3D Shape Regression for Real-time Facial Animation

Chen Cao* Yanlin Weng* Stephen Lin† Kun Zhou*

*State Key Lab of CAD&CG, Zhejiang University †Microsoft Research Asia

Abstract

We present a real-time performance-driven facial animation system based on 3D shape regression. In this system, the 3D positions of facial landmark points are inferred by a regressor from 2D video frames of an ordinary web camera. From these 3D points, the pose and expressions of the face are recovered by fitting a user-specific blendshape model to them. The main technical contribution of this work is the 3D regression algorithm that learns an accurate, user-specific face alignment model from an easily acquired set of training data, generated from images of the user performing a sequence of predefined facial poses and expressions. Experiments show that our system can accurately recover 3D face shapes even for fast motions, non-frontal faces, and exaggerated expressions. In addition, some capacity to handle partial occlusions and changing lighting conditions is demonstrated.

CR Categories: I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation;

Keywords: face tracking, monocular video tracking, 3D avatars, facial performance, user-specific blendshapes

Links: DL PDF

1 Introduction

Performance-based modeling provides an essential means of generating realistic facial animations, as detailed facial motions and expressions are often difficult to synthesize convincingly without natural examples. This approach has commonly been used in film and game production to better convey emotions and feelings through virtual characters. This form of non-verbal communication could also play an important role in personal interactions via avatars, which have been growing in popularity through online games and video chats. For such applications there is a need for performance-driven facial animation that can operate in real-time with common imaging devices.

Facial performance capture is a challenging problem that is made more manageable in many techniques by using special equipment, such as facial markers [Williams 1990; Huang et al. 2011], camera arrays [Bradley et al. 2010; Beeler et al. 2011], and structured light projectors [Zhang et al. 2004; Weise et al. 2009]. Towards a more practical solution for ordinary users, Weise et al. [2011] presented a real-time method that utilizes depth maps and video from Microsoft's Kinect camera. While compelling results have been demonstrated with their system, a method based instead on conventional web cameras would be more broadly practical because of their widespread availability with PCs as well as on tablets and smartphones. Face animation methods have also been designed for basic video input [Essa et al. 1996; Pighin et al. 1999; Chai et al. 2003; Vlasic et al. 2005], but their heavy reliance on optical flow or feature tracking can lead to instability, especially in cases of rapid head or facial motions, or changes in the lighting/background.

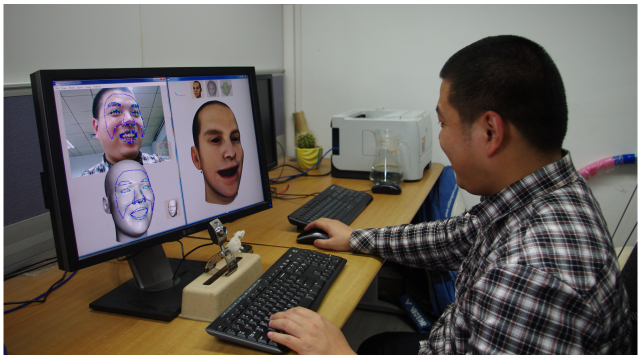

Figure 1: *Our real-time facial animation system using a web camera. The camera records* 640 × 480 *images at* 30 *fps. Our system runs at over* 24 *fps on a PC.*

In this paper, we propose a robust approach for real-time performance-based facial animation using a single web camera (see Fig. 1). As a setup for the system, the user acts out a set of standard facial expressions, the images of which are used to train a user-specific regressor that maps 2D image appearance to 3D shape. At run time, this 3D shape regressor is used in tracking the 3D positions of facial landmarks from a 2D video stream. The head's rigid transformation and facial expression parameters are calculated from the 3D landmark positions, and they are then transferred to a digital avatar to generate the corresponding animation.

Our main technical contribution is a novel 3D shape regression algorithm for accurate 3D face alignment. Regression modeling serves as an effective tool for learning a predictive relationship between input variables (e.g., a 2D face image) and an output response (e.g., the corresponding 3D facial shape) from a set of training data. To facilitate modeling, suitable training data for our regressor is efficiently constructed using the predefined setup images of the user and simple manual adjustments to automatic face alignment results on those images. From this data our 3D shape regressor learns an effective prediction model through an inherent encoding of the geometric relationships that the user's data contains. This user-specific regression modeling approach is experimentally shown to surpass previous appearance-based tracking methods that fit a generic 3D face model to 2D images.

Our facial animation system is highly practical because of its following properties:

- Ease of use: requires just an ordinary web camera and no facial markers; involves a simple one-time setup step.

*Contact: weng@cad.zju.edu.cn, kunzhou@acm.org
†stevelin@microsoft.com

ACM Reference Format
Cao, C., Weng, Y., Lin, S., Zhou, K. 2013. 3D Shape Regression for Real-time Facial Animation. ACM Trans. Graph. 32, 4, Article 41 (July 2013), 10 pages. DOI = 10.1145/2461912.2462012 http://doi.acm.org/10.1145/2461912.2462012.

DOI: http://doi.acm.org/10.1145/2461912.2462012

- Robustness: effectively handles fast motions, large head rotations, and exaggerated expressions; deals with changes in lighting and background to a moderate extent; works well in a wide range of environments from indoor offices to outdoor surroundings under sunlight.
- Computational efficiency: computationally independent of the resolution of video frames; takes less than 15ms to process a video frame at run time, without the need for a GPU.

With this level of performance and its modest requirements, the system is suitable even for mobile devices.

In the remainder of the paper, we first review some related work on facial performance capture, facial feature tracking, and 3D face models. In Sec. 3 we present an overview of our system. The following sections describe the components of our algorithm, including the 3D shape regression in Sec. 4 and facial tracking in Sec. 5. Sec. 6 presents experimental results, and the paper concludes in Sec. 7 with a discussion of future work.

2 Related Work

Facial Performance Capture. Facial animation systems based on performance capture aim to measure expressions and motions of a user's face and apply them to a target model. Various techniques have been presented for acquiring these measurements. Methods that employ direct interactions with the user are often referred to as *active sensing* techniques. They may involve placing markers on the face [Williams 1990; Huang et al. 2011] or projecting structured light patterns [Zhang et al. 2004; Weise et al. 2009] to facilitate surface point tracking or 3D modeling. These approaches can provide high resolution and reliability in face measurement, but their intrusiveness makes them less practical for general settings. An exception to this is the Kinect-based method of [Weise et al. 2011]. Because it utilizes projections of invisible, infrared light, the Kinect is relatively nonintrusive in its face capture.

Passive systems do not interfere with their environment to gain information in the sensing process, but tend to have lower accuracy as a result. Several methods employ just a single conventional camera to capture facial motions [Essa et al. 1996; Pighin et al. 1999; Chai et al. 2003; Vlasic et al. 2005]. Camera arrays have also been used with the advantage of providing multi-view stereo data for 3D reconstruction [Beeler et al. 2010; Bradley et al. 2010; Beeler et al. 2011]. Since widespread usability is a motivating factor in our work, we utilize a single video camera to record facial configurations. Though video frames provide only 2D appearance data, we infer 3D face shape from this input with the help of face models.

Appearance-based Facial Feature Tracking. Facial motion capture requires accurate tracking of facial landmarks such as the corners of the eyes and ends of the mouth. For conventional video input, optical flow is typically applied for this purpose. Tracking of individual features by optical flow, however, is unreliable especially for less salient landmarks, because of noise in the input data. For more robust tracking, geometric constraints among the features are normally incorporated in the flow computation so that the tracking of each feature is influenced by the positions of others. Different types of geometric constraints have been proposed, including restrictions on feature displacements in expression change [Chai et al. 2003], adherence to physically-based deformable mesh models [Essa et al. 1996; DeCarlo and Metaxas 2000], and correspondence to face models constructed from measured examples [Pighin et al. 1999; Blanz and Vetter 1999; Vlasic et al. 2005]. In [Weise et al. 2011], motion priors derived from pre-recorded animations are also used in conjunction with a 3D face model.

Our work also utilizes a model-based approach to locate facial landmarks, but only as part of an offline face alignment step in which errors are manually corrected by the user. At run time, instead of fitting a generic face model to optical flow, our method directly infers 3D shape from 2D image appearance through boosted regression, a powerful technique for learning a mapping function from training data. In our system, the regressor learns from user-labeled training data that is more accurate than what can be produced by automatic face alignment techniques. Moreover, the geometric relationships encoded in the training data are specific to the given user, and thus provide a more precise constraint on tracking results than that of a generic face model. These features lead to appreciable improvements in tracking performance. The use of shape regression for this application was inspired by the work in [Cao et al. 2012], which deals with 2D face alignment that is not tailored to a particular user. Our presented regression to 3D shapes allows us to take advantage of a user-specific 3D face model to better track facial expressions while accounting for configurations that are not well represented by linear interpolation of the training data.

Regression-based Generation of Virtual Avatars. Regression-based approaches have previously been used in the generation of virtual avatars. Saragih et al. [2011] synthesized training pairs for various expressions of a user and an avatar using a generic expression model, and learned a mapping function between the two. At run time, the user's face is tracked by fitting a deformable face shape model, which is then used with the learned map to generate the corresponding shape for the avatar's face. Huang et al. [2012] take a similar approach that instead learns a regression model between shape differences and appearance/texture differences from that of a neutral face. Shape differences of a source face are transferred to the target face by applying corresponding affine transformations to triangles in the facial mesh model. In contrast to these techniques, which employ regression to learn mappings between a source face and target face over various expressions, ours uses a regression-based approach to improve 3D face alignment, while taking advantage of user-specific blendshapes for expression transfer. It is through this novel face alignment technique that our system obtains robust performance over large head rotations and exaggerated expressions.

3D Face Models. Various 3D models for human faces have been used in computer graphics and computer vision. One common form is as linear models based on principal components analysis (PCA). PCA shape models have commonly been fit to face images to recover facial shape [Blanz and Vetter 1999; Matthews et al. 2007; Cootes et al. 2012]. PCA has also been used to model other facial shape descriptors, such as surface height and surface gradient fields [Castelan et al. 2007]. While PCA face models provide a compact representation and computational simplicity, these generic parametric models have a limited capacity to represent the particular detailed shape variations of an individual's facial expressions, due to the limited number of PCA dimensions that can be used in practice. In addition, using a parametric model for alignment is less than ideal, since smaller PCA model errors do not necessarily equate to smaller alignment errors. Instead of using a fixed parametric shape model, our regression-based approach encodes the shape constraints of a given user in a non-parametric manner that can flexibly represent the particular facial expressions of the user. Linear models have also been used in reconstructing 3D face shapes through structure-from-motion [Xiao et al. 2006], but as it involves factorization of a matrix containing 2D feature point positions over multiple frames, this would be unsuitable for on-the-fly processing of video streams.

Often used for face animation are models based on blendshapes, a representation whose basis consists of a set of facial poses that span a valid space of expressions. From a blendshape model, anima-

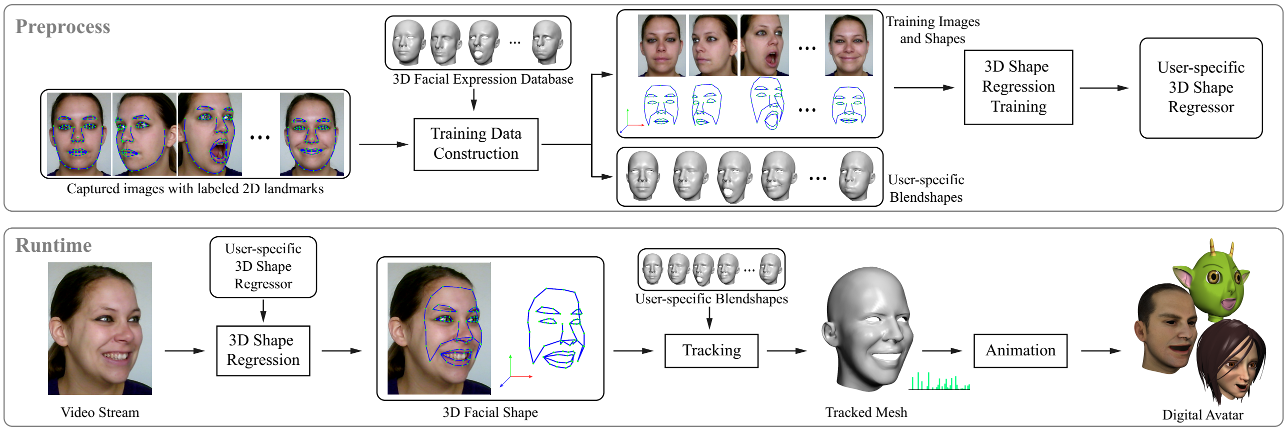
Preprocess
3D Facial Expression Database
Training Data Construction
Captured images with labeled 2D landmarks
Training Images and Shapes
3D Shape Regression Training
User-specific 3D Shape Regressor
User-specific Blendshapes
Runtime
Video Stream
User-specific 3D Shape Regressor
3D Shape Regression
3D Facial Shape
User-specific Blendshapes
Tracking
Tracked Mesh
Animation
Digital Avatar

Figure 2: *Pipeline of our system. Top: preprocessing in the one-time setup step. Bottom: run-time algorithm.*

tions can be generated by 3D shape morphing [Pighin et al. 1998] or through linear combinations of basis poses [Lewis and Anjyo 2010; Seo et al. 2011]. Multilinear models represent a decomposition of the blendshape basis that allows for separate control of different facial attributes (e.g., identity, expression, mouth articulation) [Vlasic et al. 2005]. In contrast to PCA models, blendshapes have the important advantage that a given facial expression among different people corresponds to a similar set of basis weights. Many face animation methods take advantage of this property to effectively transfer expressions from the user to an avatar. In this work we additionally utilize the blendshape representation to reduce ambiguity in fitting 3D face shapes to 2D images when constructing the regression training set. For retargeting motions between structurally different faces, Kholgade et al. [2011] presented a layered facial motion model, in which the source performance is decomposed into emotion, speech and blink layers, whose basis coefficients are then transferred to the corresponding layers of the target face. With bases composed of extreme configurations that enclose the space of each layer, this model bears similarities to blendshapes and can potentially be incorporated within our system.

3 System Overview

Our method begins with a setup step in which images are captured of the user performing a set of standard expressions and specified head poses. In these images, a set of facial landmarks are automatically located using a 2D facial feature alignment technique. Any of the landmark positions can be manually corrected by the user if needed. We then derive user-specific blendshapes from these labeled images and use the blendshape model to calculate for each image its *3D facial shape*, composed of 3D landmark positions. All of the input images and their 3D facial shapes are then used to train the user-specific 3D shape regressor. With this regressor, the 3D facial shape for each video frame can be computed in real time, and is used to estimate the rigid head transformation and facial expression. These parameters can be easily mapped to an avatar represented by expression blendshapes to drive the virtual character. Fig. 2 displays the pipeline of our system.

Coordinates Representation. In our pipeline, three different coordinate systems are used: object coordinates, camera coordinates and screen coordinates.

Object coordinates: Similar to many facial animation techniques (e.g., [Weise et al. 2011]), we represent facial expressions in terms of user-specific expression blendshapes. An expression blendshape model contains the user's neutral face mesh plus 46 blendshapes $\mathbf{B} = \{B_0, B_1, ..., B_{46}\}$ based on the Facial Action Coding System (FACS) [Ekman and Friesen 1978]. With these blendshapes, facial expressions of the user are then replicated through linear combinations. The blendshape meshes are aligned by their centroids and defined in object coordinates. A face mesh with any expression can be represented as $B = B_0 + \sum_{i=1}^{46} \alpha_i B_i$, where $\mathbf{a} = \{\alpha_1, \alpha_2, ..., \alpha_{46}\}$ is a vector of expression coefficients.

On a face mesh is a set of landmark vertices, which correspond to certain facial features such as eye corners, nose tip, points along the brow and mouth boundaries, and points along the nose and face contours. The position of each landmark in object coordinates is denoted as $\tilde{\mathbf{x}}_i$.

Camera coordinates: Any face mesh B represented in object coordinates can be transformed to camera coordinates by a translation and rotation. We represent the transformation as a 3×4 matrix $\mathbf{M}$, such that the position of a landmark in the object space, $\tilde{\mathbf{x}}_i$, can be transformed to camera coordinates through $\mathbf{x}_i = \mathbf{M}(\tilde{\mathbf{x}}_i, 1)^T$. We define the collection of 3D landmark positions in camera coordinates as the 3D facial shape S for the face mesh.

Screen coordinates: A 3D point in camera coordinates can be transformed to a 2D position in screen (or image) space through a 3×3 perspective projection matrix $\mathbf{Q}$. For a 3D point $\mathbf{p}$ in camera coordinates, the projection matrix firstly transforms it to the 2D image space with homogeneous coordinates $\mathbf{u}_h = (u_h, v_h, w)$ by $\mathbf{u}_h = \mathbf{Q}\mathbf{p}$. Then, the vertex's 2D position in the image can represented as $\mathbf{u} = (u, v)$ with $u = u_h/w, v = v_h/w$. For convenience, we denote as $\Pi_{\mathbf{Q}}$ the mapping function that transforms a 3D position $\mathbf{p}$ in camera coordinates to a 2D position $\mathbf{u}$ in screen coordinates:

$$\mathbf{u} = \Pi_{\mathbf{Q}}(\mathbf{p}). \tag{1}$$

Capture of Setup Images. In constructing a user-specific face model, we first capture a pre-defined sequence of setup images consisting of the user's face under a set of different facial configurations. These configurations are separated into two classes. The first is due to rigid motion, and is captured for 15 different head poses. These poses consist of head rotations over a sequence of angles with a neutral expression. The rotations are expressed in Euler angles as (*yaw, pitch, roll*) and include the following: yaw from $-90°$ to $90°$ at $30°$ intervals (with pitch and roll at $0°$); pitch from $-30°$ to $30°$ at $15°$ intervals except for $0°$ (with the other angles at $0°$), and roll with the same sampling as pitch. Note that the user need not match these rotation angles exactly; a rough approximation is sufficient.

The second class is non-rigid motion and consists of 15 different expressions each at three yaw angles ($-30°$, $0°$ and $30°$). Large expressions that vary widely among different people are used. Specif-

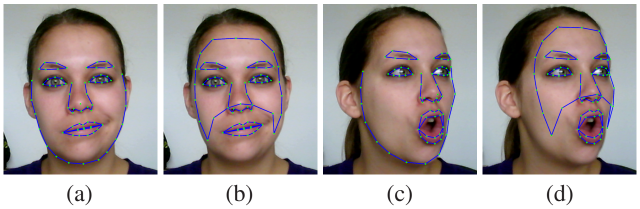

Figure 3: *Labeled landmarks for two captured images (a)(c), and projections of landmark vertices from the face meshes fit to the two images (b)(d).*

ically, they are mouth stretch, smile, brow raise, disgust, squeeze left eye, squeeze right eye, anger, jaw left, jaw right, grin, chin raise, lip pucker, lip funnel, cheek blowing and eyes closed.

In total, we capture 60 images for each user. The captured images for a user are provided in the supplementary material. Note that more images of different head poses and expressions could improve the accuracy of face tracking but would also introduce much more user interaction and increase the computational cost. In our experiments, the current image set provides a good tradeoff between accuracy and efficiency.

Landmark Labeling. To each facial image captured in the previous step we apply 2D face alignment to automatically locate 75 landmark positions, which are divided into two categories: 60 internal landmarks (i.e., features on eyes, brows, nose and mouth) located inside the face region, and 15 contour landmarks (exhibited in Fig. 3(a)(c)). Any alignment method may be used for this purpose, and in our implementation we employ the algorithm in [Cao et al. 2012]. The automatically detected positions may need adjustment in some images, so our system allows the user to manually correct them with a simple drag-and-drop tool to obtain 2D alignment results of ground truth quality. Note that each landmark position in the image corresponds with a landmark vertex in the 3D facial shape, such that the projection of the 3D vertex onto the image should match the 2D landmark point. These labeled images are used to generate user-specific blendshapes with the method described in Sec. 4.1.

3D Shape Regression. Given the labeled facial images, user-specific blendshapes and the camera's projection matrix, we fit the rigid motion parameters (transformation matrix $\mathbf{M}$) and expression coefficients $\mathbf{a}$ to the face in each image. Applying these values to the blendshape model gives us a 3D face mesh that matches the image's landmark positions. As the mesh vertices corresponding to the face contour may continually change in a video stream, for computation efficiency we replace the 15 contour landmarks with 15 internal landmarks whose 2D positions in the image are computed by projecting a set of pre-defined vertices in the fitted mesh onto the image (exhibited in Fig. 3(b)(d)). The 3D facial landmark positions on the mesh are then used to compose the 3D facial shape. The details of this procedure are presented in Sec. 4.2.

From a training set consisting of the labeled 2D images and corresponding 3D facial shapes, we learn a shape regression model for automatic 3D facial feature alignment. The training algorithm for the user-specific 3D shape regressor is described in Sec. 4.3.

At run time, the video frame along with the 3D facial shape computed for the previous frame are taken as input to the 3D shape regressor. The regression algorithm then outputs the 3D facial shape for the current frame in real time. Sec. 4.4 provides the details of this on-the-fly regression.

Tracking and Facial Animation. The calculated 3D facial shape is then used with the user-specific blendshapes to estimate the rigid motion parameters and expression coefficients. We formulate this as an optimization problem, which can be solved iteratively and gives reliable results even for strong expressions and rapid head movements. These tracking parameters are then applied to the expression blendshapes of a virtual avatar to drive its facial animation.

4 3D Shape Regression

Our regressor takes an image I and an initial estimate S^c of the 3D facial shape as input, updates S^c iteratively, and outputs a 3D facial shape S that is well aligned with the facial features in the input image. The 3D shape regression process involves four steps: generating the user-specific blendshapes, constructing the training set, training the regressor, and applying the regressor at run time. Each of these steps is described below.

4.1 User-specific Blendshape Generation

We generate user-specific blendshapes from the user's setup images with the help of FaceWarehouse [Cao et al. 2013], a 3D facial expression database containing the data of 150 individuals from various ethnic backgrounds, with 46 FACS blendshapes for each. A bilinear face model with two attributes, namely identity and expression, is built from the database, i.e., as a rank-three (3-mode) *core tensor* C_r (11K mesh vertices × 50 identity knobs × 47 expression knobs). With this tensor representation, any facial expression of any identity can be approximated by the tensor contraction

$$F = C_r \times_2 \mathbf{w}_{id}^T \times_3 \mathbf{w}_{exp}^T, \tag{2}$$

where $\mathbf{w}_{id}$ and $\mathbf{w}_{exp}$ are the column vectors of identity weights and expression weights in the tensor, respectively.

We employ an iterative two-step approach to compute the blendshapes using this bilinear face model. First, for each input image i, we find the transformation matrix $\mathbf{M}_i$, the identity weights $\mathbf{w}_{id,i}$ and the expression weights $\mathbf{w}_{exp,i}$ such that the projections of the 3D landmark vertices on the generated mesh match the 2D labeled landmark positions on the image. This can be achieved by minimizing the following energy:

$$E_d = \sum_{k=1}^{75} \left\| \Pi_{\mathbf{Q}} \left(\mathbf{M}_i (C_r \times_2 \mathbf{w}_{id,i}^T \times_3 \mathbf{w}_{exp,i}^T)^{(v_k)} \right) - \mathbf{u}_i^{(k)} \right\|^2, \tag{3}$$

where $\mathbf{u}_i^{(k)}$ is the 2D position of the k-th landmark in image i, and v_k is the corresponding vertex index on the mesh. Given the projection matrix of the camera, we can apply a coordinate-descent method to solve for $\mathbf{M}_i$, $\mathbf{w}_{id,i}$ and $\mathbf{w}_{exp,i}$.

In the second step, we refine the identity weights $\mathbf{w}_{id}$, which should be the same for all the images since they all show the same person. This is done by fixing the transformation matrix $\mathbf{M}_i$ and expression weights $\mathbf{w}_{exp,i}$ that were computed for each image i, and then computing a single set of identity weights $\mathbf{w}_{id}$ over all the images by minimizing

$$E_{joint} = \sum_{i=1}^{n} \sum_{k=1}^{75} \left\| \Pi_{\mathbf{Q}} \left(\mathbf{M}_i (C_r \times_2 \mathbf{w}_{id}^T \times_3 \mathbf{w}_{exp,i}^T)^{(v_k)} \right) - \mathbf{u}_i^{(k)} \right\|^2. \tag{4}$$

The two steps are iteratively repeated until the fitting results converge. In each iteration, we update the vertex indices corresponding to contour landmarks as in [Yang et al. 2011]. Convergence is reached in three iterations in our experiments.

Once we obtain the identity weights $\mathbf{w}_{id}^T$, we can construct the expression blendshapes $\{B_i\}$ for the user as

$$B_i = C \times_2 \mathbf{w}_{id} \times_3 (\check{\mathbf{U}}_{exp} \mathbf{d}_i), \quad 0 \le i < 47, \tag{5}$$

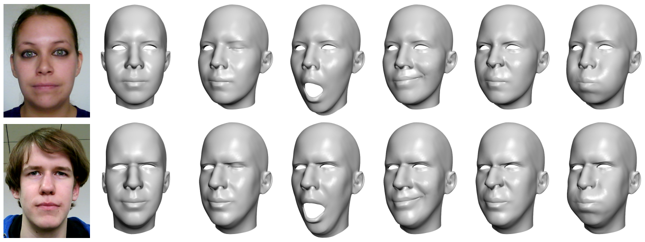

Figure 4: *Blendshape examples generated for two subjects.*

where $\check{\mathbf{U}}_{exp}$ is the truncated transform matrix for the expression mode in FaceWarehouse, and $\mathbf{d}_i$ is an expression weight vector with value 1 for the i-th element and 0 for other elements. Examples of blendshapes computed in this manner are shown in Fig. 4.

4.2 Training Set Construction

3D Facial Shape Recovery. For each of the n labeled setup images of the user with different head poses and facial expressions, we recover the corresponding 3D facial shape to use as training data for the 3D shape regressor. This requires solving for the rigid transformation $\mathbf{M}$ from object to camera coordinates and the expression coefficients of the user's blendshapes $\mathbf{a} = \{\alpha_1, \alpha_2, ..., \alpha_{46}\}$. Given the expression blendshapes $\{B_i\}$ for the user and the camera projection matrix $\mathbf{Q}$, the error between the labeled 2D landmark positions and the projection of the 3D landmark vertices is computed as the following energy:

$$E_l = \sum_{l=1}^{75} \left\| \Pi_{\mathbf{Q}}(\mathbf{M}(B_0 + \sum_{i=1}^{46} \alpha_i B_i)^{(v_l)}) - \mathbf{q}^{(l)} \right\|^2, \qquad (6)$$

where $\mathbf{q}^{(l)}$ is the 2D position of the l-th labeled facial landmark in the image, and v_l is the corresponding vertex index on the mesh.

Since the setup images consist of pre-defined standard expressions, their blendshape expression coefficients should be consistent with standard blendshape coefficients $\mathbf{a}^*$ for these expressions [Li et al. 2010]. We formulate this constraint as a regularization energy:

$$E_{reg} = \|\mathbf{a} - \mathbf{a}^*\|^2. \qquad (7)$$

We then solve for $\mathbf{M}$ and $\mathbf{a}$ via

$$\arg\min_{\mathbf{M},\mathbf{a}} E_l + w_{reg} E_{reg}. \qquad (8)$$

This energy is minimized using the coordinate-descent method, by alternately optimizing each parameter while fixing the others in each iteration. We initialize $\mathbf{a}$ to $\mathbf{a}^*$. In computing $\mathbf{M}$, we use the POSIT algorithm [Dementhon and Davis 1995] to find the rigid pose of the 3D mesh in the current iteration. In solving for $\mathbf{a}$, we use the gradient projection algorithm based on the BFGS solver [Byrd et al. 1995] to restrict the range of coefficients to between 0 and 1. The weight w_{reg} balances fitting error and consistency with the standard expression values. We set it to $w_{reg} = 10$ in our implementation. As done in solving Eq. 4, we update in each iteration the mesh vertex indices v_l corresponding to contour landmarks, using the method in [Yang et al. 2011].

After computing the rigid transformation matrix and expression coefficients for each image I_i, we generate their corresponding face meshes by $\mathbf{M}(B_0 + \sum_{i=1}^{46} \alpha_i B_i)$. From the 3D meshes we extract the 3D positions of facial landmarks to obtain the 3D facial shape for each image. We denote this set of face shapes as $\{S_i^o\}$.

Data Augmentation. To achieve better generalization in representing facial shapes, we augment the data of captured images and their 3D facial shapes. Specifically, for each captured image and its facial shape, (I_i, S_i^o), we translate the 3D shape S_i^o along each of the three camera coordinate axes to get $m - 1$ additional shapes $\{S_{ij}, 2 \le j \le m\}$. Instead of computing the images that correspond to the additional shapes, we simply record the transformation matrix $\mathbf{M}_j^a$ that maps S_{ij} back to the original shape S_i^o, which together with S_{ij}, I_i, and the camera's projection matrix $\mathbf{Q}$ provides enough information to retrieve the appearance data of the image that corresponds to the shape S_{ij}. Included with this set is the original shape S_i^o, which we will denote also as S_{i1} with an identity transformation matrix. This data augmentation process expands the n original data to $n \cdot m$, represented as $\{(I_i, \mathbf{M}_j^a, S_{ij})\}$. The augmented set of shapes $\{S_{ij}, 1 \le i \le n, 1 \le j \le m\}$ defines what we call a *3D shape space*, which spans the range of 3D facial shapes for the user.

Training Set Construction. To each of the augmented image/shape data we assign various initial shape estimates to obtain the training set for the regression algorithm. Since initial shape estimates for a video frame at run time will be determined based on the face shape of the preceding frame, we choose the initial shapes S_{ij}^c for training set construction considering both locality and randomness. For each image/shape pair $(I_i, \mathbf{M}_j^a, S_{ij})$, we first find the G most similar shapes $\{S_{i_g}, 1 \le g \le G\}$ to S_{ij} from the n original shapes $\{S_i^o\}$, and then randomly choose H shapes $\{S_{i_g j_h}, 1 \le h \le H\}$ from among the additional shapes generated from each S_{i_g} in the data augmentation step. This yields $G \cdot H$ initial shapes for S_{ij}, and each training data can be represented as $(I_i, \mathbf{M}_j^a, S_{ij}, S_{i_g j_h})$. To compute the similarity between two shapes, we first align their centroids and calculate the sum of squared distances between their landmarks. A smaller sum of squared distances indicates greater shape similarity.

After generating the initial estimate shapes, we have $N = n \cdot m \cdot G \cdot H$ training data. For all the examples in this paper, we set $m = 9$, $G = 5$ and $H = 4$.

Camera Calibration. The camera projection matrix describes a mapping from 3D points in camera coordinates to 2D points in an image. It is dependent on the internal parameters of the camera, and can be expressed as

$$\mathbf{Q} = \begin{pmatrix} f_x & \gamma & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{pmatrix}, \qquad (9)$$

where the parameters f_x and f_y represent the focal length in terms of pixel width and height, γ describes the skew between the x and y axes, and u_0 and v_0 represent the principal point, where the optical axis intersects the image plane. There exist many camera calibration algorithms (e.g., [Zhang 2000]) that can be used to compute these parameters with standard calibration targets (e.g., a planar chessboard pattern).

Currently we use a simple method to estimate $\mathbf{Q}$ directly from the user's setup images, without the need for special calibration targets. Specifically, we assume the camera to be an ideal pinhole camera, where $f = f_x = f_y$, $\gamma = 0$, and (u_0, v_0) is at the center of the image. This leaves a single unknown, f, in the matrix. Given a value of f, we can compute the user-specific blendshapes $\{B_i\}$ by solving Eq. (3) and (4), and then fit the face mesh by minimizing Eq. (8). As shown in Fig. 5, the fitting errors of face meshes in Eq. (8) form a convex function with respect to f. Since incorrect values of f lead to distortions in the projected points, the minimum error occurs at the true value of f, which was identified using [Zhang 2000]. We can solve for f with a binary search, and this simple technique was found in our experiments to give satisfactory results.

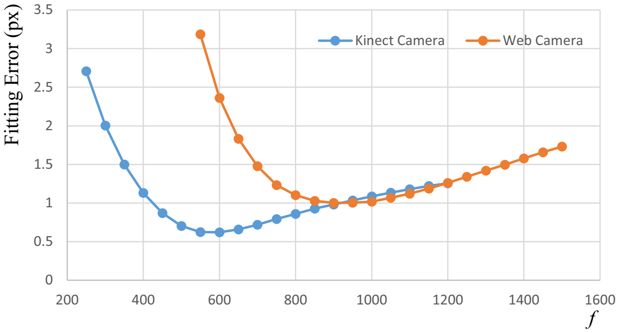

Figure 5: *Fitting errors using different f values for two cameras. Our calibration method computes $f = 580$ for the Kinect camera and $f = 960$ for the ordinary web camera. The ground truth values computed from [Zhang 2000] are 576 and 975 respectively. The relative errors are less than 2%.*

4.3 Training

With the N training data $\{(I_i, \mathbf{M}_i^a, S_i, S_i^c)\}$, we learn a regression function from S_i^c to S_i based on intensity information in the image I_i. We follow the two-level boosted regression approach (with T stages in the first level and K stages in the second level) proposed by Cao et al. [2012] to combine a set of weak regressors in an additive manner, but extend it to 3D. Each weak regressor computes a shape increment from image features and updates the current face shape according to Eq. (10). Each regressor is learnt by minimizing the sum of landmark alignment errors.

For the first level of the regressor, we generate a set of *index-pair* features that will be used by a collection of weak regressors at the second level. These index-pair features are computed from the appearance vector of a 3D facial shape (e.g., S_i^c). An appearance vector, which serves as a substitute for the face image in the regression analysis, is composed of the intensities at P randomly selected 3D facial points $\mathbf{p}$, each represented as the sum of a landmark position in S_i^c and an offset $\mathbf{d}_p$[1]. The intensity value of $\mathbf{p}$ is obtained from its corresponding 2D position in image I_i, computed as $\Pi_{\mathbf{Q_i}}(\mathbf{M}_i^a \mathbf{p})$. The intensity values of the P points are assembled to form the appearance vector V_i of 3D facial shape S_i^c. This procedure of calculating the appearance vector of S_i^c will be denoted as $App(I_i, \mathbf{M}_i^a, S_i^c, \{\mathbf{d}_p\})$. For each appearance vector V_i, we generate P^2 index-pair features by computing the differences between each pair of elements in V_i.

The second regression level employs a set of weak regressors which are collectively used to infer the 3D face shape. To learn a good regressor from the training data, effective index-pair features for the data need to be selected. We determine these features based on their correlation to the regression target, by calculating the difference between the ground truth shape S_i and the current shape S_i^c, projecting this difference to a random direction to produce a scalar, and then choosing the index-pair with the highest correlation with this scalar. This technique of random projections has been shown to be a powerful tool for dimensionality reduction [Bingham and Mannila 2001]. We repeat this process F times to yield F index-pair features, which together are used to build a primitive regressor structure called a *fern* [Dollar et al. 2010].

Algorithm 1 3D shape regression training

```
Input: N Training data $(I_i, \mathbf{M}_i^a, S_i, S_i^c)$
Output: Two-level boosted regressor
/* level one */
for $t = 1$ to $T$ do
    $\{\mathbf{d}_p^t\} \leftarrow$ randomly generate $P$ offsets
    for $i = 1$ to $N$ do
        $V_i \leftarrow App(I_i, \mathbf{M}_i^a, S_i^c, \{\mathbf{d}_p^t\})$
        Compute the $P^2$ feature values for $V_i$

    /* level two */
    for $k = 1$ to $K$ do
        for $f = 1$ to $F$ do
            $Y_f \leftarrow$ randomly generate a direction
            for $i = 1$ to $N$ do
                $\delta S_i \leftarrow S_i - S_i^c$
                $c_i \leftarrow \delta S_i \cdot Y_f$
            Find the index-pair with the highest correlation with
            $\{c_i\}$, and randomly choose a threshold value
        for $i = 1$ to $N$ do
            Calculate the features in $V_i$ using the $F$ index-pairs
            Compare the features to the thresholds and determine
            which bin the data belongs to
        for $i = 1$ to $2^F$ do
            Compute $\delta S_{b_i}$ according to Eq. (10)
            for each training data $l \in \Omega_{b_j}$ do
                $S_l^c \leftarrow S_l^c + \delta S_{b_i}$
```

In a fern, each of the F features is assigned a random threshold value between the minimum and maximum differences of element pairs in the appearance vectors, and these thresholds divide the space of index-pair feature values into 2^F bins. For each training data, we evaluate its index-pair feature values and determine which bin in the index-pair space it belongs to. After classifying all the training data in this way, we take the set of training data in each bin b, which we denote as Ω_b, and determine the regression output that would minimize its alignment error. Following [Cao et al. 2012], the regression output δS_b is calculated as

$$\delta S_b = \frac{1}{1 + \beta/|\Omega_b|} \frac{\sum_{i \in \Omega_b}(S_i - S_i^c)}{|\Omega_b|}, \tag{10}$$

where $|\Omega_b|$ is the number of training data in the bin, and β is a free shrinkage parameter that helps to overcome the problem of overfitting when there is insufficient training data in the bin.

Once we generate the fern, we update the current shapes for all training data. For all training data in each bin b, we add the regression output to the current shape: $S_i^c = S_i^c + \delta S_b$.

The regressor training procedure is iterated T times, each with K ferns, to progressively refine the regression output. Increasing the number of iterations in the two-level boosted regression (T, K) and the number of sample points P will bring more accuracy at the expense of greater computation. We empirically fixed the regression parameters to the following in our implementation: $T = 10, K = 300, P = 400, F = 5, \beta = 250$.

4.4 Run-time Regression

With this 3D shape regressor, we can obtain the 3D facial shape S for the image I in the current video frame using a few good initial estimates of S generated from the shape S' computed for the previous frame. As described in Algorithm 2, we first transform S' to its most similar shape S_r in the original shape set $\{S_i^o\}$ via a rigid rotation and translation $(\mathbf{M}^a)$. This transformation of S' to S'^* is intended to align S' (and thus the face shape S for the current frame) with the 3D shape space as much as possible. Then we find the L most similar shapes $\{S_l\}$ to the transformed shape S'^*

[1] We generate P points around the landmark vertices according to a Gaussian distribution. For each sample point, we find its nearest landmark vertex and calculate the offset $\mathbf{d}_p$.

Algorithm 2 Run-time regression

```
Input: Previous frame's facial shape $S'$, current frame's image $I$
Output: Current frame's facial shape $S$
Get the shape $S_r$ in $\{S_i^o\}$ most similar to $S'$
Find the rigid transformation $\mathbf{M}^a$ that best aligns $S'$ with $S_r$
$S'^* \leftarrow \mathbf{M}^a S'$
$\{S_l\} \leftarrow$ Choose $L$ shapes most similar to $S'^*$ in the training data
for $l = 1$ to $L$ do
    for $t = 1$ to $T$ do
        $V \leftarrow App(I, (\mathbf{M}^a)^{-1}, S_l, \{\mathbf{d}_p^t\})$
        for $k = 1$ to $K$ do
            Get the $F$ index-pairs recorded during training
            Calculate the $F$ appearance feature values for $V$
            Use the feature values to locate its bin $b$ in the fern
            Get $\delta S_b$ in $b$
            $S_l \leftarrow S_l + \delta S_b$
$S^* \leftarrow$ Compute the median shape of $\{S_l, 1 \leq l \leq L\}$
$S \leftarrow (\mathbf{M}^a)^{-1} S^*$
```

from among all the 3D shapes in the training data. Each chosen shape S_l is used as an initial shape estimate and is passed through the regressor.

In the first level of regression, we get the appearance vector V based on the image I, current shape S_l, the inverse of transformation matrix $\mathbf{M}^a$, and the offsets $\{\mathbf{d}_p^t\}$ recorded during training. In the second level of regression, we calculate the features according to the F index-pairs recorded in the fern, and compare the feature values with the recorded thresholds to locate its bin b, which contains δS_b. Finally, we update the current shape as $S_l = S_l + \delta S_b$.

Once we obtain the regression results for all $\{S_l\}$, we take the median shape as the final shape in the 3D shape space, and invert the transformation $\mathbf{M}^a$ to get the shape S for the current frame.

This run-time regression algorithm contains two elements designed to promote robust feature alignment that is free of drifting artifacts even in the case of large head rotations and exaggerated expressions. First, instead of using the shape S' from the previous frame as the initial estimate of the current shape, which is a common choice among video feature tracking algorithms, we use S' to obtain a set of similar shapes $\{S_l\}$ in the shape space to be used as initial shapes for regression. This helps the regression to better deal with uncertainty in S' and to avoid error accumulation.

Second, instead of directly choosing initial shapes similar to S' from the shape space, we choose shapes similar to the transformed shape S'^* for regression, and transform the regression result back. The reason for doing so is that regression without using the transformation M^a computes a linear interpolation of the shapes in the training data as the regression result. If the 3D facial shape for the current frame has a head orientation quite different from the shapes in the training data, linear interpolation cannot produce a good result because rotation is a nonlinear transformation. We solve this problem by computing a transformation M^a from S' to its most similar shape S_r in the original shape set $\{S_i^o\}$. Since face motion is smooth, S' and S should have similar orientations. Applying M^a to S' (and S) will align them with the shape space formed by the training data. Even if S_r and the generated initial shapes $\{S_l\}$ still have different orientations from S, the difference will be relatively small and the regression will be effective since small rotations can be well handled by linear interpolation.

Our experiments show that our regression algorithm can track 3D facial landmarks accurately and efficiently over different poses and expressions. In Fig. 6, we compare tracking results with and without the two aforementioned robustness elements.

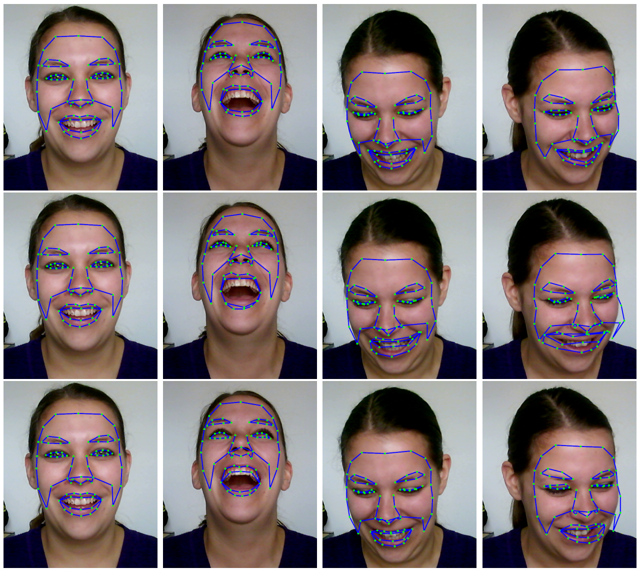

Figure 6: *Run-time regression results using our algorithm (top row). If we use the previous frame's shape as the initial shape directly, we may accumulate error which causes drift (middle row). If we omit the transformation M^a in regression, the linear interpolation cannot handle faces with large rotations (bottom row).*

5 Face Tracking

The real-time tracking process is driven by the facial shape regressor, which provides the 3D positions of landmarks in each video frame. Facial motion can be separated into two components: rigid head pose represented by the transformation matrix $\mathbf{M}$, and non-rigid expressions modeled by the expression coefficient vector $\mathbf{a}$. Both of these components can be solved by minimizing the following energy:

$$E_t = \sum_{k=1}^{75} \left\| \mathbf{M}(B_0 + \sum_{j=1}^{46} \alpha_j B_j)^{(v_k)} - S^{(k)} \right\|^2 , \qquad (11)$$

where $S^{(k)}$ is the 3D position of the k-th landmark in S and v_k is the corresponding vertex index on the face mesh.

Similar to [Weise et al. 2011], we add an animation prior to enhance temporal coherence in the tracking. Given the coefficient vectors for previous frames $\mathbf{A}_n = \{\mathbf{a}^{-1}, \mathbf{a}^{-2}, ..., \mathbf{a}^{-n}\}$, we combine them with the current frame's coefficients $\mathbf{a}$ into a single vector $(\mathbf{a}, \mathbf{A}_n)$, and formulate it as the following Gaussian mixture model:

$$p(\mathbf{a}, \mathbf{A}_n) = \sum_{s=1}^{S} \pi_s \mathcal{N}(\mathbf{a}, \mathbf{A}_n | \mu_s, Cov_s). \qquad (12)$$

This Gaussian mixture model is trained using pre-generated animation frames as in [Weise et al. 2011]. The mixture model is used to formulate a prior energy on $\mathbf{a}$ with respect to the previous frames:

$$E_{prior} = -\ln p(\mathbf{a}, \mathbf{A}_n). \qquad (13)$$

Adding the animation prior to Eq. (11), we can compute the transformation matrix and expression coefficients via

$$\arg\min_{\mathbf{M},\mathbf{a}} E_t + w_{prior} E_{prior}, \qquad (14)$$

where the weight w_{prior} is set to 1 in our experiments. This minimization problem can be solved using an iterative two-step approach. We first use the coefficients from the previous frame to initialize $\mathbf{a}$ and solve for the transformation matrix $\mathbf{M}$. This becomes a 3D registration problem between the regressed 3D facial shape and the face mesh computed using $\mathbf{a}$, which can be easily solved through singular value decomposition (SVD) on the cross-covariance matrix of the two point distributions [Besl and McKay 1992]. We then fix $\mathbf{M}$ and optimize $\mathbf{a}$. This can be performed using an iterative gradient solver. We precompute the gradients of Eq. (13) by the method described in [Weise et al. 2011], and use the gradient projection algorithm based on the BFGS solver [Byrd et al. 1995] to constrain the expression coefficients to lie between 0 and 1. We iteratively perform the two-step optimization until convergence. In our experiments, two iterations give satisfactory results. Note that unlike in solving Eq. (8), we do not need to update any vertex indices v_k in each iteration because they all correspond to internal landmarks as described in Sec. 3.

6 Experimental Results

We implemented our system on a PC with an Intel Core i7 (3.5GHz) CPU and an ordinary web camera (recording 640 × 480 images at 30 fps). The user interaction for data preprocessing is easily manageable, as the user needs only to adjust 2D landmark positions in the setup images. Tracking results (for both the rigid transformation and expression coefficients) can be transferred to any digital avatar with pre-created blendshapes (see Fig. 10). The simple requirements of the system make it well suited for interactive applications like computer games and virtual communication. Please see the supplementary video for a live demonstration of our system.

In the following, we first describe the user interaction and timings for the preprocessing stage, as well as the timings for the run-time algorithm. Next, we evaluate the accuracy of our 3D shape regression algorithm and compare it to previous methods. Finally we discuss limitations of our method.

User Interaction and Timings. In the setup stage, the user needs to perform a sequence of 60 facial poses/expressions. We show a sequence of rendered images of a standard face model with the different expressions, and ask the user to perform the head poses and expressions shown in each of the images. This simple capturing process takes less than 10 minutes even for first-time users. The automatic feature alignment algorithm [Cao et al. 2012] provides accurate 2D landmark positions for most facial features and images, but some of them still need manual adjustment. For a first-time user, this process takes about 25 minutes (for the first author of this paper, it takes less than 15 minutes). Note that the facial landmarks to be hand-labeled are relatively distinctive features, such as the face contour, eye boundaries, and mouth boundary. The 15 non-distinctive landmarks shown in Fig. 3(b)(d) are automatically computed by projecting a set of pre-defined vertices in the fitted mesh onto the image. Our algorithm is also robust to small errors in labeling. Other data preparation tasks (including user-specific blendshape generation, camera calibration and training data preparation) and the shape regression training are all automatically completed in less than 10 minutes. In total, the setup and preprocessing take less than 45 minutes for a novice user.

At run time, the shape regression is performed at 5ms per frame (with 15 initial shapes). Face tracking from the regressed shape takes about 8ms per frame. Overall the run-time performance of this algorithm is high (less than 15ms), because it is determined by the number of landmarks and initial shapes, and independently of the resolution of video frames. This makes our algorithm very promising for implementation on mobile devices. On the current PC, the system runs at over 24 fps.

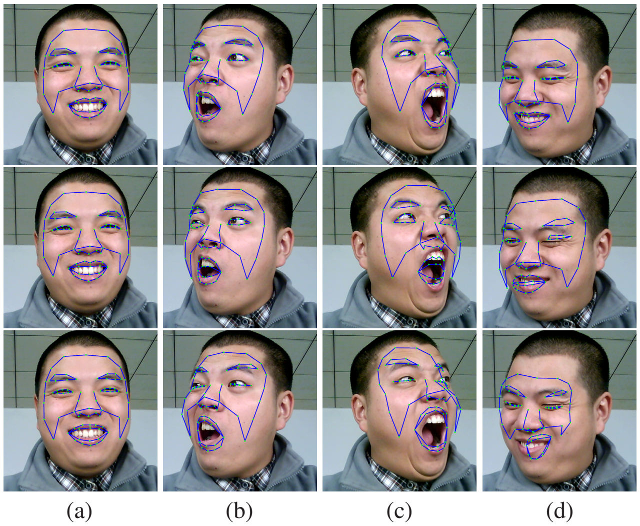

Figure 7: *Comparison of our 3D shape regression, 2D shape regression and optical flow. With the same training data, 3D shape regression (top row) can handle expressions and poses that are hard to represent by interpolation of 2D shapes (middle row). For these large head poses with expressions, optical flow results may exhibit drift (bottom row).*

Note that for the first video frame, we need to estimate the 3D facial shape and use it to initialize the regression. This is achieved by applying the 2D shape regressor [Cao et al. 2012] to automatically locate the landmark positions, and then using the 3D facial shape recovery algorithm described in Sec. 4.2. The same procedure can be performed when tracking fails.

Evaluation and Comparison. To evaluate the accuracy of our 3D shape regression algorithm, we compared it to ground truth obtained using a Kinect RGBD camera with manually labeled 2D landmark positions in each frame. While our algorithm processes only the RGB information from the Kinect, the depth data and the Kinect's projection matrix are used to determine the actual depth values of the labeled 2D landmarks. In this way, we can get the ground truth 3D facial shapes for comparison to our regressed shapes. Fig. 8 shows the depth values of a mouth corner from both the ground truth shapes and our regressed shapes over different frames (other landmarks have similar curves). It is shown that the depth estimated by our algorithm is very close to the depth acquired from the Kinect, with a difference of less than 10mm. The screen projections of landmarks from the ground truth and our regression also match closely, as shown in the supplementary video.

The 2D regression algorithm of [Cao et al. 2012] can be used to compute the 2D landmark positions for each video frame and can achieve accurate results when the face is oriented frontally (see Fig. 7(a), second row). However, if the face is oriented away from the front, the computed shapes may deviate significantly from the ground truth (see Fig. 7(c)(d), second row) as the shape space formed by the training shapes cannot represent these shapes using only linear interpolation. By contrast, our algorithm can generate much more accurate shapes (see Fig. 7, top row). It works in the 3D space and is capable of removing the effects of nonlinear pose rotations by aligning the current face to the shape space formed by the training shapes. Also, by using projections of 3D facial shapes to compute their appearance vectors in images, it can more accurately locate pixels with the same semantic meaning. Note that increasing the number of captured images and training data with different rotations can improve the accuracy of 2D regression. This, however, re-

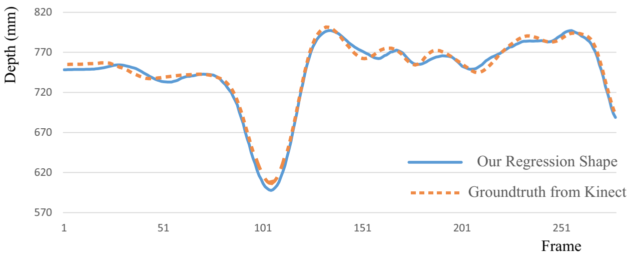

Figure 8: *Comparison of depth from 3D shape regression and ground truth from Kinect.*

RMSE	< 3 pixels	< 4.5 pixels	< 6 pixels
3D Regression	73.3%	80.8%	100%
2D Regression	50.8%	64.2%	72.5%
Optical Flow	20.8%	24.2%	41.7%

Table 1: *Percentages of frames with RMSE less than given thresholds for the tested video sequence.*

quires considerable data to handle all possible rotations, which not only is less practical to capture but also may lead to many more errors in manual landmark labeling. In Table 1, we measure the errors (in pixels) for the screen projection of landmarks of different algorithms compared with the ground truth positions. We also compare our algorithm with the face tracking method based on multilinear models [Vlasic et al. 2005], which links optical flow with the parameters of the face model. While multilinear models are powerful enough to represent different faces with different expressions, optical flow approaches may accumulate errors temporally and produce drifting artifacts in the tracking results in some situations, such as when the face moves quickly (see Fig. 7, bottom row). Our algorithm is more robust to fast motions as the facial shape is regressed with a broader set of initial shape estimates from the shape space.

Discussion. Some limitations of our shape regression approach arise from its reliance on facial appearance information in the video. Our system can deal with some partial occlusions, since distortion in a small part of the face appearance will not necessarily derail the user-specific shape regressor. However, our system may output incorrect shapes when there are larger occlusions (see Fig. 9 and the accompanying video). Once the occluder moves away, the regressor can quickly recover the correct shape. A possible way to deal with this problem to some degree is to include additional training data containing common occluders (e.g., hands). Another limitation is in handling dramatic changes in lighting. The regression algorithm can handle lighting changes to a modest extent, and it works well in situations where the background is changing and the lighting is not changing dramatically. But if the lighting environment differs substantially from that during the setup capture, the regression will not generate accurate shapes because face appearance can change significantly under different lighting conditions. This issue could be alleviated to some extent by capturing facial images in several typical environments, like in an office, hotel and outdoors, and training the regressor with them. This may broaden the range of lighting environments that can be handled.

With its use of depth data, the recent Kinect-based method in [Weise et al. 2011] is relatively more robust to occlusions and lighting changes. However, the Kinect camera has restrictions in its capture, since it does not operate effectively in outdoor environments with sunlight, and has a limited working range, meaning that the user cannot move too near or too far from the camera. Our experience with the commercial system Faceshift based on [Weise et al. 2011] indicates that the face needs to be within 1.5m to get satisfactory results. Our approach is not affected by these problems, as long as the face can be recognized in the video.

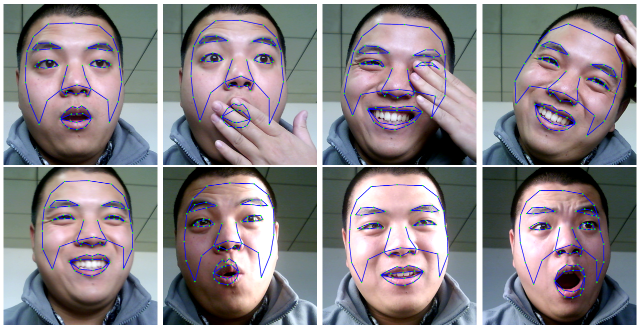

Figure 9: *Top row: regression results under occlusions. Bottom row: regression results under lighting changes.*

Currently our system employs a two-step procedure to track facial animations – first regressing the 3D positions of facial landmarks, and then computing the head poses and expression coefficients. It is possible to directly regress the rigid transformation and expression coefficients by adapting the training and run-time regression algorithms accordingly. This one-step approach could further improve the performance of our system as the number of transformation parameters and expression coefficients is much fewer than that of the landmarks. However, we did not take this one-step approach for several reasons. First, direct regression cannot guarantee that the expression coefficients are between 0 and 1. Simply clamping the coefficients may introduce undesirable results. Furthermore, it is not clear how to consider the animation prior in the regressor, which is critical for temporal coherence in the tracking. Finally, unlike our current two-step approach where the landmark positions (i.e., the regression target) are handled in the same manner, the one-step approach may need to deal with the rigid transformation and expression coefficients in different ways as they are in different spaces. For example, in the training process, when selecting effective index-pair features, the correlation to the head translation, rotation and expression coefficients needs to be computed with carefully-chosen weights to reflect their proper influences on the features. We would like to explore these problems in the future.

7 Conclusion

We presented a regression approach for 3D facial performance capture from 2D video. Our experiments show robust, real-time tracking results that compare favorably to related techniques. We believe our system significantly advances the state of the art in facial animation by providing a practical solution for ordinary users.

There are a couple of extensions to our system that we plan to investigate in future work. First, we are interested in removing the effects of lighting variations in both the training images and runtime video stream through intrinsic image decomposition. If this decomposition can be performed in real time, our algorithm can become more robust to large lighting changes without the use of additional training images. Another direction for future work is to estimate the lighting environment from the video stream, and utilize photometric techniques to recover fine-scale details of the face shape. Synthesis of these details could lead to more convincing facial animations.

Acknowledgements We thank Xuchao Gong, Shenyao Ke, Libin Hao and Yuan Du for making the digital avatars used in this paper, Marion Blatt and Steffen Toborg for being our face capture subjects, and the SIGGRAPH reviewers for their constructive comments. This work is partially supported by NSFC (No. 61003145 and No. 61272305) and the 973 program of China (No. 2009CB320801).

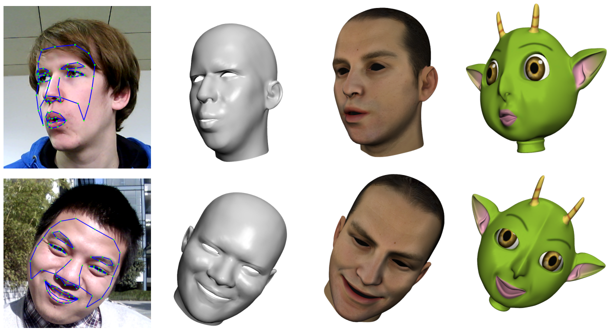

Figure 10: *Real-time performance-based facial animation. From left to right: a video frame, tracked mesh, two digital avatars.*

References

BEELER, T., BICKEL, B., BEARDSLEY, P., SUMNER, R., AND GROSS, M. 2010. High-quality single-shot capture of facial geometry. *ACM Trans. Graph. 29*, 4, 40:1–40:9.

BEELER, T., HAHN, F., BRADLEY, D., BICKEL, B., BEARDSLEY, P., GOTSMAN, C., SUMNER, R. W., AND GROSS, M. 2011. High-quality passive facial performance capture using anchor frames. *ACM Trans. Graph. 30*, 4, 75:1–75:10.

BESL, P., AND MCKAY, H. 1992. A method for registration of 3-d shapes. *IEEE Trans. Pattern Anal. Mach. Intell. 14*, 2, 239 –256.

BINGHAM, E., AND MANNILA, H. 2001. Random projection in dimensionality reduction: Applications to image and text data. In *Knowledge Discovery and Data Mining*, 245–250.

BLANZ, V., AND VETTER, T. 1999. A morphable model for the synthesis of 3d faces. In *Proceedings of SIGGRAPH*, 187–194.

BRADLEY, D., HEIDRICH, W., POPA, T., AND SHEFFER, A. 2010. High resolution passive facial performance capture. *ACM Trans. Graph. 29*, 4, 41:1–41:10.

BYRD, R. H., LU, P., NOCEDAL, J., AND ZHU, C. 1995. A limited memory algorithm for bound constrained optimization. *SIAM J. Sci. Comput. 16*, 5 (Sept.), 1190–1208.

CAO, X., WEI, Y., WEN, F., AND SUN, J. 2012. Face alignment by explicit shape regression. In *IEEE CVPR*, 2887–2894.

CAO, C., WENG, Y., ZHOU, S., TONG, Y., AND ZHOU, K. 2013. FaceWarehouse: a 3D Facial Expression Database for Visual Computing. *IEEE TVCG*, under revision.

CASTELAN, M., SMITH, W. A., AND HANCOCK, E. R. 2007. A coupled statistical model for face shape recovery from brightness images. *IEEE Trans. Image Processing 16*, 4, 1139–1151.

CHAI, J.-X., XIAO, J., AND HODGINS, J. 2003. Vision-based control of 3d facial animation. In *Symp. Comp. Anim.*, 193–206.

COOTES, T. F., IONITA, M. C., LINDNER, C., AND SAUER, P. 2012. Robust and accurate shape model fitting using random forest regression voting. In *ECCV*, VII:278–291.

DECARLO, D., AND METAXAS, D. 2000. Optical flow constraints on deformable models with applications to face tracking. *Int. Journal of Computer Vision 38*, 2, 99–127.

DEMENTHON, D. F., AND DAVIS, L. S. 1995. Model-based object pose in 25 lines of code. *Int. J. Comput. Vision 15*, 1-2, 123–141.

DOLLAR, P., WELINDER, P., AND PERONA, P. 2010. Cascaded pose regression. In *IEEE CVPR*, 1078–1085.

EKMAN, P., AND FRIESEN, W. 1978. *Facial Action Coding System: A Technique for the Measurement of Facial Movement*. Consulting Psychologists Press.

ESSA, I., BASU, S., DARRELL, T., AND PENTLAND, A. 1996. Modeling, tracking and interactive animation of faces and heads: Using input from video. In *Computer Animation*, 68–79.

HUANG, D., AND LA TORRE, F. D. 2012. Facial action transfer with personalized bilinear regression. In *ECCV*, II:144–158.

HUANG, H., CHAI, J., TONG, X., AND WU, H.-T. 2011. Leveraging motion capture and 3d scanning for high-fidelity facial performance acquisition. *ACM Trans. Graph. 30*, 4, 74:1–74:10.

KHOLGADE, N., MATTHEWS, I., AND SHEIKH, Y. 2011. Content retargeting using parameter-parallel facial layers. In *Symp. Computer Animation*, 195–204.

LEWIS, J. P., AND ANJYO, K. 2010. Direct manipulation blendshapes. *IEEE CG&A 30*, 4, 42–50.

LI, H., WEISE, T., AND PAULY, M. 2010. Example-based facial rigging. *ACM Trans. Graph. 29*, 4, 32:1–32:6.

MATTHEWS, I., XIAO, J., AND BAKER, S. 2007. 2D VS. 3D deformable face models: Representational power, construction, and real-time fitting. *Int. J. Computer Vision 75*, 1, 93–113.

PIGHIN, F., HECKER, J., LISCHINSKI, D., SZELISKI, R., AND SALESIN, D. H. 1998. Synthesizing realistic facial expressions from photographs. In *Proceedings of SIGGRAPH*, 75–84.

PIGHIN, F., SZELISKI, R., AND SALESIN, D. 1999. Resynthesizing facial animation through 3d model-based tracking. In *Int. Conf. Computer Vision*, 143–150.

SARAGIH, J., LUCEY, S., AND COHN, J. 2011. Real-time avatar animation from a single image. In *AFGR*, 213–220.

SEO, J., IRVING, G., LEWIS, J. P., AND NOH, J. 2011. Compression and direct manipulation of complex blendshape models. *ACM Trans. Graph. 30*, 6.

VLASIC, D., BRAND, M., PFISTER, H., AND POPOVIĆ, J. 2005. Face transfer with multilinear models. *ACM Trans. Graph. 24*, 3, 426–433.

WEISE, T., LI, H., GOOL, L. V., AND PAULY, M. 2009. Face/off: Live facial puppetry. In *Symp. Computer Animation*, 7–16.

WEISE, T., BOUAZIZ, S., LI, H., AND PAULY, M. 2011. Realtime performance-based facial animation. *ACM Trans. Graph. 30*, 4 (July), 77:1–77:10.

WILLIAMS, L. 1990. Performance driven facial animation. In *Proceedings of SIGGRAPH*, 235–242.

XIAO, J., CHAI, J., AND KANADE, T. 2006. A closed-form solution to non-rigid shape and motion recovery. *Int. J. Computer Vision 67*, 2, 233–246.

YANG, F., WANG, J., SHECHTMAN, E., BOURDEV, L., AND METAXAS, D. 2011. Expression flow for 3D-aware face component transfer. *ACM Trans. Graph. 30*, 4, 60:1–60:10.

ZHANG, L., SNAVELY, N., CURLESS, B., AND SEITZ, S. M. 2004. Spacetime faces: high resolution capture for modeling and animation. *ACM Trans. Graph. 23*, 3, 548–558.

ZHANG, Z. 2000. A flexible new technique for camera calibration. *IEEE Trans. Pattern Anal. Mach. Intell. 22*, 11, 1330–1334.