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

Contents

Chapter 3

Methodology

This section presents the methodology behind our proposed facial performance capture system and discusses some of the technical challenges we faced in its implementation.

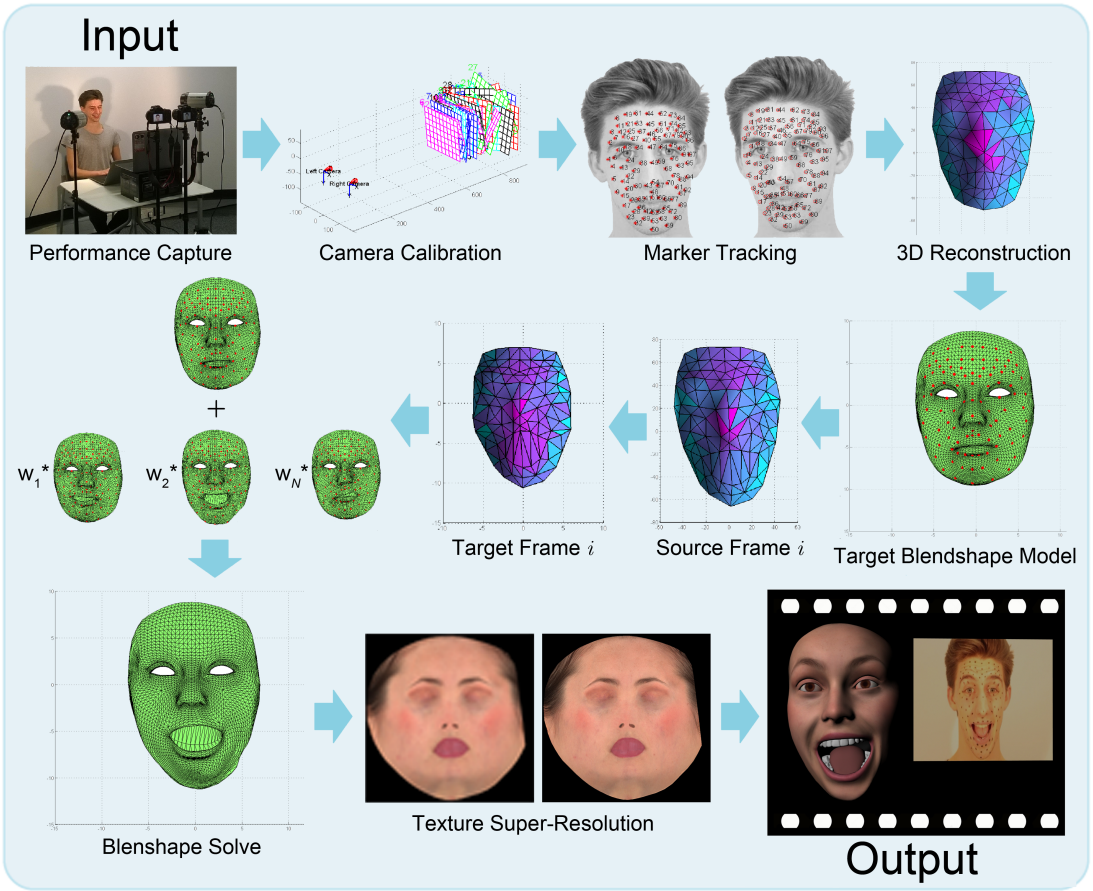


Figure 3-1: A schematic of the proposed pipeline for the performance capture system.

3.1 Data Capture

The first task was to gather some initial input data for the system by recording the facial performance of an actor in the form of a video sequence. Two DSLR cameras were positioned close to the actor's face in a stereo configuration in order to capture the actor's facial performance, as shown in Figure 3-2. A short sequence, consisting of the actor pulling a number of facial expressions and then talking, was recorded on video with a resolution of 640×480 at 60fps. The video streams were roughly synchronised afterwards during post-processing in the computer by aligning the two separate audio signals. A relatively low image resolution was chosen in order to try to limit the amount of video data we had to deal with, especially when recording sequences at higher frame-rates. The cameras were fixed on stands to prevent any camera-shake during the performance capture session and calibration process - since it is important that the camera positions remain static for accurate 3D reconstruction. Positioning the cameras close to one another, at approximately 10cm horizontal separation and at a distance of about 1m 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 actor's face; a good rule of thumb is for the cameras and subject to subtend an angle of at least 5 degrees.

Figure 3-2: The stereo camera setup used to capture the facial performance.

Markers used to track the facial performance were placed on the actor's face in the configuration shown in Figure 3-3. 97 markers were used in total, and were drawn onto the actor's face with a make-up 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 3-4, although they use fewer markers with just

60 in total. In hindsight, using slightly fewer number of markers in our system might have been preferable, as consistently detecting and tracking a large number of markers (and preserving the labelling order of the markers) became quite a time-consuming and labour intensive process, prone to error. 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, or perhaps even using retro-reflective markers as in many commercial motion capture systems [Vic].

Although the markers were placed all over the face to capture full facial performance, we tried to ensure that specific markers were placed in positions on the face which exhibit large non-rigid deformation, such as around the eyes, around the mouth and along the eye-brows, in order capture the full expressiveness of the face. In a future implementation, it would be useful to track the rigid head motion by tracking the positions of a number points on the head of the actor, for example by making the actor wear a skull cap with markers. Another useful addition would be to somehow track the eyelids so we can capture subtle eye movements and blinks. Also, since we only track points on the outside of the mouth, we do not gather information about the lip movement, such as expressions involving pursing or funnelling of the lips, etc.

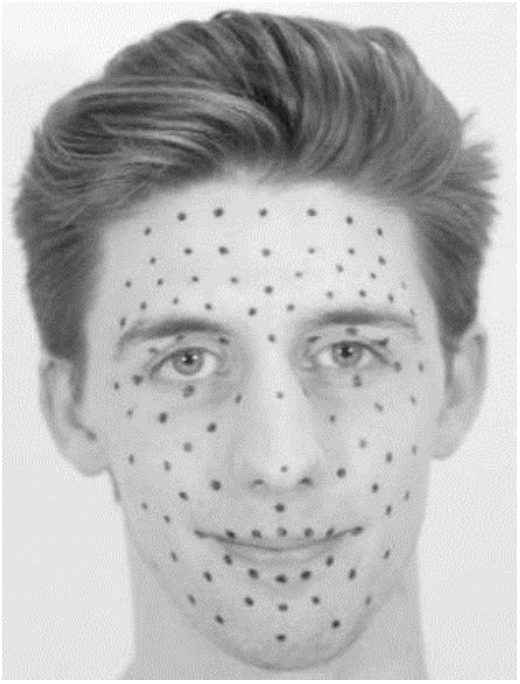


Figure 3-3: The configuration of markers used for tracking the facial performance of the actor.

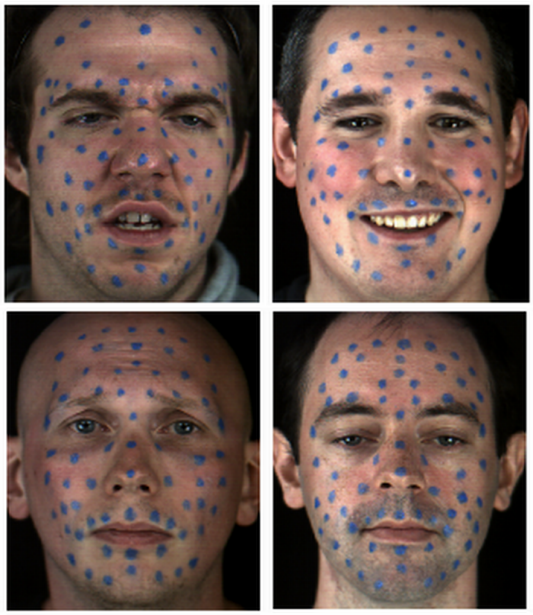


Figure 3-4: Images from the Surrey Audio-Visual Expressed Emotion (SAVEE) Database [Sur].

3.2 Camera Calibration

Once we had obtained a video sequence of the actor's face, the next stage was to calibrate the stereo camera system in order to obtain accurate 3D coordinates of the markers on the face.

Camera calibration is name given to the process of recovering the camera projection matrix **P** from images of a controlled scene. We assume the pinhole camera model and under perspective projection the map between the three-dimensional world coordinates of a point $(X, Y, Z)^T$ and its two-dimensional image pixel coordinates $(u, v)^T$ is a linear mapping in homogeneous coordinates represented by the 3×4 projection matrix:

$$\begin{pmatrix} su \\ sv \\ s \end{pmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{3.1}$$

To obtain an estimate of the camera projection matrices for both cameras, we use a checkerboard pattern of known dimensions held in a number of different positions and orientations throughout the scene, in view of both cameras simultaneously, as shown in Figure 3-5. To perform the calibration, we made use of the Camera Calibration Toolbox for Matlab [Cal] provided by Jean-Yves Bouguet, which automatically detects the grid corners in each image and solves for the intrinsic camera parameters (focal length, principle point, and distortion coefficients) and the extrinsic parameters (rigid rotation and translation). It was important to ensure that the checkboard remained within the frame of both cameras during the entire calibration process, and that the camera positions did not move between the performance capture session and the calibration procedure.

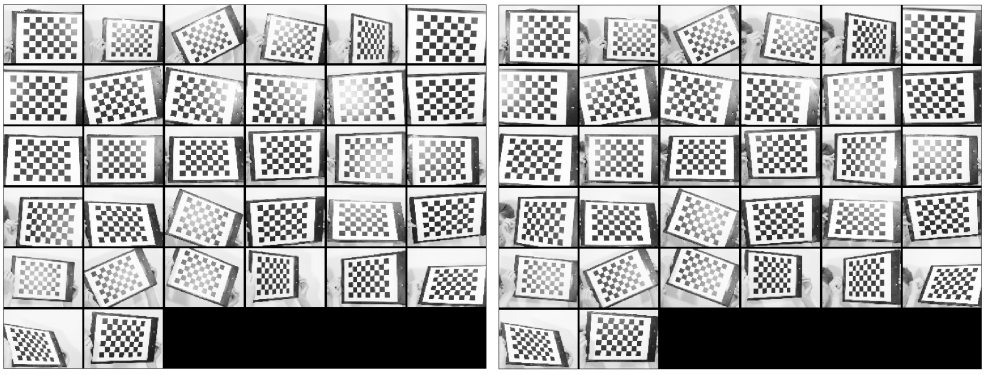


Figure 3-5: A checkerboard pattern was used to calibrate the pair of stereo cameras. The checkerboard had to be visible from both camera viewpoints simultaneously.

As a result of the stereo calibration, we obtain an estimate of the intrinsic matrices **K**, $\mathbf{K}'$ and external parameters **R**, **t** of the two cameras, from which we can compute the camera projection matrices **P** and $\mathbf{P}'$ for the left and right cameras respectively using the equation for perspective projection:

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

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

where the left camera is assumed to be centred at the origin and aligned with the world co-ordinate axes, i.e. its projection matrix is $\mathbf{P} = \mathbf{K}[\mathbf{I}|\mathbf{0}]$, and the right camera projection matrix is expressed as $\mathbf{P}' = \mathbf{K}'[\mathbf{R}|\mathbf{t}]$.

The resulting extrinsic parameters, describing the rotation $\mathbf{R}$ and translation $\mathbf{t}$ between the two camera views, are visualised in Figure 3-6. The positions of the checkerboard used for calibration are also displayed.

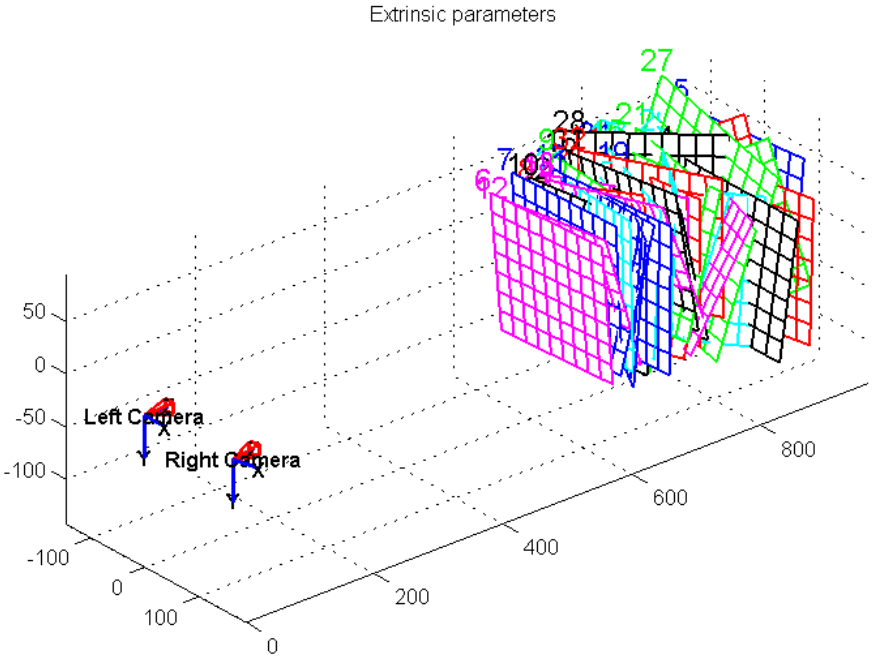


Figure 3-6: The resulting extrinsic parameters of the calibrated stereo camera setup.

From the estimated projection matrices, we can compute an estimate of the fundamental matrix $\mathbf{F}$ relating the two camera views. The fundamental matrix encapsulates the intrinsic epipolar geometry algebraically and is defined by the following equation:

$$\mathbf{u}'^T \mathbf{F} \mathbf{u} = 0$$

$$(u'v'1)\begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix}\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = 0$$

where $\mathbf{u} = (u, v, 1)^T$ and $\mathbf{u}' = (u', v', 1)^T$ are the 2D pixel coordinates of the same 3D world point imaged in the left and right views respectively. The fundamental matrix $\mathbf{F}$ is a 3×3 matrix, so has 9 elements but only has 7 DoF. This is because it has arbitrary scale, leaving 8 DoF, and is of rank 2 and singular, i.e. $\det[\mathbf{F}] = 0$, leaving 7 DoF. Using the fundamental matrix relation, the equation of the epipolar line in the right image $\mathbf{l}'$ corresponding to the point $\mathbf{u}$ in the left image, on which the feature point must lie, is therefore given by the following expression

$$\mathbf{l}' = \mathbf{F}\mathbf{u} \tag{3.2}$$

Similarly, the epipolar line $\mathbf{l}$ in the left image is given by

$$\mathbf{l} = \mathbf{F}^T\mathbf{u}'$$

The left and right epipoles $\mathbf{e}$ and $\mathbf{e}'$ are defined as the point in each image which is common to all the epipolar lines and are given by the null spaces of $\mathbf{F}$ and $\mathbf{F}^T$ respectively

$$\mathbf{F}\mathbf{e} = \mathbf{0} \qquad \mathbf{F}^T\mathbf{e}' = \mathbf{0}$$

With the epipolar geometry fully defined, we can then rectify the stereo images for every frame of the captured image sequence so that the eipolar lines are horizontal. This reduces the stereo matching problem to a 1D search problem, as the same 3D world point imaged in both cameras is constrained to lie on the same pixel row in each pair of images. To rectify each pair of stereo images we implemented a function in Matlab `StereoRectify.m`, based on the function `rectify_stereo_pair.m` provided in the Camera Calibration Toolbox for Matlab [Cal] which rectifies each pair of calibration images. This works as follows: we first bring the two cameras into the same orientation by rotating them 'minimally', so as to bring the translation vector $\mathbf{t}$ in alignment with the positive x-axis $(1, 0, 0)^T$. Global rotations are applied to both cameras so that the resulting rigid motion between the cameras is just a translation and the rotation matrix becomes $\mathbf{R} = \mathbf{I}$. The vertical component of the estimated focal length f_u must be the same in both images and we set the horizontal focal length f_v to the same as the vertical, resulting in square pixels, and new principle points are chosen to be the average of the two principle points. New camera projection matrices are computed accordingly and we warp each image so that the epipolar lines are horizontal. A pair of rectified images for a frame of the captured image sequence is shown in Figure 3-7. Matching image points and their corresponding epipolar lines are plotted.

Figure 3-7: 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.

3.3 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:

$$C_{NCC} = \sum_{i=1}^{N} \frac{(I_i^L - \mu^L)(I_i^R - \mu^R)}{\sigma^L \sigma^R} \tag{3.3}$$

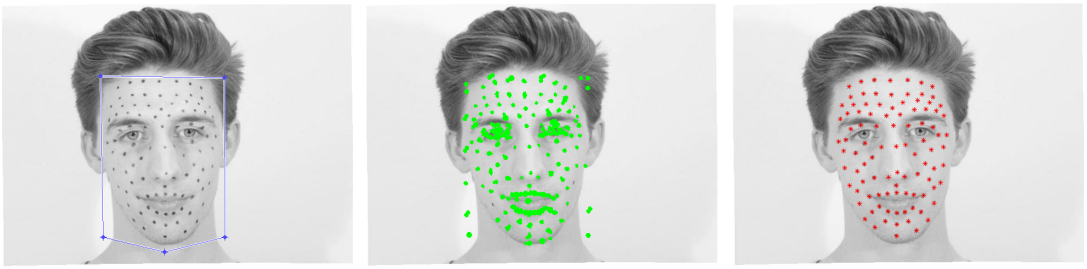


Figure 3-8: Facial markers are detected by first selecting a region of interest, detecting SIFT features, and then removing unwanted points or adding additional points.

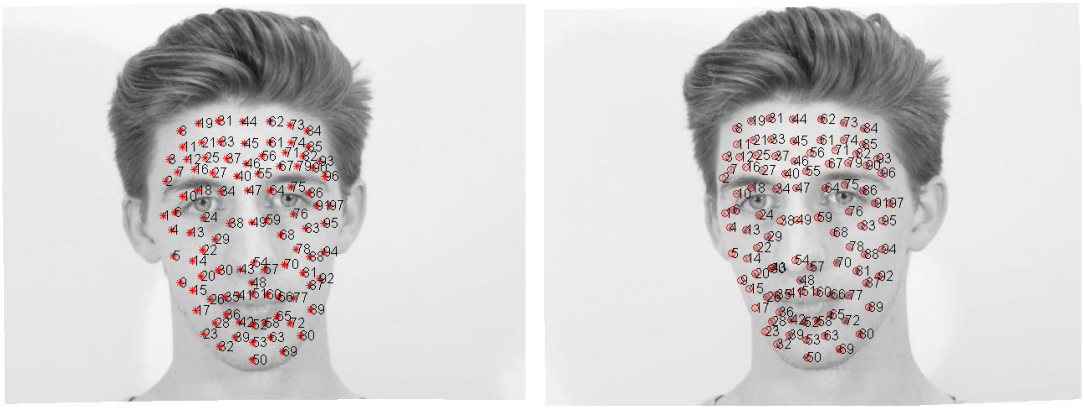


Figure 3-9: Detected facial markers in the left image are matched to markers in the right image pair by computing the normalised cross-correlation between image patches around the points.

3.4 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.

3.4.1 Optical Flow

3.4.2 KLT Tracker

[ST94]

3.4.3 Kalman Filter

3.5 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 $\mathbf{P}'$:

$$\begin{pmatrix} su \\ sv \\ s \end{pmatrix} = \mathbf{P} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \qquad \begin{pmatrix} su' \\ sv' \\ s \end{pmatrix} = \mathbf{P}' \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}. \tag{3.4}$$

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 $\mathbf{P}$

$$\begin{bmatrix} u\mathbf{p}_3^T - \mathbf{p}_1 \\ v\mathbf{p}_3^T - \mathbf{p}_2 \\ u'\mathbf{p}'^T_3 - \mathbf{p}'_1 \\ v'\mathbf{p}'^T_3 - \mathbf{p}'_2 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = 0. \tag{3.5}$$

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.

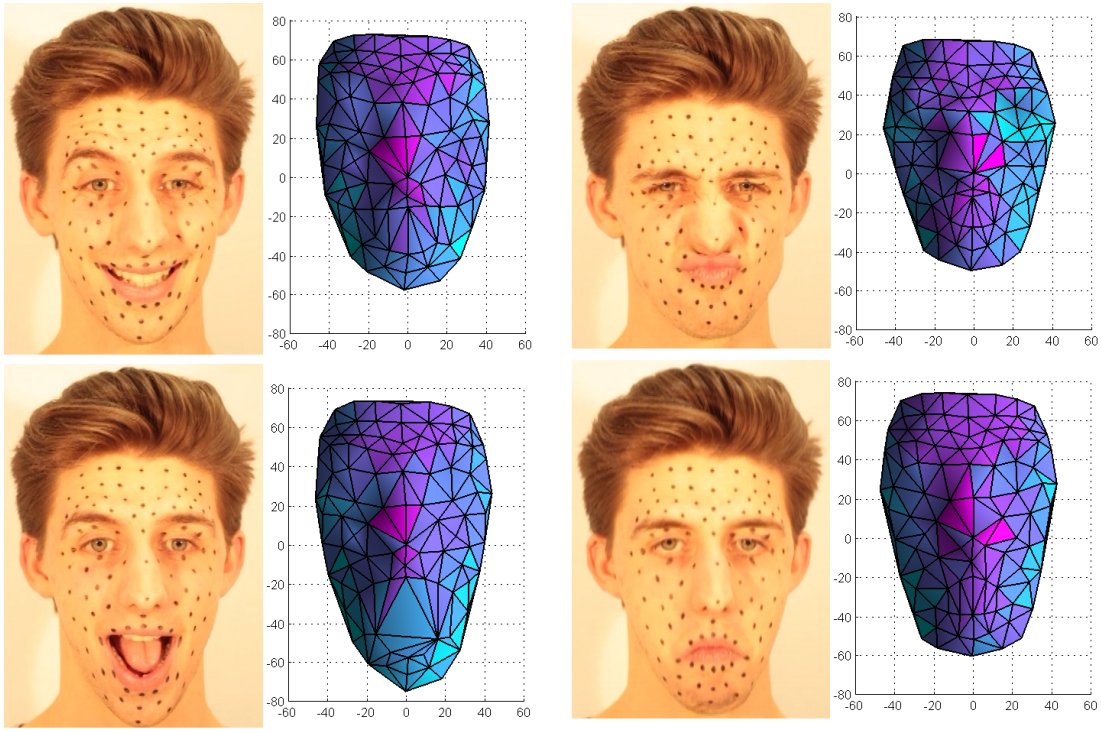


Figure 3-10: A sparse 3D reconstruction is computed for every frame of an image sequence using the detected marker positions and the calibrated camera projection matrices. A sample of image frames and the corresponding 3D reconstructions are shown above.

3.5.1 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 blend-shape 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 3-11. 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.

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 3-11, 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.

3.6 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.

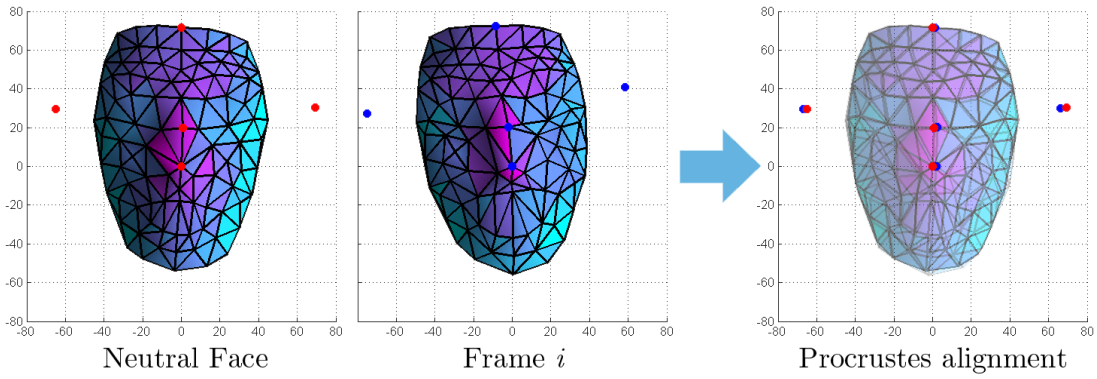


Figure 3-11: The head pose in each frame of an image sequence is aligned to the neutral face position using Procrustes analysis. The rigid vertex correspondences are shown in red for the neutral pose, and blue for a given frame.

3.6.1 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:

$$E_{bend}(f) = \lambda \iint \left(\left(\frac{\delta^2 f}{\delta x^2} \right)^2 + 2 \left(\frac{\delta^2 f}{\delta xy} \right)^2 + \left(\frac{\delta^2 f}{\delta y^2} \right)^2 \right) \mathrm{d}x \, \mathrm{d}y, \tag{3.6}$$

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:

$$f(x, y) = \sum_{i=1}^{N} \omega_i \, \phi(\|(x, y) - (x_i, y_i)\|), \quad \text{where } \phi(r) = r^2 \log(r). \tag{3.7}$$

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

$$\sum_{i=1}^{N} \omega_i = \sum_{i=1}^{N} \omega_i x_i = \sum_{i=1}^{N} \omega_i y_i = 0. \tag{3.8}$$

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

$$\begin{bmatrix} K & P \\ P^T & 0 \end{bmatrix} \begin{bmatrix} \omega \\ o \end{bmatrix} = \begin{bmatrix} v \\ o \end{bmatrix}, \tag{3.9}$$

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. 3.8, 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].

3.6.2 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].

3.6.3 Animation

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 [ARL*09], [Facb]. 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.

Initial Approach

We manually choose a sparse set of points that corresponds to the sparse source mesh. 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 using TPS. The transformation is stored, and it is used to map all the frames 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. 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 cannot 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.

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. 3-12.

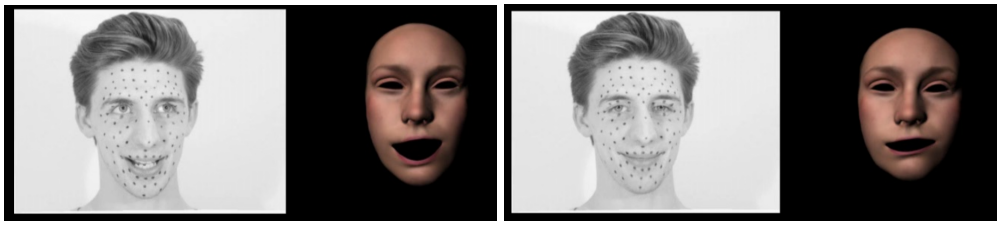


Figure 3-12: Our first animation produced by warping the ector's expressions at each frame of the sequence.

Introduction of Blendshapes

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 blendshapes is that in most situations a simple linear relation exists between an arbitrary facial expression and the set of key-shapes. The shape may encode the entire geometry of the case, the displacement from the neutral face, or the local features of a face. If no such shapes are available, they may be producing using PCA. Such set of blendshapes 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 blendshapes $\mathbf{B} = [\mathbf{b}_0, \ldots, \mathbf{b}_n]$ 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:

$$\mathbf{F}(\mathbf{w}) = \mathbf{b}_0 + \sum_{i=1}^{N} w_i \, |\mathbf{b}_i - \mathbf{b}_0|, \tag{3.10}$$

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 blendshape model, i.e. $w_i \in [0, 1]$. Eq. 3.10 may be written as a linear system of equations as follows:

$$\mathbf{F}(\mathbf{w}) - \mathbf{b}_0 = \hat{\mathbf{B}}\mathbf{w}, \tag{3.11}$$

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

A number of different numerical optimisation methods where tested when solving for the blendshape 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. 3.11. 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:

$$\min_{\mathbf{w}} ||\hat{\mathbf{B}}\mathbf{w} - (\mathbf{F}(\mathbf{w}) - \mathbf{b}_0)||_2^2, \ \mathbf{w}_i \geq 0, \ i = 1, \ldots, n. \tag{3.12}$$

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.

Two-dimensional Sequence

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. 3-13. The sparse points from these frames are then used as blendshapes. 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. 3-14 and Fig. 3-15 illustrate the concept of blendshapes. Frame 10 is very close to one of the blendshapes, 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 or approximately 0.09 centimetres. Note that as expected both methods use frame 9 with a large weight. However, the other blendshapes are different for the two methods. This suggests that none of the blendshapes 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 blendshape, resulting in an increase in average error for the two methods (approximately 0.49 in both cases). As expected, both methods use the two blendshapes 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 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, at least for this data set the difference does not seem significant.

Figure 3-13: Set of facial expressions used as blendshapes. The first frame is used as a neutral expression.

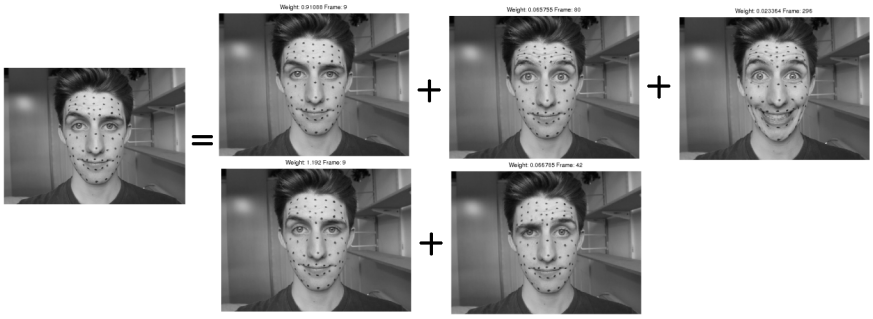


Figure 3-14: Frame 9 on the left represented as a linear combination of the key frames. The top solution was obtained using the `lsqnonneg` solver while the bottom one was obtained using `lsqlin`.

Principal Component Analysis

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

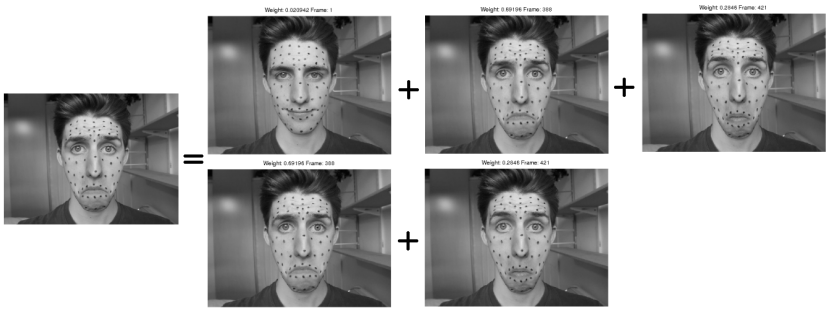


Figure 3-15: Frame 400 on the left represented as a linear combination of the key frames. The top solution was obtained using the `lsqnonneg` solver while the bottom one was obtained using `lsqlin`.

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 blendshapes, see Sec. 2.3. The main advantages of using such blendshapes is that the construction does not require artistic skills, and the shapes are guaranteed to be orthogonal to each other. The second argument is particularly important in compression algorithms, i.e. when the aim is to find the smallest possible set of blendshapes 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 blendshapes, 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 blendshapes from the Digital Emily project.

Animation of a 3D Model

Next, we shall consider the animation of the previously introduced three dimensional motion sequence. The captured sequence contains 371 frames and shows the actor talking from approximately 12 seconds. The corresponding sequence in the Emily domain is obtained using a linear elasticity method, see Sec. 3.6.1. We use 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 blendshape 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; we try to replicate the pattern we used when capturing the actor's face, see Fig. 3-3. 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 the blendshapes, see Fig. 3-16. 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 centimetres for the constrained linear least-squares solver, see Fig. 3-17. A number of factors may increase the error. Firstly, the expressions captured in Emily blendshapes 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 blendshapes. Secondly, as the blendshapes are not orthogonal, they may combine in unexpected ways, for example cancel each other. This behaviour may be avoided if a set of PCA-produced shapes is used. Also, note that the mesh of Emily 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 blendshapes.

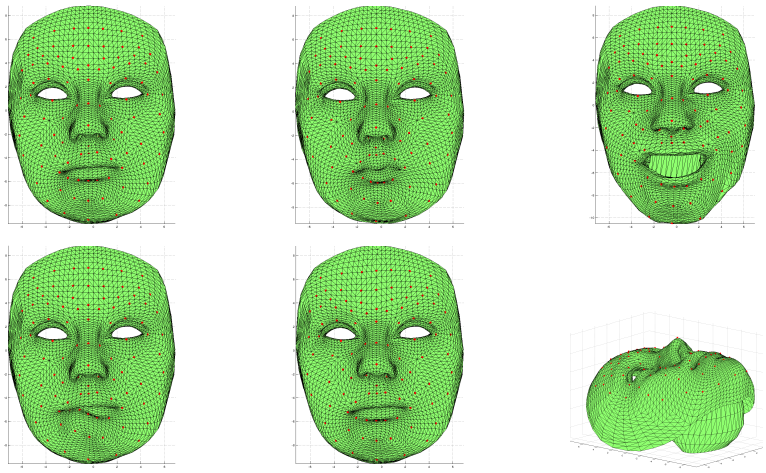


Figure 3-16: Examples of marked blendshapes.

The use of blendshapes 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 sparse 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.

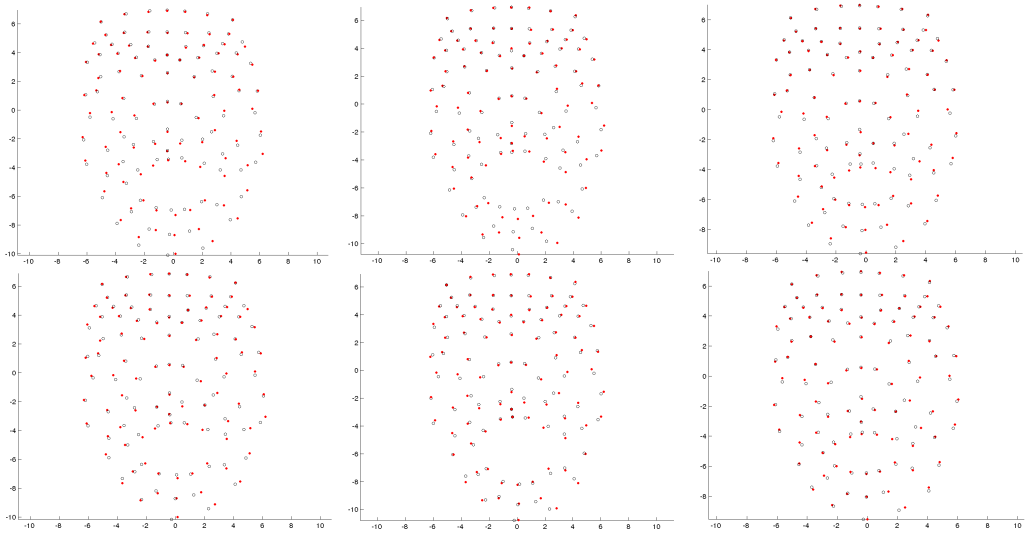


Figure 3-17: The red dots mark the position of the markers in the motion sequence while the black cirles correspond to the reconstructed position of the markers. The top results were produced using the non-negative least-squares solver while the bottom results were produced by the linear least-squares solver.

Moreover, the solution produced by the constrained non-negative least squares 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 is closely related to the previous frames [BWP13]. This constraint may be included 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. The resulting problem is generally hard to solve, but there exist a number of algorithms that minimise the sum of Euclidean norms. Beside the one mention in the paper by Bouaziz et al., we have considered the algorithms proposed by Andersen et al., and Xue and Ye [ACCO00], [XY97]. However, after examining the solution produced by our current methods, we noticed that it automatically obeys this smoothness constraint; this may be explained by the fact that the input sequence was filmed at a relatively high frame rate (60 frames per second). Specifically, we compared the change in blendshape weights from frame to frame; the average difference is 0.007, which, when reconstructed, is hardy noticeable. The highest differences (of around 0.3) 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 low frame rate, then the implementation of the temporal constraints might improve the quality of the resulting 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 blendshape 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. 3-18. Thus the

only notable advantage of using fewer blendshapes 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.

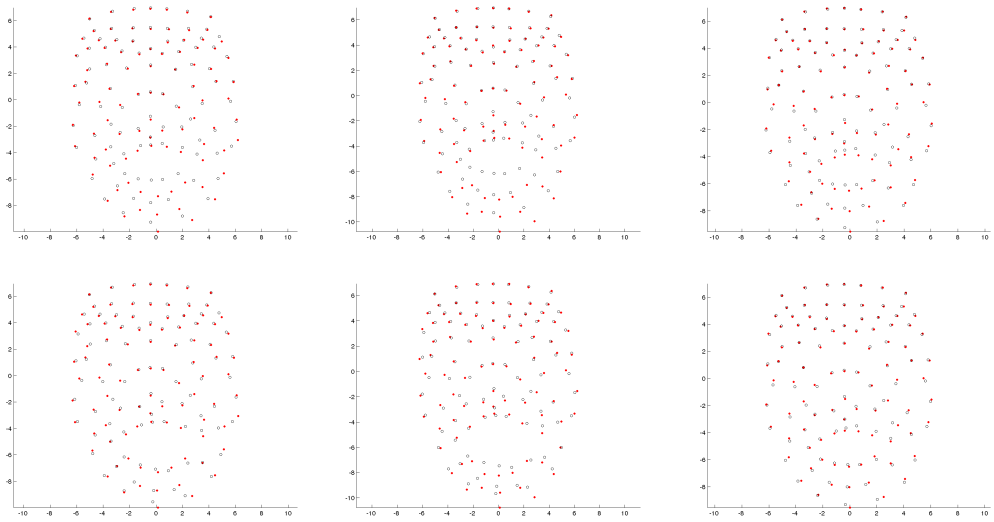


Figure 3-18: The reconstruction error for a model with 19 blendshapes. The top results were produced using the non-negative least-squares solver while the bottom results were produced by the linear least-squares solver. Colours as in Fig. 3-17.

For comparison, we attempt the opposite approach; we include extra blendshapes 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 members of the original set, and that improve the fitting accuracy. Unfortunately, such corrective method cannot be applied to a manually constructed set of blendshapes since it relies on vector orthogonality. Instead, we manually construct a set of new shapes that are more extreme than our original blendshapes; 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 of the new expressions in comparison to the existing ones. The four extra blendshapes are shown in Fig. 3-19; they were constructed by increasing the weights on the original blendshapes beyond the predefined limit. Fig. 3-20 shows the activation of the extra blendshapes throughout the motion sequence. Both of the optimisation methods favour the first additional blendshape that corresponds to the smile in Fig. 3-19 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 systematic approach needs to be used to produce more diverse shapes. For example, an additional

Figure 3-19: Additional blendshapes.

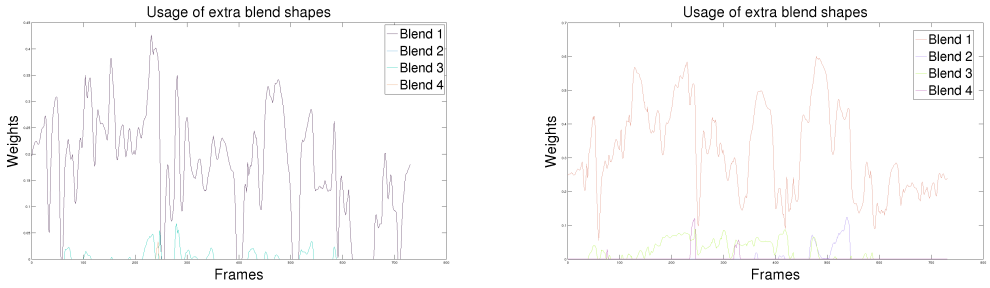


Figure 3-20: The weights associated with the extra blendshapes as a function of frames in the sequence. The left diagram corresponds to the constrained non-negative least-squares solver while the right one gives he results for the linear least-squares solver.

optimisation step may be included that randomly alters the existing blendshapes, and uses a small subset of the altered shapes to find the extension of the blendshape set 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 blendshapes 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.

Estimation of Weights in Actor's Domain

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 3-21. 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.

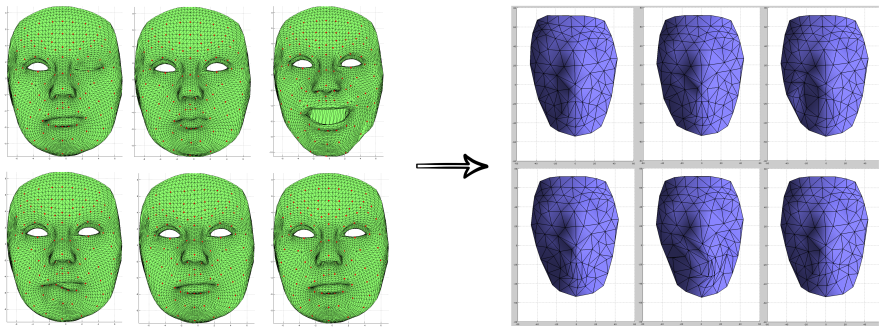


Figure 3-21: Use the inverse thin plate spline transform $\mathbf{T}^{-1}$ to produce a set of blendshapes in the actor's domain. The left images are the sparse Emily blendshapes (red dots) while the right images are the sparse actor's blendshapes (vertices of the mesh)

Fig. 3-22 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. The mean reconstruction error is smaller than 0.01 centimeters which is an improvement over the previous methods. We test our new approach using another motion capture sequence; in the new sequence the actor pulls the expressions listed in the Facial Action Coding System [FACc]. This sequence is more challenging as the expressions are diverse and intricate as opposed to the rather monotonous motion in the talking sequence.

We note that our final animation was not always able to capture the slight smiles that are noticeable in the recorded 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. 3-23. 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 an additional small positive weight to the blendshapes that correspond to a smiling face, see Sec. 4.

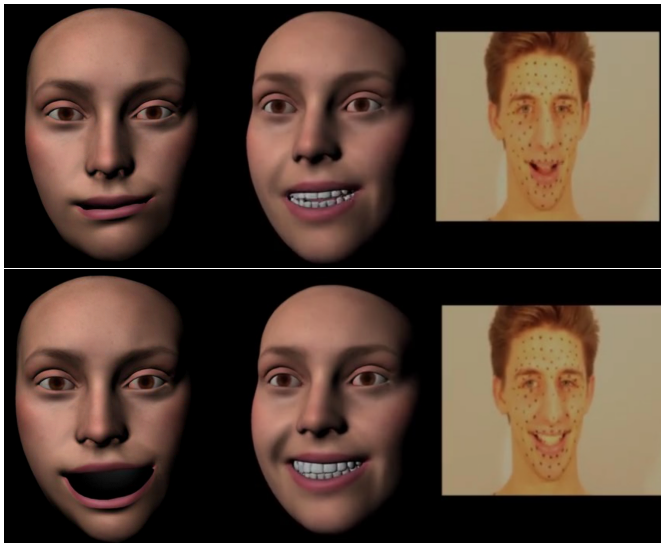


Figure 3-22: The expressions on the left were produced using our original approach while the expressions in the middle use the weights that were calculated in the actor's domain.

Figure 3-23: The comparison of neutral expressions.

3.7 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].

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

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 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 3-24, 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.

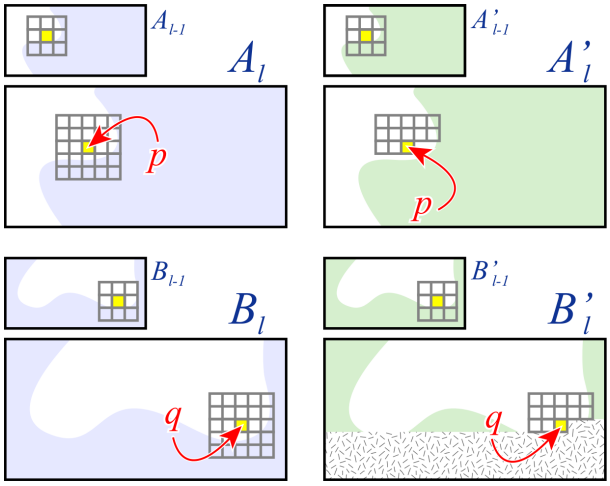


Figure 3-24: Neighbourhood matching for the Image Analogies framework, image taken from [Ash01].

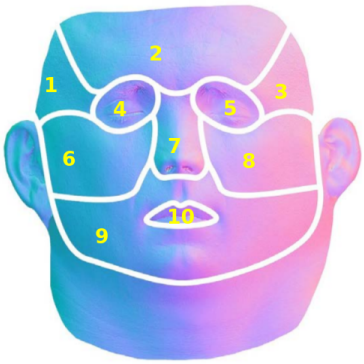


Figure 3-25: Texture segmentation in common coloured areas, image taken from [GTB*13].

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 3-26. 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.

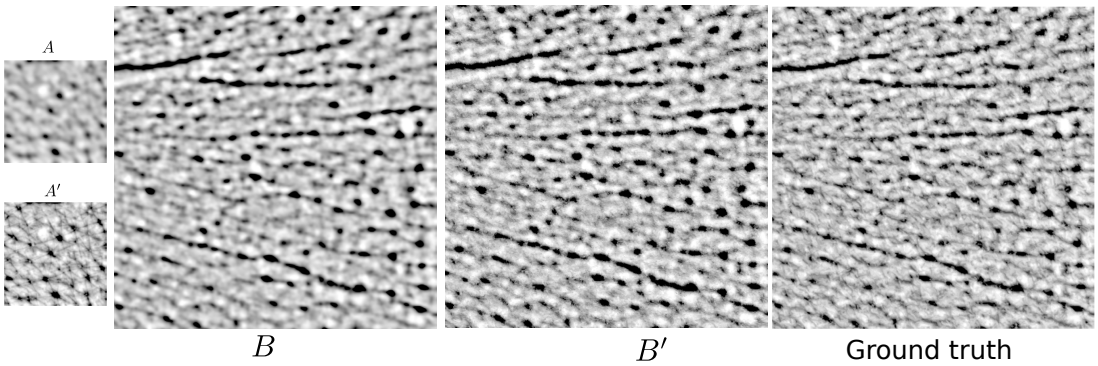


Figure 3-26: Bum map deblurring, A, A', B and ground truth images images taken from [GTB*13].

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 3-25 and results are shown in Figure 3-27.

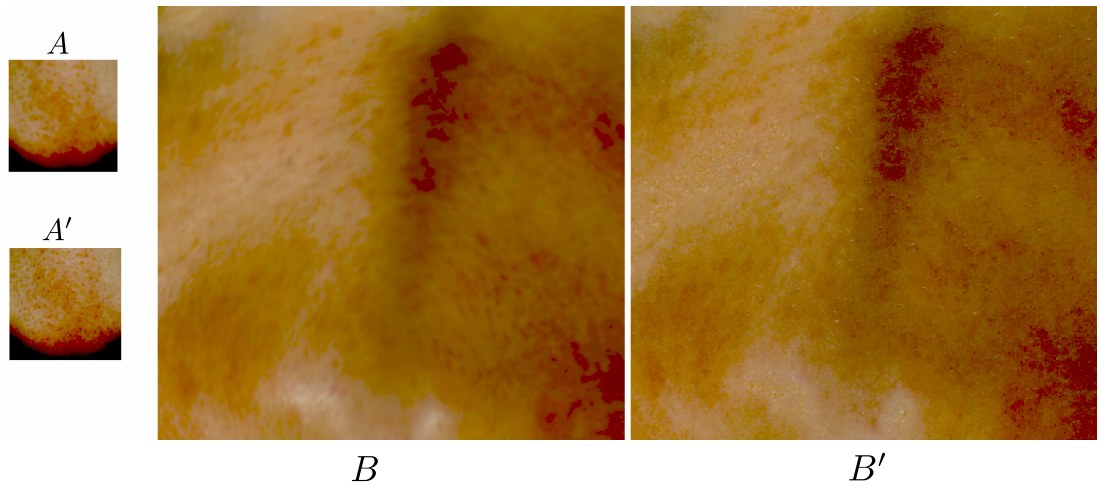


Figure 3-27: Texture quality increase using image analogies, the texture is false-coloured to highlight the differences.

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 3-28.

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 3-30. 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 3-31.

The advantages of using a bump mapping are shown in Figure 3-32. In this example, a render with the original texture is shown in Figure 3-32a. The original RGB texture was converted to a gray scale image and applied directly as a bump map texture, results are shown in Figure 3-32b, note how the addition of high frequency imperfections adds extra realism to the image.

Figure 3-28: 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].

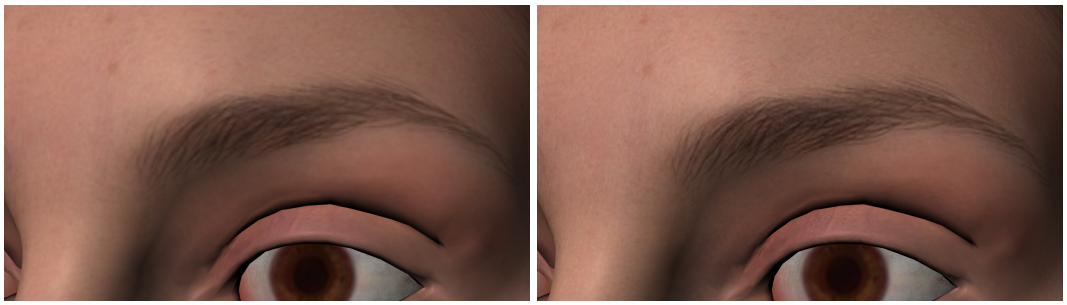


Figure 3-29: Close up of texture super-resolution for Emily, (left) original texture, (right) synthesized texture with higher resolution.

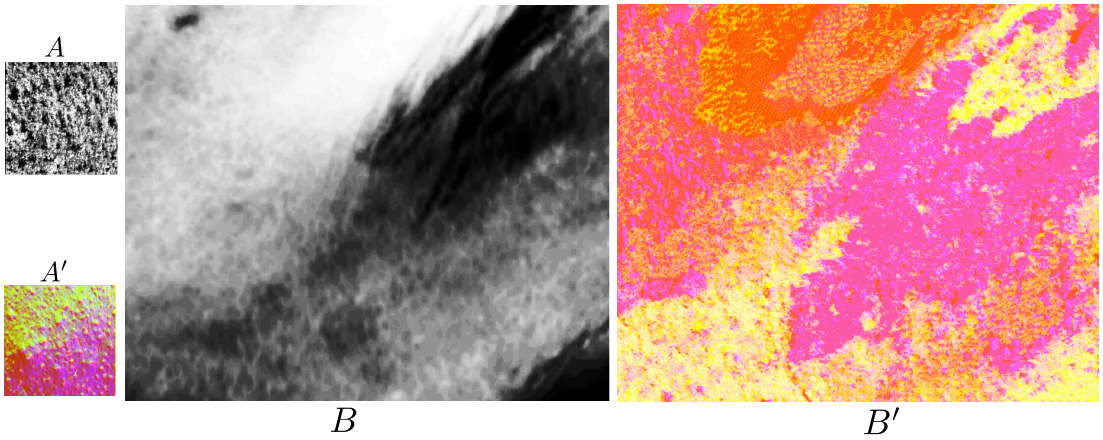


Figure 3-30: Normal synthesis from albedo image.

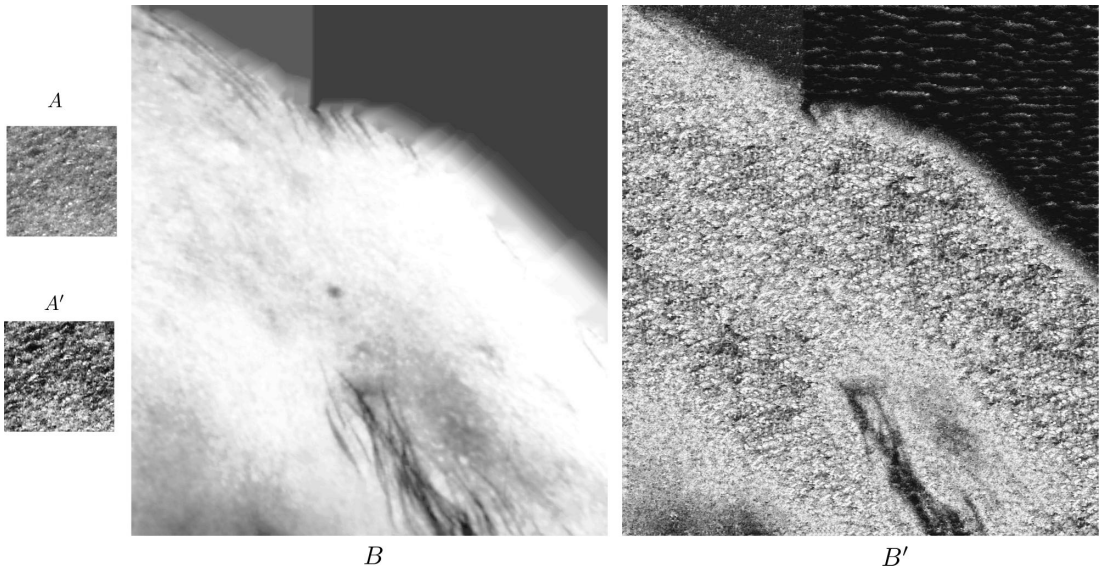
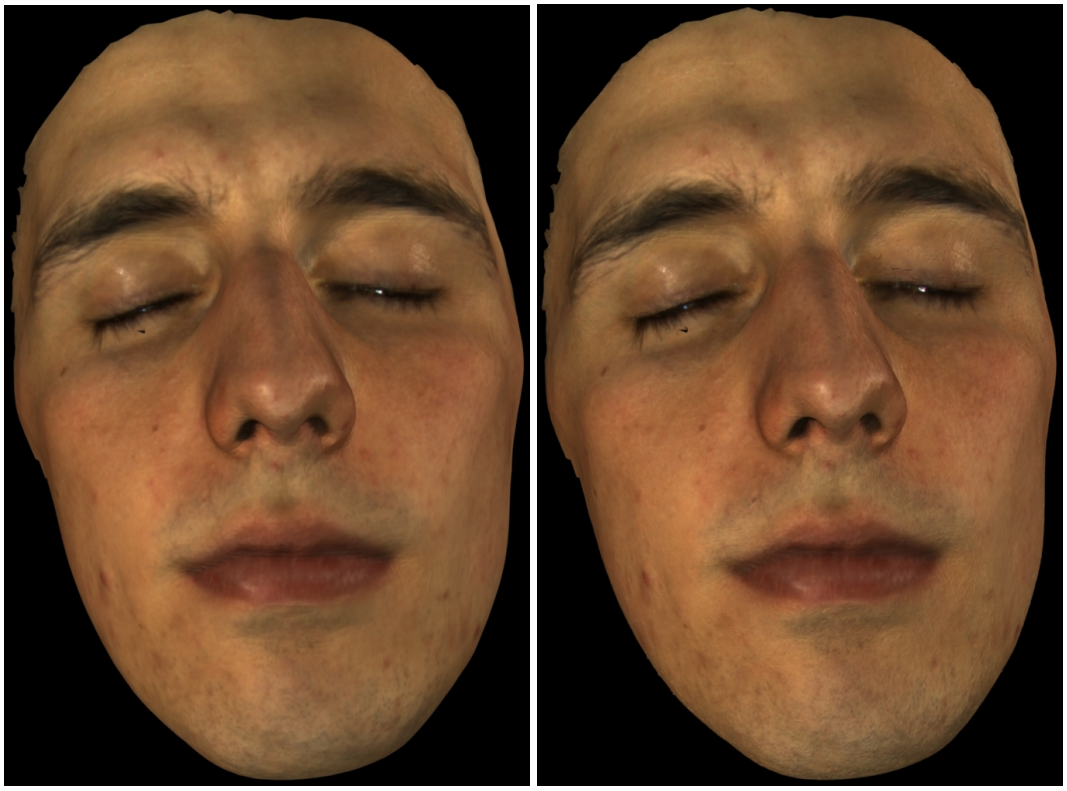


Figure 3-31: Improving the quality of a bump map from a gray scale texture.



(a) Render with the original texture.

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

Figure 3-32: The advantages of using a bump map..

Bibliography

[ACCO00] Andersen K., Christiansen E., Conn A., Overton M.: An efficient primal-dual interior-point method for minimizing a sum of euclidean norms. *SIAM Journal on Scientific Computing 22*, 1 (2000), 243–262. cited By 60.

[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.

[FACc] Facial Action Coding System. http://www.cs.cmu.edu/~face/facs.htm. Accessed: 2015-05-27.

[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.

[UV] Uncanny Valley. http://en.wikipedia.org/wiki/Uncanny_valley. Accessed: 2015-05-27.

[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.

[XY97] Xue G., Ye Y.: An efficient algorithm for minimizing a sum of euclidean norms with applications. *SIAM Journal on Optimization 7*, 4 (1997), 1017–1036. cited By 53.

[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.