al. introduced a method that is able to create detailed wrinkles on the target model [BBA*07]. The approach is based on decomposition of the model into coarse features from the motion capture sequence, and fine features from the accompanying video. The novel part of their method relies on capturing, analysing and reproducing the wrinkled surface. In the tracking stage, uniform B-Splines are fitted to each fold of the skin. Then the shape of each fold is estimated by exploiting the self-shadowing effect, and finding the gradient that describes the shape of each fold. Moreover, the coupling behaviour is modelled separately. New wrinkles are synthesised by minimising the associated non-linear shell energy; the geometrical model, proposed by Grinspun et al. estimates the local curvature of a surface, and is able to capture the behaviour of thin flexible structures [GHDS03]. The facial model with detailed wrinkles relies on the quality of the captured data; in particular, the areas that are prone to small-scale deformations have to be painted in a way that minimises diffuse reflection.

One of the most successful attempts to produce a photo-realistic facial animation is known as the Digital Emily project [ARL*09]. The proposed method involves collecting high resolution data, building a detailed facial rig of the actor's face, producing an animation from the video data, and reproducing the high quality results. The blend shape model was created using approximately 30 three-dimensional scans; since each scan contained more than one discrete shape, a splitting algorithm had to be used to obtain larger set of controllable shapes. Additional subtle effects, such as the motion of the deformation of the eyelid when the eyes are closed were included to increase the realism of the results. The animation is produced using a number of example poses, and generating predictions for the required pose of the digital model. The entire production process is very time consuming and requires manual input from skilful artists and animators. The authors do not provide a detailed description of the method but their controllable facial rig is publicly available.

One recent development was proposed by Bouaziz et al., who develop a method for real-time facial animation [BWP13]. The new method does not rely on the construction of a three-dimensional expression model associated with the actor prior to the face animation stage. The presented method uses a consumer-level device that captures both the intensity and the depth; previous research by Baltrušaitis et al. indicates that the use of multimodal data improves the quality of feature tracking [BRM12]. To achieve real-time results, Bouaziz et al. use a template blend shape model that is constantly updated to better match the face of the performer. This update includes two steps. Firstly, PCA is used to construct a low dimensional representation of a large set of different facial meshes, then any new face is constructed by merging the average face with a linear combination of the orthonormal basis vectors produced by PCA. Secondly, surface deformation field is used to further refinement the blend shape model. Then the tracking, retargeting and production of an animation is performed in parallel with the optimisation algorithm that aims to personalise the blend shape model.

Concurrently, Li et al. combined blend shape models with facial tracking to animate a target character face [LYYB13]. The authors employ per frame correctives to achieve run time shape correction. The data is captured using RGB-D (intensity and depth) cameras. The initial state of this method consists of capturing a three-dimensional model of an actor's face, and PCA

and a large database of captured facial data is used to construct a generic face. Then a blend shape model is constructed using the deformation transfer algorithm, proposed by Sumner and Popović [SP04]. Then the motion is produced using blend shapes, and it is refined using an orthogonal adaptive basis. This basis is constructed using PCA, and it contains the initial blend shapes as well as a set of corrective shapes; these additional basis elements correspond to expressions that are not contained in the original shape set. The corrective shapes capture the fine-scale details. Due to the non-linear nature of the resulting blend shape model, Laplacian deformations are used to align the tracked data to the three-dimensional model. Then the expectation-maximisation algorithm is used in the adaptive PCA space to iterative improve the space of the corrective shapes. Specifically, given a number of sufficiently diverse input samples and the initial guess of the corrective space, the algorithm estimates the coefficients of the corrective space. Then, the algorithm finds the model that best explains the samples given the corrective coefficients. After each two cycles of the algorithm, the newly acquired corrective space is orthonormalised, and used to decrease the error during the tracking. This method is capable of producing appealing visual results in real-time. However, it requires a scan of the actors face, and the corrective shapes are not directly used during the retargeting.

A different direction was chosen by Garrido et al., who aim to reconstruct high quality three-dimensional models using data only from a monocular video [GVWT13]. The method relies on a pre-designed blend shape model, and uses optical flow to produce three-dimensional motion. Meanwhile, Xu et al. improved the retargeting of facial animation from an actor onto a digital model [XCLT14]. The authors use a multi-scale approach, where a targeted optimisation algorithm is used to achieve best results at the course and the fine scale. Moreover, the proposed method provides the user with a set of control tools that are used to alter the automatically produced results. This blend shape model produces high-quality visual results but it still requires significant input from the user.

2.4 Skin Rendering

Rendering realistic skin is a challenging task. As social beings we interact with interact with other individuals on a daily basis, which has made human perception quite sensitive to skin appearance, even more so with human faces. Skin is composed of several layers with different properties, to accurately simulate skin the light transport between this layers has to be simulated. The full effect of light scattering between two points on the surface can be modelled using a Bidirectional Surface-Scattering Distribution Function (BSSRDF).

Weyrich et al [WMP*06] proposed a two-layer model for skin rendering, the outer layer simulates the air-oil interface and the inner layer models the subsurface scattering in the skin. The authors considered the scattering to be homogeneous, with this assumption they measured the skin BRDF of several subjects in a light dome, while the scattering was sampled at three points in the face with a custom made sensor. The BRDF data was fitted to a Blinn-Phong and a Torrance-Sparrow isotropic models, and the scattering was fitted with a single transport coefficient. Donner et al [DWd*08] also proposed a two-layer model, however the authors allow for the layers to be heterogeneous. With this addition they are able to introduce the

effects of haemoglobin, veins and tattoos. Emotional induced haemoglobin variations have also been explored [JSB*10]. The authors measured the haemoglobin distributions of several subjects in different poses, then a linear combination of the captured data would determine the final haemoglobin distribution for a new sequence. Recently, Iglesias et al [IGAJG15] introduced a five-layer model to handle skin ageing. Haemoglobin, collagen and fat changes with age are modelled using the different layers.

Normal maps are use to alter the normals of the scene objects during rendering. This technique is used to add geometric detail to an object at rendering time without actually changing the geometry. Normal maps for skin rendering are usually captured using expensive light domes with a number of synchronized cameras [GTB*13, WMP*06].

Another technique to increase the quality of a face render is to scale the resolution of the textures being used. According to citeTian2011 super-resolution techniques can be classified into four classes: *frequency-domain*-based, *interpolation*-based, *regularization*-based and *learning* -based.

Frequency-domain-based super-resolution methods started with the seminal work of [TH84], the authors transformed the image using the discrete Fourier transform, where it is possible to formulate a system of equations relating the discrete and the continuous Fourier transform coefficients. The system is solved, the new coefficients are apply in the Fourier domain and a new image of higher resolution is retrieved by performing the inverse transformation. More recently, extensions that include denoising stages [CB05] or error handling for registration and blur identification [JF09] has been proposed.

Interpolation-based approaches used the fact that if we have several low-resolution images of a scene, each of them provides an amount of additional information about the scene. This technique projects all the low-resolution images into the hight resolution space and fuses the data maximizing the quality. Bose et al. [BA06] proposed a fusing technique using the moving least square (MLS) method, the authors estimated the pixel value as a polynomial approximation of the pixel neighbourhood with adaptive polynomial order for each pixel. Takeda et al. [TFM07] proposed a method to generalize this and other problems such as denoising or interpolation into a generalized kernel regression problem.

Regularization-based methods tackle the problem from a probabilistic viewpoint. An estimation using the maximum likelihood (ML) of the high-resolution image $\mathbf{X}^{ML}$ was proposed by Tom et al. [TK95], such that

$$\mathbf{X}^{ML} = \arg\max_{\mathbf{X}} P(\mathbf{Y}|\mathbf{X}), \tag{2.1}$$

where $\mathbf{Y}$ the unknown high-resolution image and $\mathbf{X}$ is the low resolution image. Extensions using maximum a posterior (MAP) with Markov random fields [HMD06] and Markov Chain Monte Carlo methods [TM10] have been researched.

In the *learning*-based area, Wie et al. [WL00] proposed a technique to synthesized new

textures based on a sample image and a random noise. The main idea of the authors work was to maximize the local similarity when choosing a new pixel in the synthesized texture image, based on the already built neighbourhood in the new image. Ashikhmin et al. [Ash01] presented a method to generate new textures using a goal image by greedily extending existing patches whenever possible. Hertzmann et al. [HJO*01] combined and extended Ashikhmin et al. [Ash01] and Wie et al. [WL00] methods by adding a second example image and using more complex distance metric to choose the next synthesized pixel. Graham et al. [GTB*13] applied Hertzmann et al. [HJO*01] example-based filter to generate bump maps with increased quality for skin rendering. An alternative approach using a dictionary of samples was presented by Jianchao et al. [YWHM10]. This method is restricted to generating super-resolution images, however, the previous methods support a wide variety of filter effects. For an in depth analysis of super-resolution techniques, we refer the readers to Tian et al. [TM11] survey.

Chapter 3

Methodology

3.1 Pipeline

The proposed pipeline is shown below...

3.2 Data Capture

Two DSLR cameras were positioned in a stereo configuration to capture the actor's facial performance, as shown in Figure 3-1. The facial performance was recorded on video with a resolution of 640×480 at 60fps, and the cameras were roughly synchronised by aligning the two separate audio signals. Fixing the cameras close to one another at approximately 10cm separation, and about 1m distance from the subject, meant that the images from both views would be quite similar, allowing for easier feature matching across the two images. However, the camera spacing was sufficiently large to enable both sides of the face to be in view, and to allow for triangulation of points on the face; a good rule of thumb is for the cameras and subject to subtend an angle of at least 5 degrees.

Markers used to track the facial performance were placed on the face in the configuration shown in Figure 3-2. 97 markers were used in total, and were drawn onto the actor's face with a makeup pencil so as to be easily removed after the capture session. The positions of the markers were roughly based on those used in the Surrey Audio-Visual Expressed Emotion (SAVEE) Database [Sur], as shown in Figure 3-3, although they use fewer markers with 60 in total. In hindsight, using slightly fewer number of markers might have been preferable, as consistently detecting and tracking a large number of markers became quite time-consuming and labour intensive. Furthermore, rather than simply using a black pencil, it might have advantageous to use markers of a significantly different colour to the tones of the human face in order to aid marker detection and tracking, perhaps even using retro-reflective markers as in many commercial motion capture systems [Vic].

Figure 3-1: The stereo camera setup used to capture the facial performance.

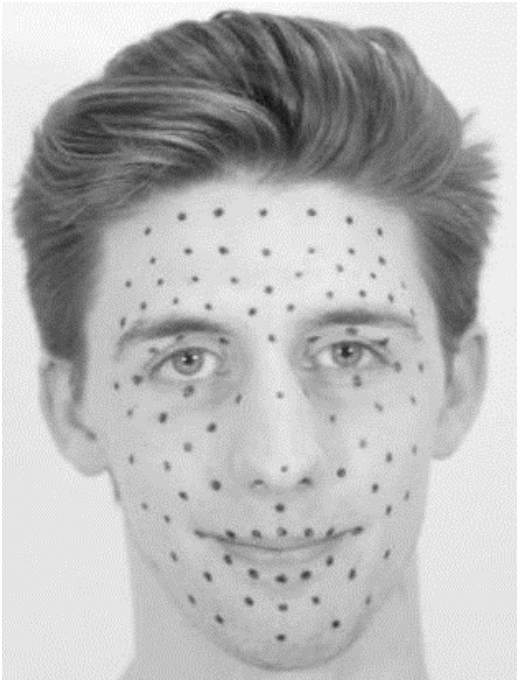


Figure 3-2: The configuration of markers used for tracking the facial performance of the actor.

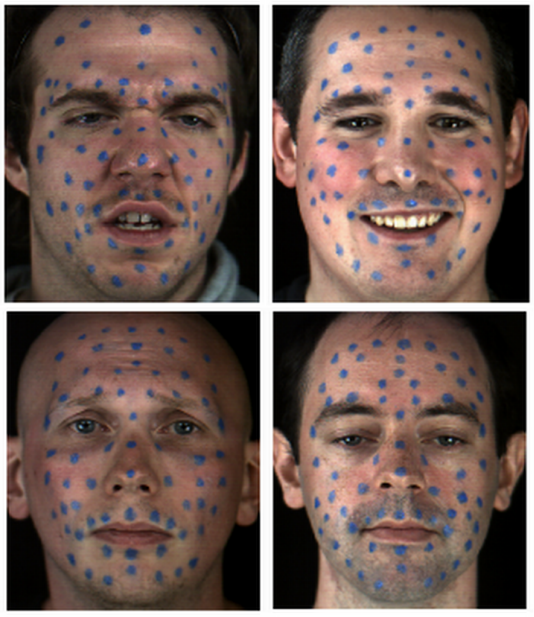


Figure 3-3: Images from the Surrey Audio-Visual Expressed Emotion (SAVEE) Database [Sur].

3.3 Camera Calibration

In order to obtain the 3D coordinates of the markers on the face, the cameras first needed to be calibrated. This was achieved using a checkerboard pattern held in a number of different positions in front of both cameras simultaneously, as shown in Figure 3-4. We made use of the Camera Calibration Toolbox for Matlab [Cal] provided by Jean-Yves Bouguet. It was important to ensure that the checkboard remained within the frame of both cameras during the entire calibration process.

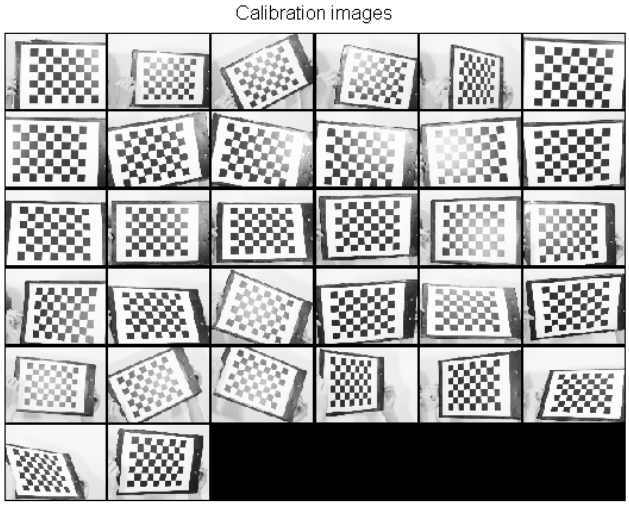


Figure 3-4: A checkerboard pattern was used to calibrate the pair of stereo cameras.

As a result of the stereo calibration, we obtain an estimate of the intrinsic and external parameters of the two cameras, from which we can compute the camera projection matrices using the perspective projection equation:

$$\begin{pmatrix} su \\ sv \\ s \end{pmatrix} = \mathbf{K}[\mathbf{R}|\mathbf{t}] \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} su \\ sv \\ s \end{pmatrix} = \begin{bmatrix} fk_u & 0 & u_0 \\ 0 & fk_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

The resulting extrinsic parameters, describing the rotation and translation between the views, are visualised in Figure 3-5.

Given the estimated camera projection matrices, we can then rectify the stereo images for every frame from a captured image sequence, so that the eipolar lines are horizontal. This

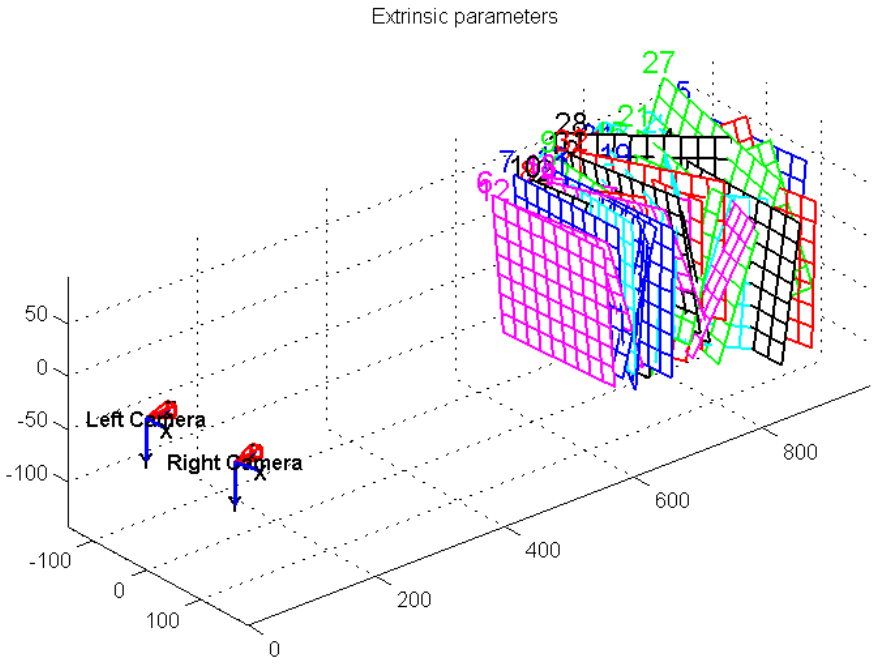


Figure 3-5: The resulting extrinsic parameters of the calibrated stereo camera setup.

reduces the stereo matching problem to a 1D search problem, as the same 3D is constrained to lie on the same pixel row in each pair of images. A pair of rectified images and corresponding epipolar lines are shown in Figure 3-6.

Figure 3-6: Using the stereo calibration results, each pair of images in a captured image sequence is rectified so that corresponding epipolar lines lie on the same pixel row. This reduces the stereo correspondence problem to a 1D search along the epipolar lines.

3.4 Initial Marker Detection

The facial markers are detected in the first frame of a captured image sequence and then are tracked throughout the length of the sequence. The initial marker detection is achieved by detecting SIFT features in the image. We employ the open source Matlab implementation VLFeat [VF08] to do this.

The marker detection process for the first frame of a given stereo image sequence is carried out as follows. First the user defines a region of interest in the left image around the face of the actor to be tracked, by drawing a contour in the image. We make use of the Matlab function `roipoly` for this. Then, SIFT features are detected within the selected image region, choosing the appropriate SIFT parameters to pick out features of around 5 pixels in diameter - this may take a few tries to get right. Once reasonable SIFT features have been detected, additional feature points can then be added manually, and also wrongly detected points can be removed, by interactively selecting certain points the user wishes to change. Once the user has successfully detected all 97 facial markers in the left image, the corresponding marker positions are found in the right image by searching along the corresponding epipolar lines; simply the same row of pixels in the right image since the images have been stereo rectified beforehand. The matching points in the right image are found by computing the normalised cross-correlation between image patches:

$$C_{NCC} = \sum_{i=1}^{N} \frac{(I_i^L - \mu^L)(I_i^R - \mu^R)}{\sigma^L \sigma^R} \tag{3.1}$$

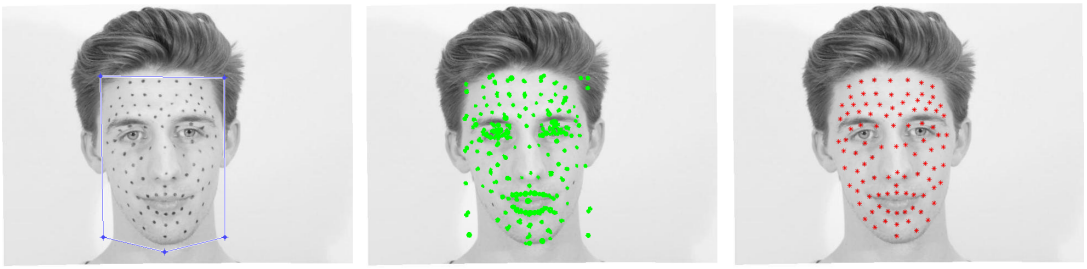


Figure 3-7: Facial markers are detected by first selecting a region of interest, detecting SIFT features, and then removing unwanted points or adding additional points.

3.5 Marker Tracking

With the facial markers detected in the first frame, the next task was to track the positions of the markers as they move throughout the image sequence.

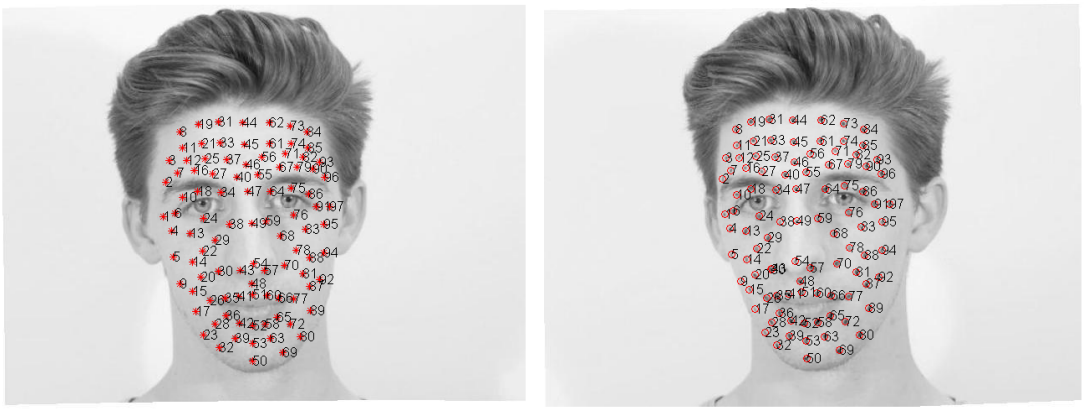


Figure 3-8: Detected facial markers in the left image are matched to markers in the right image pair by computing the normalised cross-correlation between image patches around the points.

3.5.1 Optical Flow

3.5.2 KLT Tracker

[ST94]

3.5.3 Kalman Filter

3.6 Sparse Facial Reconstruction

Once we have obtained the 2D image coordinates of the facial markers for every frame in a stereo image sequence, we then compute the sparse 3D reconstruction over the sequence. In one frame, given a pair of image correspondences $\mathbf{w} = (u, v, 1)^T$ and $\mathbf{w}' = (u', v', 1)^T$ of the same 3D point $\mathbf{X} = (X, Y, Z, 1)^T$, we can write two equations using the projection matrices $\mathbf{P}$ and $\mathbf{P}'$:

$$\begin{pmatrix} su \\ sv \\ s \end{pmatrix} = \mathbf{P} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \qquad \begin{pmatrix} su' \\ sv' \\ s \end{pmatrix} = \mathbf{P}' \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}. \tag{3.2}$$

For each pair of image points, two equations can be written:

$$u = \frac{p_{11}X + p_{12}Y + p_{13}Z + p_{14}}{p_{31}X + p_{32}Y + p_{33}Z + p_{34}} \quad u' = \frac{p'_{11}X + p'_{12}Y + p'_{13}Z + p'_{14}}{p'_{31}X + p'_{32}Y + p'_{33}Z + p'_{34}}$$
$$v = \frac{p_{21}X + p_{22}Y + p_{23}Z + p_{24}}{p_{31}X + p_{32}Y + p_{33}Z + p_{34}} \quad v' = \frac{p'_{21}X + p'_{22}Y + p'_{23}Z + p'_{24}}{p'_{31}X + p'_{32}Y + p'_{33}Z + p'_{34}}$$

Rearranging, these can be written in the form $\mathbf{AX} = 0$ as follows, where $\mathbf{p}_i$ denotes the *ith* row of the projection matrix $\mathbf{P}$

$$\begin{bmatrix} u\mathbf{p}_3^T - \mathbf{p}_1 \\ v\mathbf{p}_3^T - \mathbf{p}_2 \\ u'\mathbf{p'}_3^T - \mathbf{p'}_1 \\ v'\mathbf{p'}_3^T - \mathbf{p'}_2 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = 0. \tag{3.3}$$

Two equations are acquired from each image point in the two views, giving a total of four equations with three unknowns - the problem is over-constrained. A linear least-squares solution can be found by performing Singular Value Decomposition on the matrix $\mathbf{A}$. The 3D coordinates of the point $\mathbf{X}$ are the singular vector corresponding to the smallest singular value of $\mathbf{A}$, i.e. the last column of the matrix $\mathbf{V}$, where $\mathbf{A} = \mathbf{UDV}^T$. This is implemented in Matlab in the function `Reconstruct.m`, which takes in the two camera projection matrices for a pair of stereo cameras and the matching 2D image points, and returns an array of corresponding 3D points.

3.6.1 Stabilising Head Movement

It is important for the actor's head to remain still throughout the sequence so that the reconstructed facial expressions are expressed as true non-rigid deviations from the neutral facial expression. Proper stabilisation of the head is required to avoid artefacts in the resulting blendshape facial animation and retargeting process [BB14]. If the rigid head movement is not removed, then the expression shapes will contain this 'baked-in' rigid motion, and any facial animation constructed from the expressions will also contain this unwanted rigid head motion. In industry, face-stabilisation is still usually performed manually; Weise *et al*. [WLGP09] aim to remove the rigid motion component of multiple expressions using ICP registration and Vlasic *et al*. [VBPP05] use Procrustes alignment.

In our implementation, we attempt to remove the majority of the actor's rigid head motion by performing a Procrustes analysis [Gow75] using a number of approximately rigid points. At least four points are required for the Procrustes alignment. The rigid points were chosen to be the tops of the ears, the tip and bridge of the nose, and the centre point on the top of the forehead, as shown by the dots in Figure 3-10. However, none of these points are actually truly rigid, so this is just an approximation to the true stabilisation. In a future implementation, it would be better to actually track a few points known to be practically rigid, for example by placing markers on the top of the head.

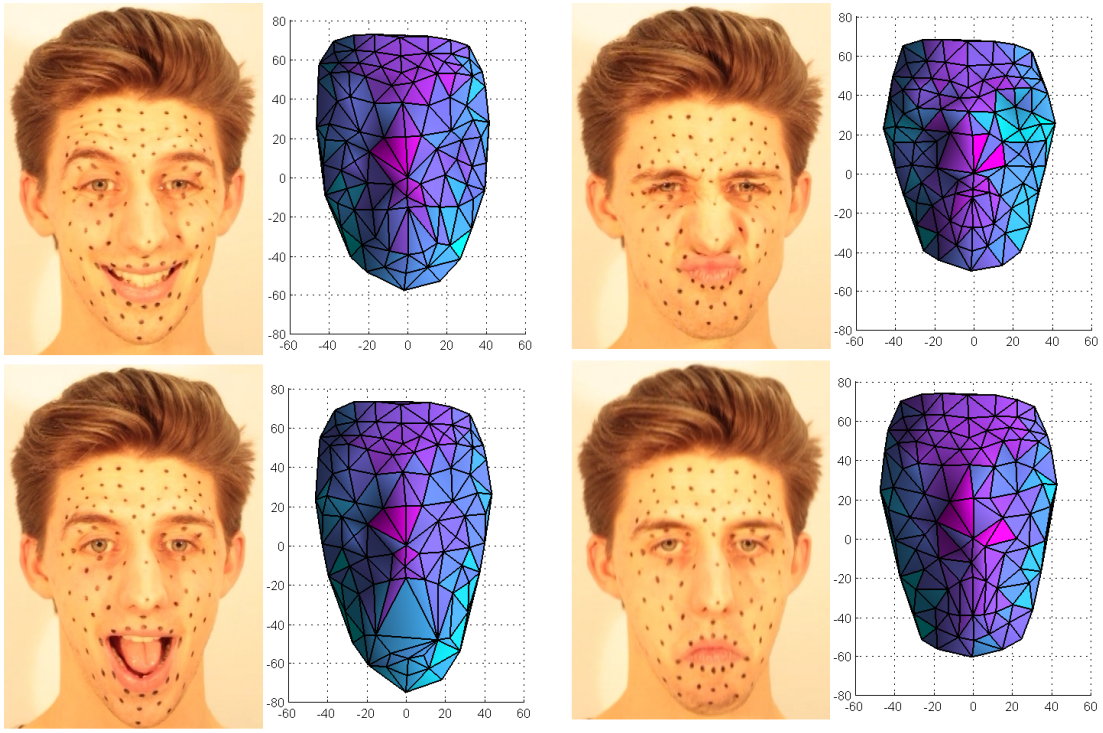


Figure 3-9: A sparse 3D reconstruction is computed for every frame of an image sequence using the detected marker positions and the calibrated camera projection matrices. A sample of image frames and the corresponding 3D reconstructions are shown above.

The Procrustes analysis computes a least-squares fit of each facial expression in a given frame to the neutral face, given the rigid vertex correspondences. In Figure 3-10, the rigid correspondences are plotted in red for the neutral face, and blue for a particular frame. We do this for the entire image sequence so that in every frame the head pose is aligned to the neutral position. We have found that the average error over an image sequence is usually around the order of 10^{-3}. However, it is not simply the case that the smaller the error means the better the alignment of the expression to the neutral face, because the head can be orientated in such a way that can make the error in the rigid vertex correspondences small, but can result in an unnatural head pose. Through experimentation, we have found that choosing points that span the entire face and are placed the extremities of the face, such as the ears, generally results in a better alignment than rather selecting a few points close together around the centre of the face, such as around the mostly rigid nose region. This makes sense because a small deviation in points around the centre of mass will lead to larger displacements in the other points, whereas small deviations in points at the edges will have much less of an impact on points in the centre of the face.

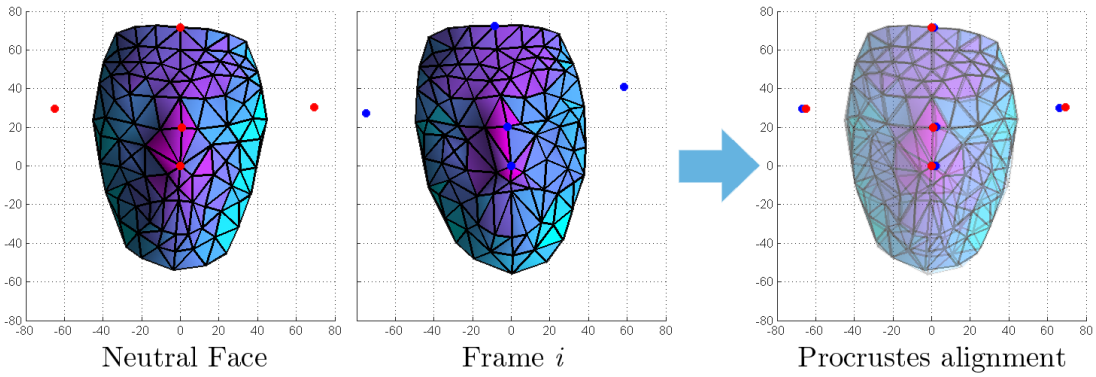


Figure 3-10: The head pose in each frame of an image sequence is aligned to the neutral face position using Procrustes analysis. The rigid vertex correspondences are shown in red for the neutral pose, and blue for a given frame.

3.7 Facial Animation

Given the motion capture data, our goal is to produce an animation that resembles the captured motion. We use the digital Emily model designed by Alexander et al. [ARL*09]. In this section we introduce the techniques used to match three-dimensional surfaces and point clouds, and the optimisation methods used to estimate the parameters that describe the motion of the digital model.

3.7.1 Thin Plate Splines

Thin plate theory addresses problems that commonly arise in areas of natural sciences and engineering when trying to model the behaviour of a thin sheet of some material. The possible processes include but are not limited to stretching, bending, crumpling, buckling, shrinking, straining and tearing. The corresponding mathematical model is based on ideas from differential geometry as well as continuum mechanics, and the set of equations describing the aforementioned phenomena are often notoriously difficult to solve. Therefore, in Computer Graphics, as well as other fields, a number of simplifying assumptions are made when constructing a thin plate model.

A thin plate is considered to be a two dimensional object, i.e. it is assumed that the thickness is infinitesimal. The geometry of the object is often simplified to reduce the computational cost. Thin plate splines (TPS) are a two-dimensional counterpart of the cubic spline [SS91]. TPS are a deformation method based on the assumption that a thin surface deforms in a way that minimises the surface bending energy. The bending energy is proportional to the change in the second fundamental form. Specifically, given two corresponding sets of point $\{(x_i, y_i\}_i^N$ and $\{v_i\}_i^N$ there exists a height field mapping between the two, $f : \mathbb{R}^2 \to \mathbb{R}$. The bending energy corresponding to this mapping is proportional to the second order derivatives of the mapping:

$$E_{bend}(f) = \lambda \iint \left(\left(\frac{\delta^2 f}{\delta x^2} \right)^2 + 2 \left(\frac{\delta^2 f}{\delta xy} \right)^2 + \left(\frac{\delta^2 f}{\delta y^2} \right)^2 \right) \mathrm{d}x \, \mathrm{d}y, \tag{3.4}$$

where λ is smoothing parameter, which balances the quality of fit and the amount of bending, i.e. the wiggliness of the function. TPS finds the transformation that fits the data while minimising the bending energy. Note that TPS may also be defined in terms of the radial basis functions that are used for smooth scattered data fitting. The RBF solution for thin plate splines is:

$$f(x, y) = \sum_{i=1}^{N} \omega_i \, \phi(\|(x, y) - (x_i, y_i)\|), \quad \text{where } \phi(r) = r^2 \log(r). \tag{3.5}$$

To ensure that the function f has square-integrable second derivatives, the following conditions are imposed:

$$\sum_{i=1}^{N} \omega_i = \sum_{i=1}^{N} \omega_i x_i = \sum_{i=1}^{N} \omega_i y_i = 0. \tag{3.6}$$

These conditions and the data fitting requirement may be combined into a linear system of equations:

$$\begin{bmatrix} K & P \\ P^T & 0 \end{bmatrix} \begin{bmatrix} \omega \\ o \end{bmatrix} = \begin{bmatrix} v \\ o \end{bmatrix}, \tag{3.7}$$

where K encodes the relation between the data points, i.e. $K_{ij} = \phi(\|(x_i, y_i) - (x_j, y_j)\|)$, $P_{i\cdot} = (1, x_i, y_i)$ contains the variables from Eq. 3.6, ω contains the values of ω_i, and v contains the target function values v_i. The variables 0 and o denote the zero matrix and the zero column vector respectively. Solving this linear system of equations gives the desired transformation in two dimensions. The method extends naturally to three-dimensional problems by including a dependence on an additional variable in the calculations described above. Radial basis functions are commonly used in the field of performance-driven animation [JTDP03].

3.7.2 Non-Rigid ICP

Iterative Closest Point (ICP) is an algorithm used to align two partially overlapping point clouds by minimising the square error between corresponding points. The quality of the results produced by this algorithm is sensitive to the initial guess at a solution, i.e. ICP only refines the initial estimation. See Alg. 1 for the outline of the algorithm. Mean square error algorithm is used to calculate the average of the squared errors between the target and the transformed source points. Thus the objective function is a function of rotations and translations. The output of the algorithm is a transformation matrix that provides the optimal mapping between the two point clouds within a given threshold. The transformation matrix may be split into rotation and translation. The algorithm solves a linear system of equations where the unknowns are the coefficients in the rotation matrix and the translation vector and the known point cloud values are used as coefficients. The method is often used to find a transformation between a point cloud and a surface; in that case the surface normals are used as additional input.

Algorithm 1: Iterative Closest Point.

```
Data: source point cloud
      target point cloud;
      initial guess;
      error threshold;
while error > threshold do
    for point in source do
        find closest point in target;
    end
    use mean squared error to find best transformation that aligns source to target ;
    apply transformations;
end
```

The original ICP algorithm is limited to rigid transformations, i.e. rotation and translation, which have only 6 degrees of freedom. However, in order to capture scaling, shearing and other more complicated transformation, an affine mapping should be used. Thus an extension of the ICP algorithm, called non-rigid ICP is used when additional degrees of freedom are present. Allen et al. proposed a non-rigid registration method that uses a numerically non-linear solver to find a smooth affine mapping [ACP03]. Additional constrained are imposed by manually matching some known marker locations in the two datasets. The method exhibits slow convergence and often leads to local minima; consequently, the authors use a multi-resolution approach where the optimisation is first performed on a low resolution mesh, and then optimised on the high resolution mesh. Amberg et al. combined the ICP algorithm with the method of Allen et al. to overcome the issues with convergence [ARV07].

3.7.3 Blendshape model

The aim of this project is to used captured data of a face to animate a given model. We use the digital Emily model that comes with 68 controllable blendshapes ranging from simple eyebrow movements to complicated lip corner pulls. Each blendshape has a weight w_i associated with it, and $w_i \in [0, 1]$. The model includes eyes and teeth. The aim is to find a combination of these blendshapes that produce a specified facial expression. In particular, we are interested in reproducing the captured sequence of expressions.

First attempt. We manually construct a sparse set of points that corresponds to the sparse source mesh Fig. **??**???. Then the neutral expression of the subject is matched with the neutral expression of Emily by transforming the sparse source mesh to the sparse target mesh. The transformation is then stored, and it is used to map all the expressions in the captured sequence. This produces a sparse Emily sequence; using TPS again we warp the dense Emily mesh according to the positions of the sparse points. The quality of the resulting animation is poor for a number of reasons, see Fig.**??**. Firstly, the sparse points are not able to constrain the dense mesh, especially since parts of the face, for example the eyelids and the lips are not tracked in the sparse mesh. Consequently, numerous artefacts appear on the dense mesh, and they are particularly noticeable around the lips as no boundary conditions are imposed. Secondly, though there is a clear correspondence between the generated Emily sequence and the source sequence, the generated expressions look unnatural. This may be explained by the differences in the geometry and anatomy of the two faces; for instance, the source may be able to pull expressions that may not be mimicked by the model. In addition, the errors in tracking, reconstruction, and manual selection of sparse points on Emily further reduce the quality of the animation.

Simplified mesh. Our initial attempt to improve the method involved simplifying the dense Emily model by removing the part of the mesh that corresponds to the lips. Though the simpler mesh exhibited slightly better behaviour, i.e. there were fewer visible artefacts, the sparse point cloud was still unable to constrain the dense mesh well enough to produce realistic results, see Fig. **??**.

Blend shapes. In an attempt to improve the visual quality of the generated animation, we decided to use a set of Emily key-shapes, that were readily available to us. The main argument for using these blend shapes is that in most situations a simple linear relation exists between an arbitrary facial expression and the set of key-shapes. There exist a number of different options for create a set of shapes; the shape may encode the entire geometry of the case, the displacement from the neutral face, or the local features of a face. Alternatively, a set of blend shapes may be producing using a dimensionality reduction technique. Such set of blend shapes is orthogonal, thus the features encoded by each shape are controlled independently; however, such shapes do not allow for intuitive facial transformations making it difficult to perform manual post-process editing.

Let us define a set of blend shapes $\mathbf{B} = [\mathbf{b}_0, \ldots, \mathbf{b}_n]$ where $\mathbf{b}_0$ is the mesh of a neutral face, and each of the $\mathbf{b}_i$ correspond to a basic facial expression, for example the raise of an eyebrow. Unless stated otherwise, we shall use 68 shapes provided with the digital Emily model. Then

a new facial expression $\mathbf{F}(\mathbf{w})$ may be constructed by finding an appropriate linear combination of the offsets of the basic shapes:

$$\mathbf{F}(\mathbf{w}) = \mathbf{b}_0 + \sum_{i=1}^{N} w_i \, |\mathbf{b}_i - \mathbf{b}_0|, \tag{3.8}$$

where $\mathbf{w} = [w_1, \ldots, w_n]$ is a set of weights that describe how much each of the basic shapes affect the new expression $\mathbf{F}(\mathbf{w})$. Typically, a convexity constraint is imposed on the weights; in our case the choice of constraints is influenced by the constraints present in our blend shape model, i.e. $w_i \in [0, 1]$. Eq. 3.8 may be written as a linear system of equations as follows:

$$\mathbf{F}(\mathbf{w}) - \mathbf{b}_0 = \hat{\mathbf{B}}\mathbf{w}, \tag{3.9}$$

where $\hat{\mathbf{B}}$ is the original set of blend shapes without the neutral expression.

A number of different numerical optimisation methods where tested when solving for the blend shape weights. An unconstrained solution may be calculated by inverting matrix $\hat{\mathbf{B}}$, and multiplying it from the left with the left hand side of Eq. 3.9. Instead we use Matlab's constrained non-negative least-squares solver (`lsqnonneg`) and constrained linear least-squares solver (`lsqlin`) [MAT13]. For the non-negative least-squares solver we formulate our problem as follows:

$$\min_{\mathbf{w}} ||\hat{\mathbf{B}}\mathbf{w} - (\mathbf{F}(\mathbf{w}) - \mathbf{b}_0)||_2^2, \ \mathbf{w}_i \geq 0, \ i = 1, \ldots, n. \tag{3.10}$$

The solver uses an active set method, and iteratively improves the active set of basis vectors; it converges in finite time. The same formulation is used for the linear least squares solver though it includes an additional upper limit on the argument. This solver uses a reflexive Newton method which is able to accurately locate the local minimisers of large systems, and exhibits global convergence. It is also able to maintain sparsity of matrices but is generally slower.

Our initial tests of the optimisation algorithms were carried out on a two-dimensional sequence of facial motion. Approximately 90 points on a sequence were marked and tracked to produce a sparse motion sequence. Then a set of key frames were hand-picked to represent the extreme cases of the facial motion, see Fig. 3-11. The sparse points from these frames are then used as blend shapes. The two solvers were tested on the entire motion sequence. We shall compare the results by estimating the average error between the pixels in the original sequence and the weighted sequence.

Fig. 3-12 and Fig. 3-13 illustrate the concept of blend shapes. Frame 10 is very close to one of the blend shapes, and both optimisation methods are able to represent this expression well; the average error associated with the constrained non-negative least-squares solver is less than 0.29, which corresponds to approximately 0.01 centimetres, while the average error for the second solver is less than 0.21 (approximately 0.09 centimetres). Note that as expected both methods use frame 9 with a large weight. However, the other blend shapes are different for the two methods. This suggests that none of the blend shapes is able to capture the subtle change between frames 9 and 10. In comparison, frame 400 is twelve frames away from a frame that was chosen as a blend shape, resulting in an increase in average error for he two

methods (approximately 0.49 in both cases). As expected, both methods use the two blend shapes that are the nearest to frame 400 in terms of timing. The weights used for these two shapes are the same to three significant digits, and the extra shape used by the first method is has a low weight assigned to it. Note that the first method does not obey the restriction of $\sum_{i=1}^{N} w_i = 1$. Though the second method is able to produce slightly more accurate results, the difference does not seem significant.

Figure 3-11: Set of facial expressions used as blend shapes. The first frame is used as a neutral expression.

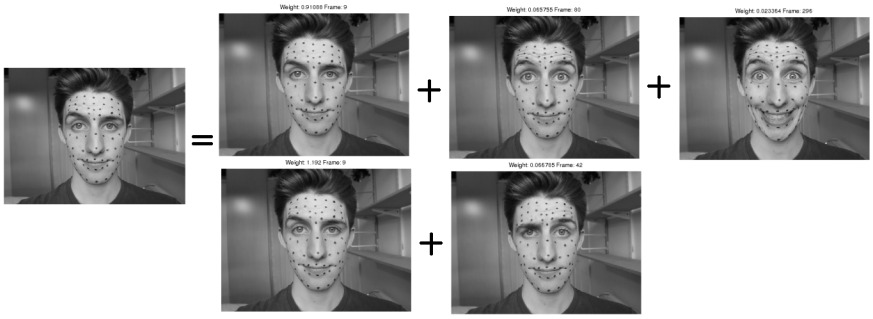


Figure 3-12: Frame 9 on the left represented as a linear combination of the key frames. The top solution was obtained using the `lsqnonneg` solver while the bottom one was obtained using `lsqlin`.

PCA. An alternative set of key shapes may be constructed by applying Principal Component Analysis (PCA) to the observed high-dimensional data. Generally, PCA is used to transform a set of correlated observations into a set of values of linearly uncorrelated principal components. If the number of principal components is smaller than the number of original variables, then the method may be used for dimensionality reduction. PCA tries to find a set of orthonormal axes which under projection preserve the highest variance. When formulated as an eigenvalue problem, it finds the eigenvectors of the covariance matrix with the largest eigenvalues.

PCA has been used in facial animation as an automatic way of constructing a set of blend shapes. The main advantages of using such blend shapes is that he construction does not re-

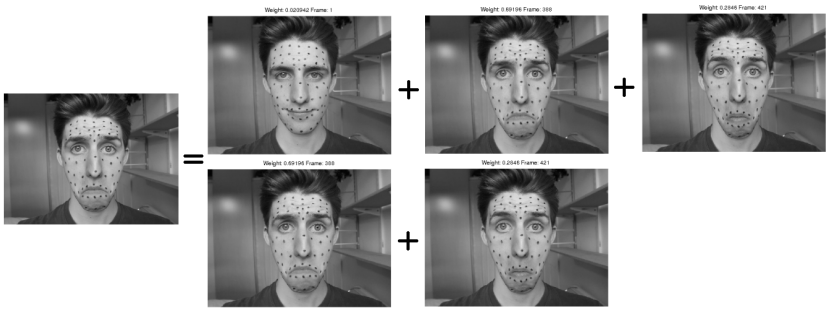


Figure 3-13: Frame 400 on the left represented as a linear combination of the key frames. The top solution was obtained using the `lsqnonneg` solver while the bottom one was obtained using `lsqlin`.

quire artistic skills, and the set of shapes are guaranteed to be orthogonal. The second argument is particularly important in compression algorithms, i.e. when the aim is to find the smallest possible set of blend shapes that is able to represent the data while minimising the squared reconstruction error. We use the Matlab implementation of PCA to find a set of PCA eigenvectors that represent the directions with highest variance in our data. Given this new set of shapes, we use the same optimisation algorithm as previously described, replacing the neutral expression with the mean of the data.

To compare the results with a different number of blend shapes, we calculate the average reconstruction error for a 500 frame sequence. When the first 3 shapes are used, the error is approximately 0.16 centimetres and 0.25 centimetres for the two optimisation methods respectively. Introduction of five additional shapes reduces the error by approximately 0.02 centimetres. Thus PCA does not outperform the model with manually designed set of key shapes. An additional drawback of the PCA basis is that it is not intuitive, i.e. the key shapes do not correspond to recognisable facial expressions. As a consequence, manual editing of the resulting animation is very complicated. Throughout the rest of the project we use the standard set of 68 blend shapes from the Digital Emily project [ARL*09].

3D model. Next, we shall consider the animation of a three dimensional motion sequence introduced in Sec. **??**. This sequence is obtained using a linear elasticity method, as previously described. We analyse the results from a talking sequence that contains 371 frames. As in the two-dimensional case, at each frame of the sequence, the chosen optimisation method calculates the best possible combination of blend shape weights given the Emily motion sequence. In particular, the input sequence is a sparse sequence of the motion that was transformed from the capture space to the digital model space.

We start by manually marking the three-dimensional face of Emily; the try to replicate the pattern we used when capturing the actor's face, see Fig. 3-2. The markers are placed at the vertices of the mesh, and the indices of these vertices are used to find the location of the corresponding markers on the meshes of blend shapes; Fig. 3-14 illustrates the locations of markers on a number of blend shape meshes. Then the optimisation methods are used to extract the best linear combination of shapes together with the corresponding weights. The

average reconstruction error for the non-negative least-squares solver is approximately 0.26 centimetres, compared to 0.23 for the constrained linear least-squares solver, see Fig. 3-15. A number of factors may increase the error. Firstly, the expressions captured in Emily blend shapes are fairly mild, i.e. the shapes do not capture the most extreme motion of the face. Consequently, the estimated facial movements may not be expressed as a linear combination of the blend shapes. Secondly, as the blend shapes are not orthogonal, they may combine in unexpected ways, for example cancel each other. This behaviours may be avoided if a set of PCA-produced shapes is used. Also, note that the mesh used is not symmetrical, i.e. the left side of the face does not match the right side. This may lead to a sideways motion that cannot be captured by the blend shapes.

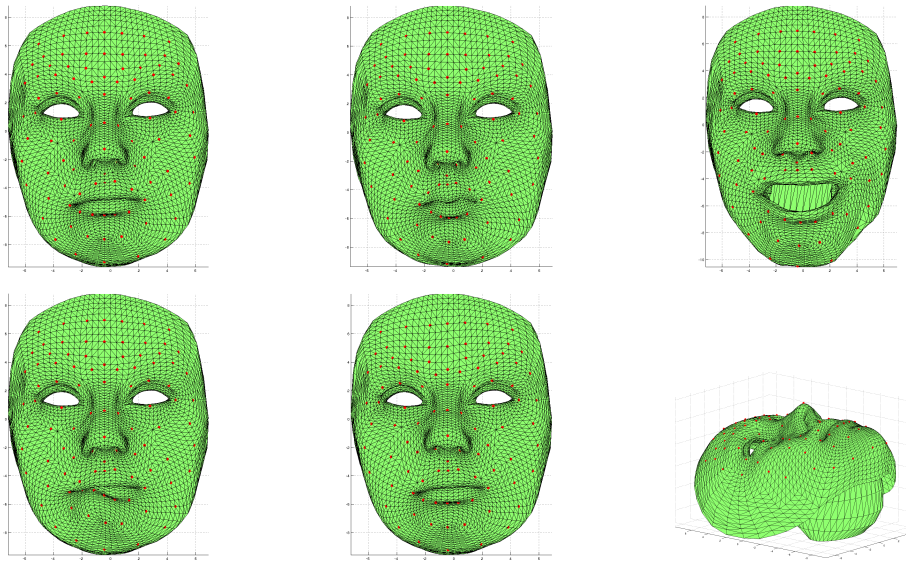


Figure 3-14: Examples of marked blend shapes.

The use of blend shapes greatly improves the visual quality of the resulting animation. However, the animation still lacks realism and expressiveness, thus we implement a number of changes to the original model. Firstly, we aim to find a sparse solution, i.e. minimise the number of blendshapes used at any given time. This is done by adding a penalty term on the $L1$-norm of the weights to the original minimisation problem. Then an iterative method is used to find the optimal set of weights; we use a Matlab implementation of the LASSO algorithm [Sch05]. After careful adjustment of the scaling parameter, the least squares solver is able to reduce the mean reconstruction error by approximately 0.03 centimetres. However, the best results were achieved without imposing the upper and lower bound on the weights. We found that in our case it is more import to impose these bounds than to favour sparse solutions. Moreover, the solution produced by the initial solver uses on average 21 blendshapes per frame, which does not seem unreasonable given the complexity of human facial motion.

As noted by Bouaziz et al., a motion capture sequence also has temporal constraints, i.e. each frame in closely related to the previous frames [BWP13]. This constraint may be included

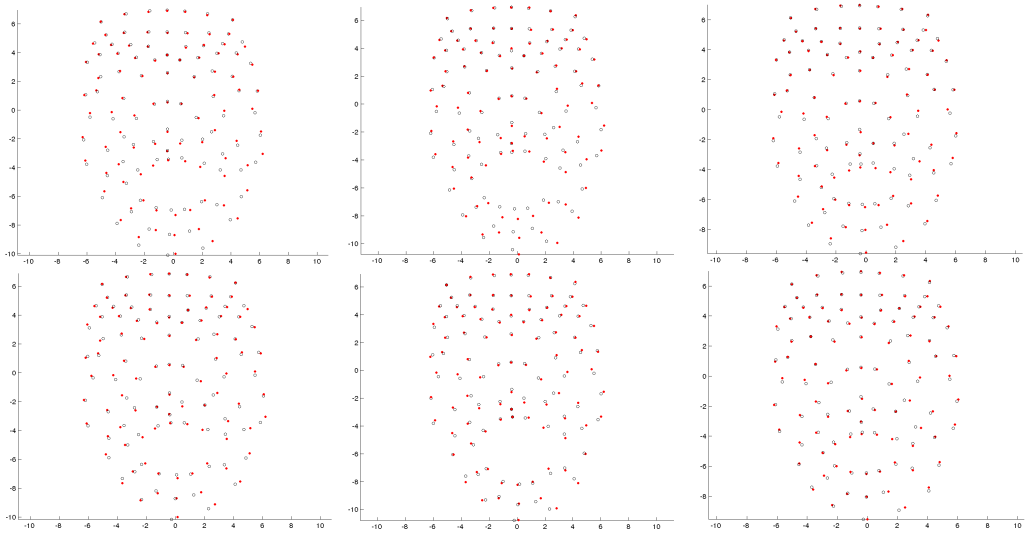


Figure 3-15: The red dots mark the position of the markers in the motion sequence while the black cirles correspond to the reconstructed position of the markers. The top results were produced using the non-negative least-squares solver while the bottom results were produced by the linear least-squares solver.

in the minimisation problem by adding a smoothness term, $\|\mathbf{w}^{t-2} - 2\mathbf{w}^{t-1} + \mathbf{w}^{t}\|_2^2$, where $\mathbf{w}$ is the vector of weights, and t is the frame number. However, the origin solution automatically obeys this smoothness constraint as the the input sequence was filmed at a relatively rate of 60 frames per second. Specifically, we compared the change in blendshape weights from frame to frame; the average difference is 0.007, which is hardy noticeable. The highest differences (of around0.3) appear during the opening and closing of the mouth; these transitions look natural at the original frame rate. We note that if the motion capture data was recorded at a lower frame rate, then the temporal constraints should improve the quality of the animation.

Next, we examined the effect of using a reduced set of key expressions in the solver; this leads to a smaller system of equations to be optimised. The reduced set contains 19 shapes included in the visual set of controls in the original blend shape model. The smaller set of shapes leads to a slight increase (of approximately 0.03 centimetres) in reconstruction error. There is no noticeable effect on the visual quality of the resulting animation, see Fig. 3-16. Thus the only notable advantage of using fewer blend shapes is the acceleration of the optimisation step. However, if the performance of the algorithm is the main objective, then a PCA basis is a better choice than a reduced non-orthogonal set of shapes.

We then attempt an opposite approach, i.e. we include extra bend shapes to account for the features that are not present in the original set. This approach was inspired by the research of Li et al., who introduced corrective shapes to systematically extend the initial set of shapes [LYYB13]. The authors start by using linear combinations of PCA principal components to construct new facial expressions. They then extract a number of samples that cannot be explained by the original set of principal components. An iterative procedure is used to produce corrective shapes that are orthogonal to the member of the original set, and that improve

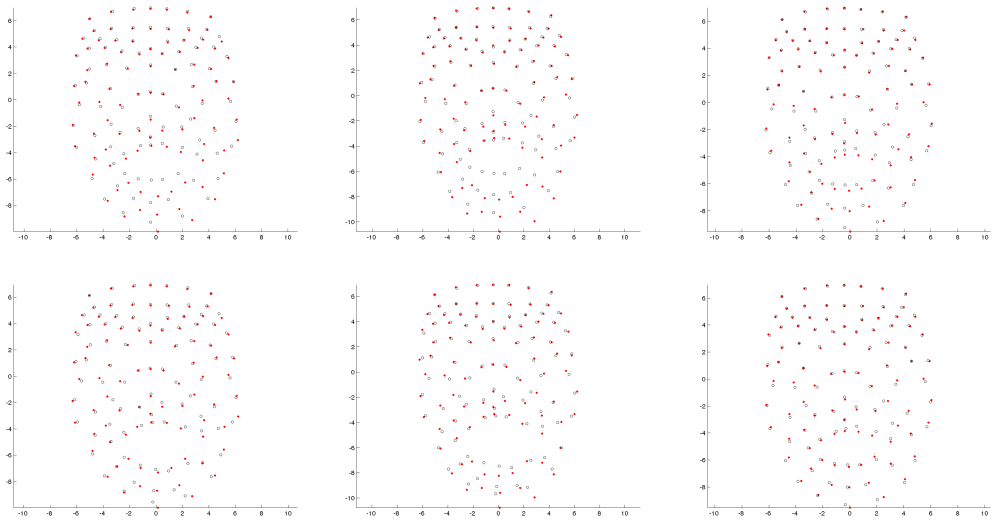


Figure 3-16: The reconstruction error for a model with 19 blend shapes. The top results were produced using the non-negative least-squares solver while the bottom results were produced by the linear least-squares solver. Colours as in Fig. 3-15.

the fitting accuracy. Unfortunately, such corrective method cannot be applied to a manually constructed set of blend shapes since it relies on vector orthogonality. Instead, we manually construct a set of new shapes that are more extreme than our original blend shapes; we then include these new shapes in the solver. The criteria for choosing the new expressions are, (a) the likelihood of that or similar expressions, and (b) the dissimilarity between the new and the existing expressions. The four extra blend shapes are shown in Fig. 3-17; they were constructed by increasing the weights on the original blendshapes beyond the predefined limit. Fig. 3-18

Figure 3-17: Additional blend shapes.

shows the activation of the extra blend shapes throughout the motion sequence. Both of the optimisation methods favour the first additional blendshape that corresponds to the smile in Fig. 3-17 while the other shapes are only used with low weights. Despite the extension to the set of shapes, the reconstruction error decreased by less then 0.01 centimetres for both methods. This is explained by the fact that the extra blenshapes are closely related to the original ones, as they are in fact a linear combination of the original blendshapes. A more system-

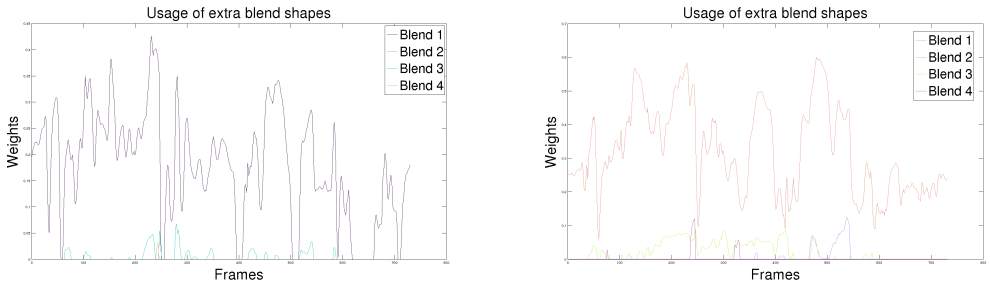


Figure 3-18: The weights associated with the extra blend shapes as a function of frames in the sequence. The left diagram corresponds to the constrained non-negative least-squares solver while the right one gives he results for the linear least-squares solver.

atic approach needs to be used to produce more diverse shapes. For example, an additional optimisation step may be included that randomly alters the existing blendshapes, and used a small subset of the altered shapes to find the extension that minimises the reconstruction error. In particular, we choose a set of new shapes that are constructed by adding a small random quantity to the average of the existing set. Then these shapes are included in the optimisation algorithm. The new shapes are stored if they reduce the reconstruction error, and new blend shapes are suggested until the error decreases below a chosen threshold. Using this iterative procedure, we are able to reduce the reconstruction error by about 10%; however, this approach is very computationally intensive, and the new shapes cannot be easily included in the existing model.

At this stage our best results still lack expressiveness and visual appeal. In the current pipeline, we first transform the motion sequence from the actor to Emily, and then solve for the blendshapes in the Emily domain. This approach relies on the transformation to produce valid motion in the Emily domain; however, the transformation is geometrical, and it does not guarantee to produce a realistic and natural-looking animation. Therefore, we test the opposite approach. We first find the thin plate spline transformation $\mathbf{T}$ from the actor's neutral expression to Emily's neutral expression. Then the inverse transformation $\mathbf{T^{-1}}$ is used to map the Emily blendshapes to the actor's domain. We thus have matching sets of sparse blendshapes in both domains, see Fig reffig:EtoR. The weights are calculated at each frame using the original captured sequence and the newly designed blendshapes in the actor's domain; these weights are then applied directly in the Emily domain.

Fig. 3-20 compares the results produced using the two antipode approaches. The new approach is able to better mimic the motion of the actor and it improves the appearance of the digital model, i.e. the new animation appears more natural and realistic. Note that the original method tries to reproduce the actor's performance using the transformed geometrical data while the new method relies more on the semantics associated with the blendshapes. In particular, we do not expect the actor and Emily expressions to match identically, for example one of them might be able to raise the eye-brows higher than the other. The new approach is better at respecting and enhancing the different features of the two faces. We note that our final animation was not always able to capture the slight smiles that are noticeable in the recorded

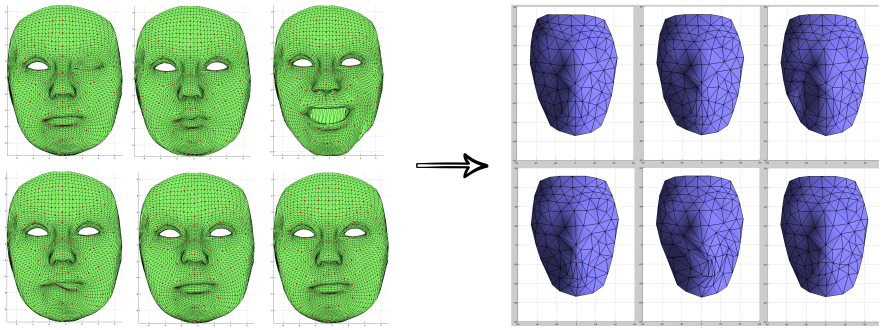


Figure 3-19: Use the inverse thin plate spline transform $\mathbf{T}^{-1}$ to produce a set of blendshapes in the actor's domain. The left images are the sparse Emily blendshapes (red dots) while the right images are the sparse actor's blendshapes (vertices of the mesh)

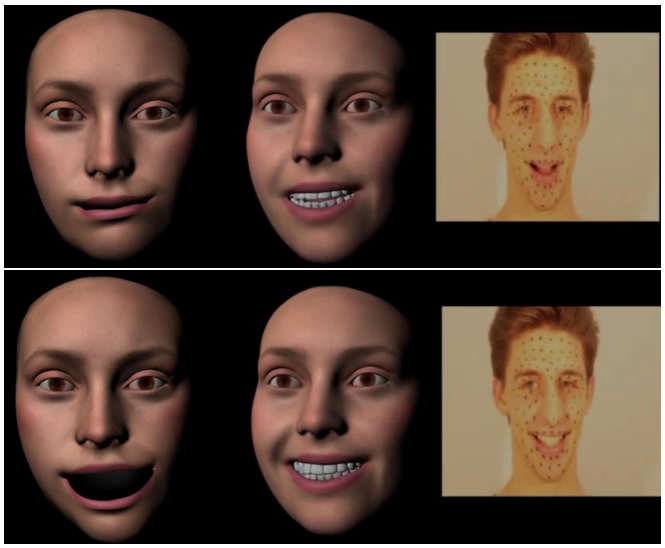


Figure 3-20: The comparison of on the left were produced using our original approach. The expressions in the middle where produced using the weights that were calculated in the actor's domain.

Figure 3-21: The comparison of neutral expressions.

sequence. This may be explained by the poor match of neutral expressions. Specifically, all of the methods that we used rely on the transformation between the neutral faces. However, the actor's neutral expression is more positive (i.e. the corners of the lips are raised) than that of Emily, see Fig. 3-21. This may be fixed by choosing a better matching neutral face from the actor's sequence, or by mimicking the actor's expression on Emily, and using that as her neutral. In our final results we use the second approach, and apply a additional small positive weight to the blendshapes that correspond to a smiling face, see Sec. 4.

There are a number of different techniques that could have been used to construct blendshapes in the actor's domain. Deng et al. proposed designing the blendshapes by hand [DCFN06]. In particular, the authors choose a number of characteristic frames in the motion capture sequence, and manually design a set of shapes that perceptually match the expressions observed in these frames. They then form correspondences between the PCA coefficients of motion capture frames and the weights of the blendshapes. Finally, Radial Basis Functions are used to map the new frames in the sequence onto the three-dimensional model to produce the animation. The decision not to use this approach was made because of the extensive manual effort required to design the set of initial correspondences between the chosen frames and the three-dimensional mesh. For the same reason we choose not to use a muscle actuation basis, proposed by Choe and Ko [CK05]. The design of a muscle actuation basis is based on the human facial anatomy, and is driven by the actuation of different facial muscles.

3.8 Skin Rendering

Our objective in skin rendering is to generate a face that would be indistinguishable from a real one. In our case, we have have taken a 3D scan of a subject and our aim is to improve the realism when rendering it. For this task we will mainly look at the techniques presented by Hertzmann et al [HJO*01] and Graham et al [GTB*13].

Before we begin, lets explain the Image Analogies framework [HJO*01] in more detail. Given three images A, A' and B, where A is an unfiltered example, A' is a filtered example, and B is an input image, the algorithm will generate an output image B' such that B' relates in the same way to B, as A' does to A. A k-d tree for an Approximate Nearest-Neighbour Search (ANN) is built using a feature vector from a neighbour pixel p in A and A'. The closest match for a neighbourhood in pixel q in B and B' is located in the tree. A detailed description of the

Algorithm 2: Image Analogies

Data: A unfiltered example, A' filtered example, B unfiltered source, L number of levels, k coherence parameter, t neighbourhood size.

Result: B' filtered source image.

Compute Gaussian pyramids for A, A' and B;

Compute features for A, A' and B;

Build k-d tree for $\{A, A'\}$;

for *l = 0 to L* **do**

  for *each pixel* $q \in B'_l$ **do**

    p_{app} = ANN search of q neighbourhood from $\{B, B'\}$;

    $r^* = \arg\min_{r \in N(q)} \|F_l(s(r) + q - r) - F_l(q)\|^2$;

    $p_{coh} = s(r^*) + (q - r^*)$;

    $d_{app} = \left\|F_l(p_{app}) - F_l(q)\right\|^2$;

    $d_{coh} = \|F_l(p_{coh}) - F_l(q)\|^2$;

    if $d_{coh} < d_{app}(1 + k2^{l-L})$ **then**

      $p = p_{coh}$

    else

      $p = p_{app}$

    end

    $B'_l(q) = A'_l(p)$;

    $s_l(q) = p$;

  end

end

return B'_L

algorithm in pseudo code is shown in Algorithm 2, where F is computed a weighted distance over the feature vectors using a Gaussian kernel and s is a data structure such that $s(q) = p$. Based on Ashikhmin's work, a match that is coherent to what has been already synthesized is computed. These two candidates are weighted and the best one is chosen. The whole process is carried in a multiresolution pyramid, as shown in Figure 3-22, where l indicates the current level, in essence this means that the neighbourhoods also include the previous level in the search. We found three Image Analogies implementations available [ImAa, ImAb, ImAc]. The first one is a simple single threaded implementation, the second one was done with CUDA, however the author's single threaded code produced overall better results.

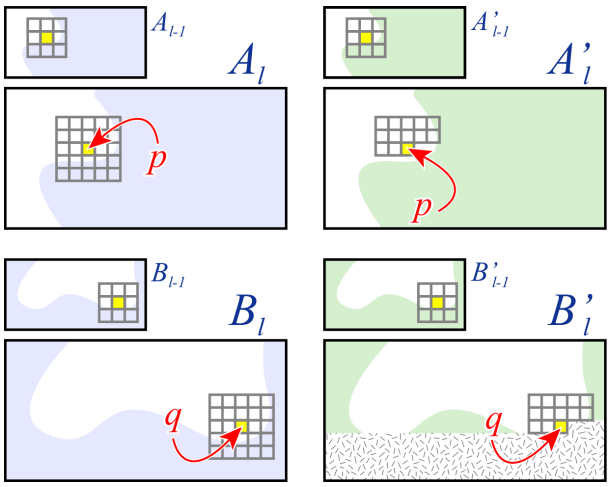


Figure 3-22: Neighbourhood matching for the Image Analogies framework, image taken from [Ash01].

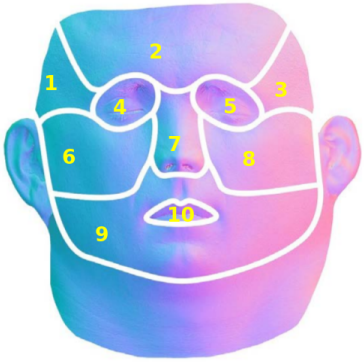


Figure 3-23: Texture segmentation in common coloured areas, image taken from [GTB*13].

As a first step, we tried to reproduce the results for bump mapping quality increase by Graham et al [GTB*13], results are shown in Figure 3-24. The authors add an extra parameter $\alpha \in \{0, \ldots, 1\}$ to control Hertzmann's image synthesis process. To be more precise, a match between A and $\{B, B'\}$ will be weighted by $1 - \alpha$, and a match between A' and $\{B, B'\}$ will be weighted by α. The logic behind this addition is to encourage more details of A' to be included in B'. The modifications required to include this addition begin by building two separated k-d trees for A and A', when choosing the best match both distances will be weighted as explained above and the smaller one will be chosen. Also in the coherence match two searches will be done and weighted accordingly, and the final pixel will be chosen without further adjustments.

We also tried applying the Image Analogies filter to generate increased quality textures, which is in essence a deblurring filter. The idea is to improve a low quality texture from a 3D scan using pictures of the texture taken at a closer range. To achieve this we took a close up high quality sample A', to generate A, the sample A' was blurred using a Gaussian kernel until it look qualitative similar to the 3D scan texture B, with this three inputs we generated a picture B' of higher quality. Since faces have differentiated areas, this process was done separately for each relevant section in the face texture image, the generated patches are stitched together using linear interpolation. The texture segmentation is shown in Figure 3-23 and results are

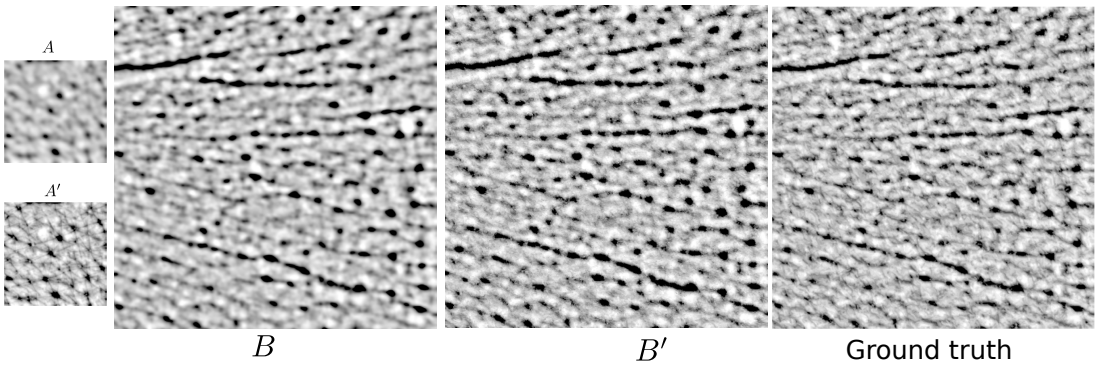


Figure 3-24: Bum map deblurring, A, A', B and ground truth images images taken from [GTB*13].

shown in Figure 3-25.

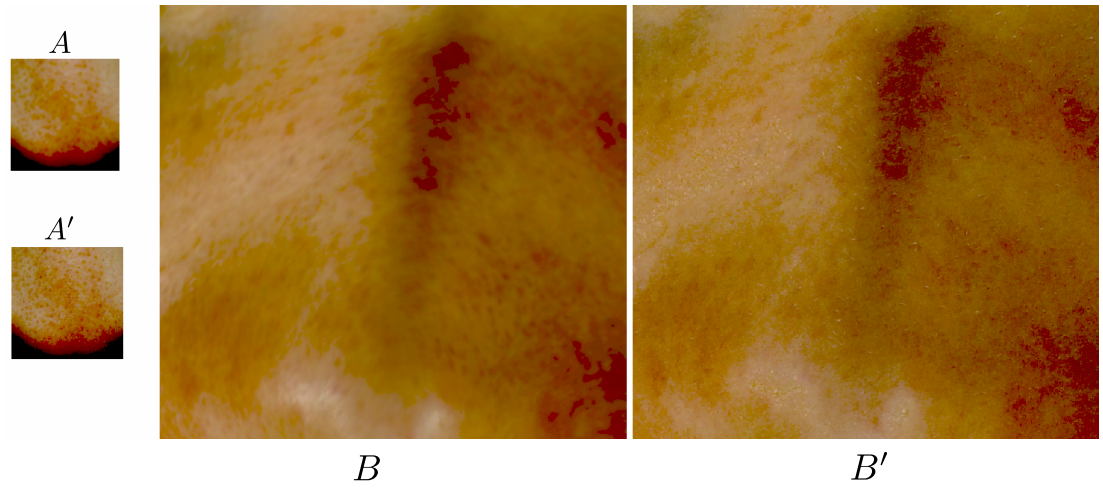


Figure 3-25: Texture quality increase using image analogies, the texture is false-coloured to highlight the differences.

Another option for increasing the quality of the texture is to apply image super-resolution techniques, we used Jianchao et al [YWHM10] work for this purpose. The idea is that by doubling the resolution of the original texture, yet avoiding blur by adding information from a dictionary of high resolution images, the final rendering quality of the face will increase. Results for this approach are shown in Figure 3-26.

Generating normals maps using Image Analogies is another interesting area, as it could provide an alternative to the costly standard capture methods. For this we tried to generate a normal maps from albedo images and from bump maps generated from the previous 3D scan texture, results are shown in Figure 3-28. To create the bump maps, the textures were transformed to gray scale and a histogram equalization was applied. In order to improve the quality of the bump maps, we also applied the Image Analogies filter to them using a known good bump map as a filtered example, results are shown in Figure 3-29.

Figure 3-26: Super resolution example, left original texture, left-centre face rendered with original texture, right-centre super-resolution texture, right face rendered with super-resolution texture, original data from [Faca].

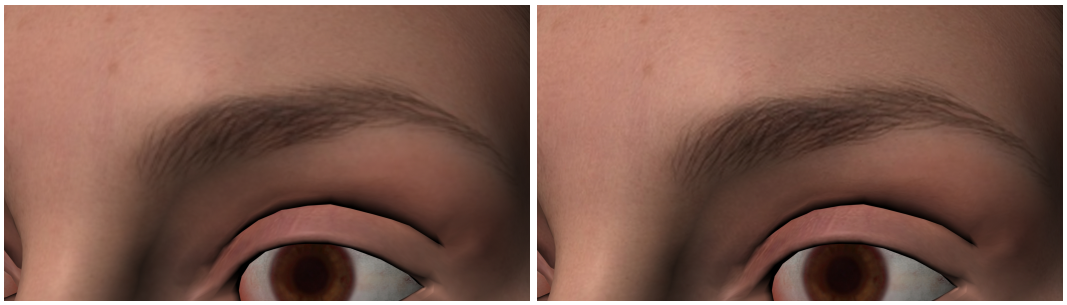


Figure 3-27: Close up of texture super-resolution for Emily, (left) original texture, (right) synthesized texture with higher resolution.

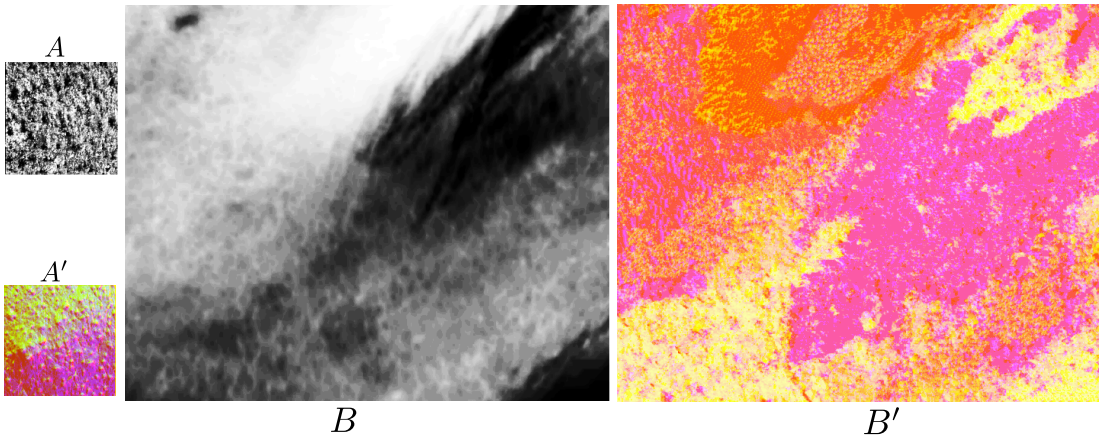


Figure 3-28: Normal synthesis from albedo image.

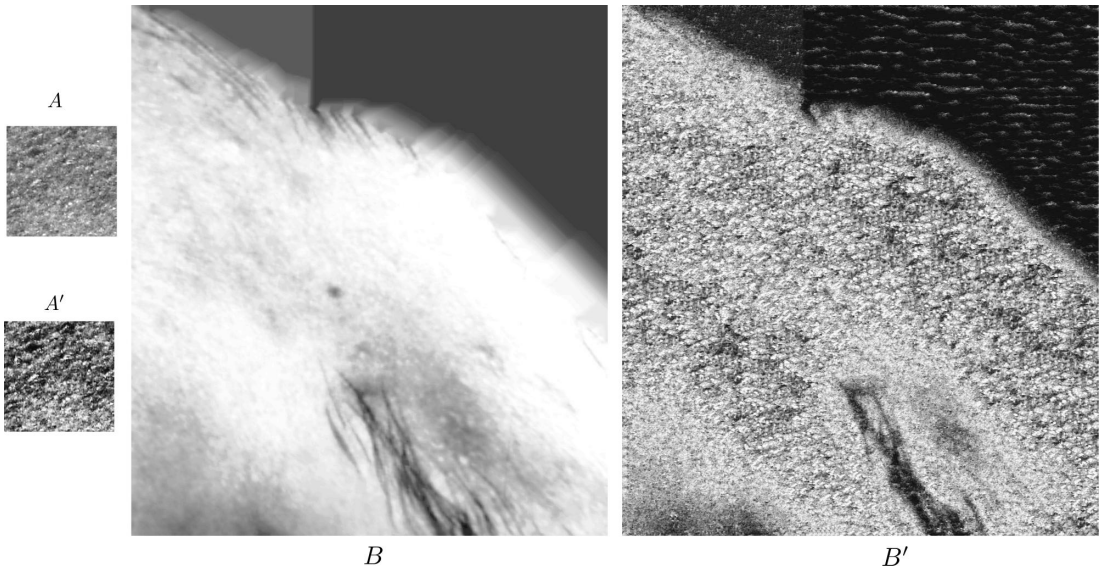
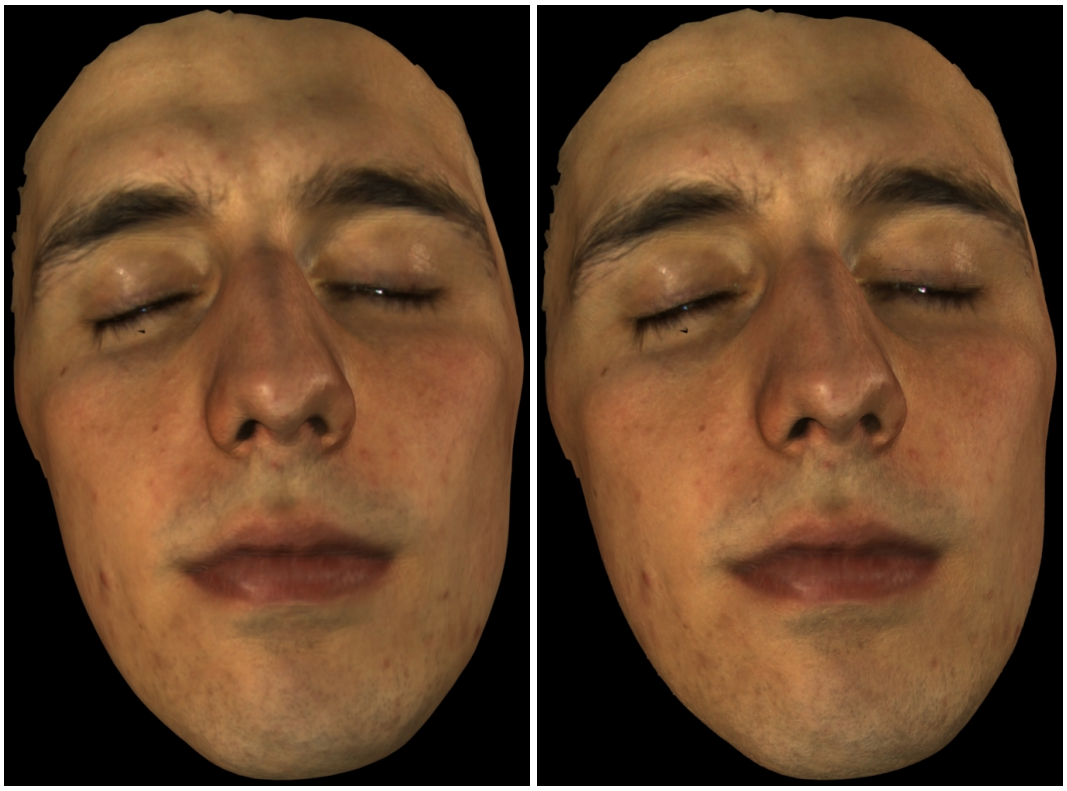


Figure 3-29: Improving the quality of a bump map from a gray scale texture.



(a) Render with the original texture.

(b) Render with the original texture and bump mapping.

Figure 3-30: The advantages of using a bump map..

The advantages of using a bump mapping are shown in Figure 3-30. In this example, a render with the original texture is shown in Figure 3-30a. The original RGB texture was converted to a gray scale image and applied directly as a bump map texture, results are shown in Figure 3-30b, note how the addition of high frequency imperfections adds extra realism to the image.

Chapter 4

Implementation details

Talk about Maya plugins and any other implementation stuff worth talking about. ..... Introduce the topic

We used the rigs for Maya that are provided for free by [Facb], the first thing to fix is to reset the paths for the textures in all the materials in the Hipershade window in Maya®. The next step was to write plugin to load the weights from Matlab and set keyframes for each of them; to do this our plugin would create a new Maya®command that could be easily executed. The work flow for writing the code would be to first do the action in Maya manually if possible, see what MEL commands where executed, write them in the plugin and if possible rewrite them as pure C++ code instead of MEL commands. The file format for the weights is one weight per line in frame order, as shown in Figure 4-1, where *N* is the number of weights and *F* is the number of frames. Once we have the weights loaded, the connections between the blenshape object weights and its outputs have to be broken, this is done in order to be able to set the keyframes, otherwise the setKeyframe function would not work, an extract of the code is shown in Listing 4.1. Additionally, we program our code to comply with Maya®undo command policy, which means using the *DagModifier* class or manually undoing the actions, however in the final stages of our implementation it was faster to reload the scene that to undo de command.

$$\begin{bmatrix} w_{00} \\ w_{10} \\ \vdots \\ w_{ij} \\ \vdots \\ w_{NF} \end{bmatrix}$$

Figure 4-1: File format for weights import and export.

Once the weights were added, we noticed that the teeth and tongue of the Emily rig would

Listing 4.1: Breaking the weights connections and setting keyframes.

```
// Break connections.
for (unsigned int i = 0; i < (unsigned int)(numWeights); i++){
                cmd = "disconnectAttr shapesBS_";
                cmd = cmd + names[i];
                cmd = cmd + ".output shapesBS.";
                cmd = cmd + names[i];
                dgMod.commandToExecute(cmd);
}

// Set key frames
for (unsigned int j = 0; j < weights.at(i).size(); j++){
                cmd = "setAttr shapesBS.weight[";
                cmd = cmd + j;
                cmd = cmd + "] ";
                cmd = cmd + weights.at(i).at(j);
                dgMod.commandToExecute(cmd);

                cmd = "setKeyframe { \"shapesBS.w[";
                cmd = cmd + j;
                cmd = cmd + "]\" }";
                dgMod.commandToExecute(cmd);
        }
}
```

Listing 4.2: Teeth and tongue control based on relevant weights.

```
cmd = "setAttr con_jaw_c.translateY ";
cmd += -3.2 * (weights.at(i).at(58) + weights.at(i).at(55)) + 1;
dgMod.commandToExecute(cmd);
cmd = "setKeyframe  \"con_jaw_c.translateY\"";
dgMod.commandToExecute(cmd);
```

not move. Both meshes were actually regulated by the position of a control object in the scene, so the solution was to translate the object by a factor of the mean of all the weights involved in the mouth control, as shown in Listing 4.2.

The next step was to include head rotation and translation back into the animation. In order to achieve this, we saved the inverse rotation from the Procustes analisys that was performed on Section 3.6.1, into a file with the same format as shown in Figure 4-1. We then read the data into Maya®and applied the transformation for each frame; for the translation we did a equivalent procedure with a translation file. Note that Matlab matrices are column-wise while Maya®matrices are row-wise, so we read them taking into account this consideration. The matrix multiplication order also plays in important role here, as shown in Equation 4.1, to achieve the correct result we need to undo the translation first and rotate the result.

$$\mathbf{x}_{new} = R\mathbf{x} + \mathbf{t} \;\rightarrow R^{-1}(\mathbf{x}_{new} - \mathbf{t}) = \mathbf{x}, \tag{4.1}$$

where $\mathbf{x}$ is the original point, $\mathbf{x}_{new}$ is the transformed point, R is the rotation matrix and $\mathbf{t}$

is a translation vector. We implemented this transformation in Maya®using the "xform -m" command, so we first apply the translation and then apply the rotation with an extra parameter "-r" so that both matrices get multiplied in the desired order.

Having eye movement is quite important to achieve realism in the animation, for this purpose we tracked the pupils world positions in the input frame sequence. This positions were saved into separate files for each eye using the format shown in Figure 4-1. The positions were loaded into Maya®and the eye control object was translated using the offsets with the respect to the first frame, which was assumed to be looking straight. With this first approach each eye can move independently which leads to unsatisfactory results, an initial solution to this problem was to move both eyes using the mean offset for each frame. Nevertheless, this would introduce considerable error when one of the eyes winks, as the tracking becomes completely unreliable for the winking eye which in turn drags the other eye. Our solution involves interpolating between the two offsets using variable factors, the idea is to have both eyes moving together, rapidly ignore if one eye goes off and while sustaining smooth movements. The final offset d_f will be computed as $d_f = \alpha d_l + \beta d_r$, where α and β are the interpolation factors, d_l and d_r are the offsets for the left and right eye, respectively. Figure 4-2 shows the α_x values along the x direction, where t is the change threshold and l is the maximum offset limit. Initially α_x is 0.5, however if the offset goes beyond a threshold t, α will decrease linearly until it reaches the limit l. The same criteria will be applied in the y direction, and the value for that eye will be $\alpha = \min\{\alpha_x, \alpha_y\}$. The β factor will be computed as $\beta = 1 - \alpha$, however, if the right eye is the one surpassing the threshold t, then the whole process will be reversed. Additionally, if both eyes surpass the threshold t, α and β will be 0.5, this will allow the eyes to move together beyond the threshold if needed.

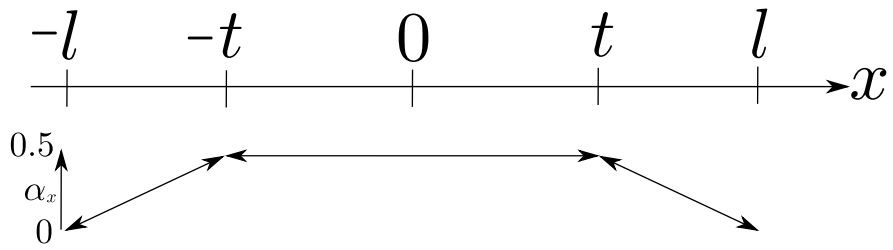


Figure 4-2: Offset interpolation for eye movements in Maya®.

Chapter 5

Results

Chapter 6

Conclusions

Bibliography

[ACP03] Allen B., Curless B., Popović Z.: The space of human body shapes: Reconstruction and parameterization from range scans. *ACM Trans. Graph. 22*, 3 (July 2003), 587–594.

[ARL*09] Alexander O., Rogers M., Lambeth W., Chiang M., 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.

[ARV07] Amberg B., Romdhani S., Vetter T.: Optimal step nonrigid icp algorithms for surface registration. cited By 37.

[Ash01] Ashikhmin M.: Synthesizing natural textures. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics* (New York, NY, USA, 2001), I3D '01, ACM, pp. 217–226.

[BA06] Bose N., Ahuja N.: Superresolution and noise filtering using moving least squares. *Image Processing, IEEE Transactions on 15*, 8 (Aug 2006), 2239–2248.

[BB14] Beeler T., Bradley D.: Rigid stabilization of facial expressions. *ACM Transactions on Graphics (TOG)* (2014).

[BBA*07] Bickel B., Botsch M., Angst R., Matusik W., Otaduy M., Pfister H., Gross M.: Multi-scale capture of facial geometry and motion. *ACM Trans. Graph. 26*, 3 (July 2007).

[Ben05] Bennett D.: The faces of "the polar express". In *ACM SIGGRAPH 2005 Courses* (New York, NY, USA, 2005), SIGGRAPH '05, ACM.

[BRM12] Baltrušaitis T., Robinson P., Morency L.-P.: 3d constrained local model for rigid and non-rigid facial tracking. pp. 2610–2617. cited By 14.

[BV99] Blanz V., 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.

[BWP13] Bouaziz S., Wang Y., Pauly M.: Online modeling for realtime facial animation. *ACM Trans. Graph. 32*, 4 (July 2013), 40:1–40:10.

[Cal] Camera Calibration Toolbox for Matlab. http://www.vision.caltech.edu/bouguetj/calib_doc/. Accessed: 2015-02-10.

[CB05] Chappalli M., Bose N.: Simultaneous noise filtering and super-resolution with second-generation wavelets. *Signal Processing Letters, IEEE 12*, 11 (Nov 2005), 772–775.

[CK05] Choe B., 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.

[CLK01] Choe B., Lee H., Ko H.-S.: Performance-driven muscle-based facial animation. *Journal of Visualization and Computer Animation 12*, 2 (2001), 67–79. cited By 55.

[DCFN06] Deng Z., Chiang P.-Y., Fox P., 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.

[DWd*08] Donner C., Weyrich T., d'Eon E., Ramamoorthi R., Rusinkiewicz S.: A layered, heterogeneous reflectance model for acquiring and rendering human skin. *ACM Trans. Graph. 27*, 5 (Dec. 2008), 140:1–140:12.

[Faca] Faceware Tech. http://facewaretech.com/. Accessed: 2015-05-19.

[Facb] Faceware Tech: Free Rigs. http://facewaretech.com/learn/training-assets/. Accessed: 2015-05-19.

[GGW*98] Guenter B., Grimm C., Wood D., Malvar H., Pighin F.: Making faces. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1998), SIGGRAPH '98, ACM, pp. 55–66.

[GHDS03] Grinspun E., Hirani A. N., Desbrun M., Schröder P.: Discrete shells. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2003), SCA '03, Eurographics Association, pp. 62–67.

[Gow75] Gower J.: Generalized procrustes analysis. *Psychometrika 40*, 1 (1975).

[GTB*13] Graham P., Tunwattanapong B., Busch J., Yu X., Jones A., Debevec P., Ghosh A.: Measurement-based synthesis of facial microgeometry. *Computer Graphics Forum 32*, 2pt3 (2013), 335–344.

[GVWT13] Garrido P., Valgaert L., Wu C., Theobalt C.: Reconstructing detailed dynamic face geometry from monocular video. *ACM Trans. Graph. 32*, 6 (Nov. 2013), 158:1–158:10.

[HJO*01] Hertzmann A., Jacobs C. E., Oliver N., Curless B., Salesin D. H.: Image analogies. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2001), SIGGRAPH '01, ACM, pp. 327–340.

[HMD06] Humblot F., Mohammad-Djafari A.: Super-resolution using hidden markov model and bayesian detection estimation framework. *EURASIP Journal on Applied Signal Processing 10* (2006), ID.

[IGAJG15] Iglesias-Guitian J. A., Aliaga C., Jarabo A., Gutierrez D.: A biophysically-based model of the optical properties of skin aging. *Computer Graphics Forum (EUROGRAPHICS 2015) To Appear* (2015).

[ImAa] Image Analogies: C++ single threaded code. http://vis.berkeley.edu/courses/cs294-69-fa11/wiki/index.php/FP-JeffDonahue. Accessed: 2015-05-19.

[ImAb] Image Analogies: CUDA code. https://sites.google.com/a/college.harvard.edu/yaoyu-imageanalogies/parallel. Accessed: 2015-05-19.

[ImAc] Image Analogies: Hertzmann's code. http://www.mrl.nyu.edu/projects/image-analogies/. Accessed: 2015-05-19.

[JF09] Ji H., Fermuller C.: Robust wavelet-based super-resolution reconstruction: Theory and algorithm. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 31*, 4 (April 2009), 649–660.

[JSB*10] Jimenez J., Scully T., Barbosa N., Donner C., Alvarez X., Vieira T., Matts P., Orvalho V., Gutierrez D., Weyrich T.: A practical appearance model for dynamic facial color. In *ACM SIGGRAPH Asia 2010 Papers* (New York, NY, USA, 2010), SIGGRAPH ASIA '10, ACM, pp. 141:1–141:10.

[JTDP03] Joshi P., Tien W. C., Desbrun M., Pighin F.: Learning controls for blend shape based realistic 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. 187–192.

[LYYB13] Li H., Yu J., Ye Y., Bregler C.: Realtime facial animation with on-the-fly correctives. *ACM Trans. Graph. 32*, 4 (July 2013), 42:1–42:10.

[MAT13] MATLAB: *version 8.1.0.604 (R2013a)*. The MathWorks Inc., Natick, Massachusetts, 2013.

[NN01] Noh J.-y., 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.

[PHL*98] Pighin F., Hecker J., Lischinski D., Szeliski R., Salesin D. H.: Synthesizing realistic facial expressions from photographs. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1998), SIGGRAPH '98, ACM, pp. 75–84.

[PKC*03] Pyun H., Kim Y., Chae W., Kang H. W., 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.

[PSS02] Pighin F., Szeliski R., Salesin D.: Modeling and animating realistic faces from images. *International Journal of Computer Vision 50*, 2 (2002), 143–169. cited By 48.

[Sch05] Schmidt M.: Least squares optimization with l1-norm regularization. *CS542B Project Report* (2005).

[Sco] Scopus. http://www.scopus.com/. Accessed: 2015-05-21.

[SP04] Sumner R. W., Popović J.: Deformation transfer for triangle meshes. *ACM Trans. Graph. 23*, 3 (Aug. 2004), 399–405.

[SS91] Sibson R., Stone G.: Computation of thin-plate splines. *SIAM J. Sci. Stat. Comput. 12*, 6 (Sept. 1991), 1304–1313.

[ST94] Shi J., Tomasi C.: Good features to track. *IEEE Conference on Computer Vision and Pattern Recognition* (1994), 593–600.

[Sur] Surrey Audio-Visual Expressed Emotion (SAVEE) Database. http://personal.ee.surrey.ac.uk/Personal/P.Jackson/SAVEE/Database.html. Accessed: 2015-02-20.

[TFM07] Takeda H., Farsiu S., Milanfar P.: Kernel regression for image processing and reconstruction. *Image Processing, IEEE Transactions on 16*, 2 (Feb 2007), 349–366.

[TH84] Tsai R., Huang T. S.: Multiframe image restoration and registration. *Advances in computer vision and Image Processing 1*, 2 (1984), 317–339.

[TK95] Tom B., Katsaggelos A.: Reconstruction of a high-resolution image by simultaneous registration, restoration, and interpolation of low-resolution images. In *Image Processing, 1995. Proceedings., International Conference on* (Oct 1995), vol. 2, pp. 539–542 vol.2.

[TM10] Tian J., Ma K.-K.: Stochastic super-resolution image reconstruction. *Journal of Visual Communication and Image Representation 21*, 3 (2010), 232 – 244.

[TM11] Tian J., Ma K.-K.: A survey on super-resolution imaging. *Signal, Image and Video Processing 5*, 3 (2011), 329–342.

[VBPP05] Vlasic D., Brand M., Pfister H., Popović J.: Face transfer with multilinear models. *ACM Trans. Graph. 24*, 3 (July 2005), 426–433.

[VF08] Vedaldi A., Fulkerson B.: VLFeat: An open and portable library of computer vision algorithms. http://www.vlfeat.org/, 2008.

[Vic] Vicon. http://www.vicon.com/System/Markers. Accessed: 2015-05-20.

[WHL*04] Wang Y., Huang X., Lee C.-S., Zhang S., Li Z., Samaras D., Metaxas D., Elgammal A., Huang P.: High resolution acquisition, learning and transfer of dynamic 3-d facial expressions. *Computer Graphics Forum 23*, 3 SPEC. ISS. (2004), 677–686. cited By 60.

[WL00] Wei L.-Y., Levoy M.: Fast texture synthesis using tree-structured vector quantization. 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. 479–488.

[WLGP09] Weise T., Li H., Gool L. V., Pauly M.: Face/off: Live facial puppetry. *Proceedings of the 2009 ACM* (2009).

[WMP*06] Weyrich T., Matusik W., Pfister H., Bickel B., Donner C., Tu C., McAndless J., Lee J., Ngan A., Jensen H. W., Gross M.: Analysis of human faces using a measurement-based skin reflectance model. In *ACM SIGGRAPH 2006 Papers* (New York, NY, USA, 2006), SIGGRAPH '06, ACM, pp. 1013–1024.

[XCLT14] Xu F., Chai J., Liu Y., Tong X.: Controllable high-fidelity facial performance transfer. *ACM Trans. Graph. 33*, 4 (July 2014), 42:1–42:11.

[YWHM10] Yang J., Wright J., Huang T., Ma Y.: Image super-resolution via sparse representation. *Image Processing, IEEE Transactions on 19*, 11 (Nov 2010), 2861–2873.