Video Based Face Tracking and Animation

Changwei Luo[1], Jun Yu[1], Zhigang Zheng[1],
Bin Cai[2], Lingyun Yu[1], and Zengfu Wang[1,2(✉)]

[1] Department of Automation,
University of Science and Technology of China, Hefei, China
luocw@mail.ustc.edu.cn
[2] Institute of Intelligent Machines,
Chinese Academy of Sciences, Hefei, China
zfwang@ustc.edu.cn

Abstract. We propose a system for video based face tracking and animation. With a single video camera, our system can accurately track the facial feature points of a user, and transfer the tracked facial motions to the avatar's face. We use constrained local model (CLM) to track the feature points. The original CLM only makes use of local texture and performs an exhaustive local search around the current estimate of feature points. This often leads to local minima. To overcome this problem, we incorporate the global texture into CLM. The improved CLM not only gives discriminative capability to each feature point, but also gives good match to the whole texture. After obtaining the 2D positions of the feature points, we estimate blendshape coefficients based on a set of user-specific 3D key shapes. Finally, facial animations are created using blendshape interpolation. Experiments demonstrate the effectiveness of our system.

Keywords: Face tracking · Facial animation · Local texture · Global texture

1 Introduction

Generating 3D facial animation from a video is useful in many applications. Taking video games for an example, instead of using a keyboard or a mouse, the players can control the facial expression of a virtual hero by performing desired facial actions in front of a camera. This would greatly increase the interest of the game players. Besides, video conference has become an efficient means to achieve collaboration over long distances. However, there are several disadvantages for video conference. A crucial disadvantage is the lack of anonymity. Unlike text or voice based conversational systems, video conference systems will immediately reveal a person's face. This is unacceptable for many applications where anonymity is required. An attractive solution for this problem is using avatars or virtual faces.

Various methods have been proposed for creating avatar animation from videos [1–5]. Although these methods have different acquisition systems and

Y.-J. Zhang (Ed.): ICIG 2015, Part III, LNCS 9219, pp. 522–533, 2015.
DOI: 10.1007/978-3-319-21969-1_47

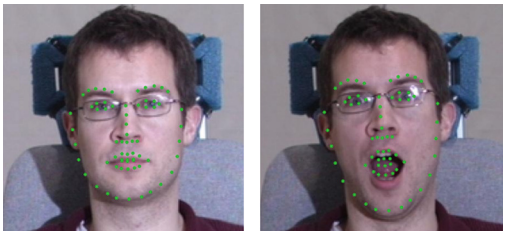

Fig. 1. Two examples of CLM fitting. The images are from Multi-PIE database [6]

processing pipelines, they share many fundamental principles. In [2,3], stereo cameras are used for facial performance capture. These systems require careful calibration and are not easy to use. In this paper, we focus on video based facial animation using a single camera.

Synthesizing video based facial animation require accurate face tracking. In [7], optical flow is applied for facial feature tracking. Tracking by optical flow is unreliable especially for less salient landmarks. Geometric priors as well other constraints are often incorporated to prevent drifts [8,9]. The active appearance model (AAM) is popular for face tracking [10,11]. However, the tracking accuracy is insufficient. Compared with AAM, the constrained local model (CLM) [12] has shown good performance for face tracking. CLM outperforms AAM in that it is discriminative and generalize well to unseen texture variation. It also offers greater invariance to illumination variation. In [13], the displaced dynamic expression (DDE) model is proposed for real-time face tracking. Although the DDE model has shown good performance, the training of a DDE model is tedious.

In our system, CLM is used to track the feature points in a video. The original CLM [12] only makes use of local texture and performs an exhaustive local search around the current estimate of feature points. This often leads to local minima. To improve the tracking accuracy, we incorporate the global texture of AAM into CLM. The improved CLM not only gives discriminative capability to each landmark, but also gives good match to the whole texture. Our method effectively incorporates shape prior, local texture and global texture.

After obtaining the 2D positions of the feature points, we estimate blendshape coefficients based on a set of user-specific 3D key shapes. The 3D key shapes are automatically generated based on the user's face model and the target face model. Finally, facial animations are created using blendshape interpolation.

2 Facial Tracking

Given the video of a person's face, we use constrained local model [12] to track the feature points. For face images with neutral facial expressions, the accuracy of CLM fitting is relatively high. However, for face images with various facial expressions, CLM is not able to accurately locate all the feature points. This is the case especially for feature points on the lip contours (see Fig. 1). The reason

is that the texture around the lips exhibits large variations. The local detectors (i.e., the patch experts) of CLM fail to discriminate these feature points. To improve the tracking accuracy, we add global texture of AAM to CLM.

2.1 Constrained Local Model

The CLM consists of a shape model and set of patch experts. The shape model describes the rigid and non-rigid shape variations. A face shape is defined on a set of feature points (landmarks), and is represented by a vector $s = (\mathrm{x}_1, \mathrm{x}_2, \cdots, \mathrm{x}_v)^T$. s contains the coordinates of v feature points, $\mathrm{x}_i = (x_i, y_i)$ is the 2D location of the ith feature point. The shape s can be expressed as a linear combination of mean shape s_0 and n orthonormal bases s_i,

$$s = s_0 + \sum_{i=1}^{n} s_i p_i \tag{1}$$

where $\mathrm{p} = [p_1, p_2, ..., p_n]$ are the shape parameters. The feature point x_i of shape s is then placed in the image frame by applying a 2D similarity transform $\mathrm{F}(\mathrm{x}_i; \mathrm{q})$.

$$F(\mathrm{x}_i, \mathrm{q}) = \begin{bmatrix} q_1 & -q_2 \\ q_2 & q_1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} q_x \\ q_y \end{bmatrix} \tag{2}$$

where $\mathrm{q} = [q_1, q_2, q_x, q_y]$ are the pose parameters.

For a CLM with v feature points, there are also v patch experts, each corresponding to a feature point. The ith patch expert is used to calculate the probability of alignment for the ith feature point $p(l_i = 1|\mathrm{x}_i, I)$.

$$p(l_i|\mathrm{x}_i, I) = \frac{1}{1 + \exp\{\rho \cdot C_i(\mathrm{x}_i; I) + \varepsilon\}} \tag{3}$$

where I denotes the input image. $l_i \in \{1, -1\}$ is a discrete random variable denoting whether the ith feature point is aligned or misaligned. The constants ρ and ε are learned through cross-validation [12]. $C_i(\mathrm{x}_i; I)$ is the output of a classifier. Here we use the linear support vector machine (SVM).

$$C_i(\mathrm{x}_i; I) = w_i^T \cdot \psi(\mathrm{x}_i) + b_i \tag{4}$$

where w_i^T and b_i are the weights and biases of the SVM classifier, $\psi(\mathrm{x}_i; I)$ is an image patch centered at x_i.

By performing an exhaustive local search within the search window Ψ_i, we obtain a response map for each feature point. Each response map is approximated by a Gaussian distribution $N(\mathrm{x}_i; \mathrm{u}_i, \Sigma_i)$, u_i and Σ_i are the mean and covariance of the Gaussian distribution, respectively. u_i can be chosen as the locations with maximum response in the response map, and

$$\Sigma_i = \sum_{\mathrm{x} \in \Psi_i} \frac{p(l_i = 1|\mathrm{x}, I)}{\sum_{\mathrm{y} \in \Psi_i} p(l_i = 1|\mathrm{y}, I)} (\mathrm{x} - \mu_i)(\mathrm{x} - \mu_i)^T \tag{5}$$

The parameters p and q of CLM are solved by minimizing the following objective function [12,14]:

$$Q_{clm} = \frac{1}{2} \cdot ||\mathrm{p}||^2_{\Lambda^{-1}} + \frac{1}{2} \cdot \sum_{i=1}^{v} ||\mathrm{F}(\mathrm{x}_i; \mathrm{q}) - \mathrm{u}_i||^2_{\Sigma_i^{-1}} \tag{6}$$

2.2 Combining Global Texture of AAM

The texture of AAM is defined within the mean shape s_0, and is also called shape normalized texture. Figure 2 shows the process of generating shape normalized texture. To reduce the space of texture variation and the computation load, only only part of the global texture is used (see Fig. 2(d)).

Texture variation is modeled by the linear combination of mean texture A_0 and m orthonormal vectors A_i.

$$A(\mathrm{x}) = A_0(\mathrm{x}) + \sum_{i=1}^{m} c_i A_i(\mathrm{x}) \tag{7}$$

where $\mathrm{c} = [c_1, c_2, ..., c_m]$ is the texture parameters.

Fig. 2. The process of generating shape normalized texture. (a) The mean shape. (b) An example facial image. (c) Shape normalized texture. (d) Only the texture around the lips and the eyes is used

To combine the texture of AAM into CLM, we define an improved objective function of CLM as follows:

$$Q = (1 - \omega_a) Q_{clm} + \omega_a K \cdot Q_{aam} \tag{8}$$

where

$$Q_{aam} = \frac{1}{2} ||A_0(\mathrm{x}) + \sum_{i=1}^{m} c_i A_i(\mathrm{x}) - I(W(\mathrm{x}; \mathrm{p}, \mathrm{q}))||^2 \tag{9}$$

K is a scaling factor that makes Q_{aam} and Q_{clm} commensurate. ω_a is a weight that balances Q_{aam} and Q_{clm}. $W(\mathrm{x}; \mathrm{p}, \mathrm{q})$ is the piecewise affine warp function.

The objective function (8) is minimized using Gauss-Newton gradient descent method. Give the initial estimate of p, we iteratively solve for increments to the parameter Δp, and then update p $\leftarrow$ p + Δp.

$$\Delta \mathrm{p} = -H^{-1} \cdot \frac{\partial Q}{\partial \mathrm{p}} \tag{10}$$

where H is the Hessian matrix, $\frac{\partial Q}{\partial \mathrm{p}}$ is the derivative of Q with respect to p.

$$\frac{\partial Q}{\partial \mathrm{p}} = (1-\omega_a)\frac{\partial Q_{clm}}{\partial \mathrm{p}} + \omega_a K \cdot \frac{\partial Q_{aam}}{\partial \mathrm{p}} \tag{11}$$

For the Q_{clm} term, we have

$$\frac{\partial Q_{clm}}{\partial \mathrm{p}} = \Lambda^{-1}\mathrm{p} + \sum_{i=1}^{v} J_i^T \Sigma^{-1}(\mathrm{F}(\mathrm{x}_i;\mathrm{q}) - \mathrm{u}_i) \tag{12}$$

where $J_i = \frac{\partial \mathrm{F}(\mathrm{x}_i;\mathrm{q})}{\partial \mathrm{p}^T}$. For the Q_{amm} term, we have

$$\frac{\partial Q_{aam}}{\partial \mathrm{p}} = SD \cdot [I(W(\mathrm{x};\mathrm{p},\mathrm{q})) - A_0(\mathrm{x})] \tag{13}$$

$SD = [SD_1, SD_2, ..., SD_n]^T$, and

$$SD_j = \nabla I \frac{\partial W}{\partial p_j} - \sum_{i=1}^{m} [{A_i}^T(\mathrm{x}) \cdot \nabla I \frac{\partial W}{\partial p_j}] \cdot A_i(\mathrm{x}) \tag{14}$$

The Hessian matrix H is computed as follows,

$$H = (1-\omega_a)H_c + \omega_a K \cdot H_a \tag{15}$$

where

$$H_a = SD \cdot (SD)^T \tag{16}$$

$$H_c = \Lambda^{-1} + \sum_{i=1}^{v} J_i^T \Sigma_i^{-1} J_i \tag{17}$$

The initial estimate of p can be obtained by using the results of CLM fitting. We solve for q using a similar procedure.

2.3 Adaptive Control of Weight

The weight ω_a on the Q_{aam} term should be controlled appropriately to ensure better performance. In the early iterations, the value of Q_{aam} is usually very large. The Q_{aam} term should have little influence. In the later iterations, the search points are closer to the true landmark points and the Q_{aam} term becomes

much smaller, the weight on Q_{aam} should be increased. Therefore, we use the following function to adaptively control the weight:

$$\omega_a^{(i)} = \exp(-a \cdot (\frac{Q_{aam}^{(i-1)}}{Q_0})^2) \tag{18}$$

where a is a constant. $\omega_a^{(i)}$ is the weight for the ith iteration ($i = 1, 2, 3, ...$). $Q_{aam}^{(i-1)}$ is valuated at the ith iteration. Q_0 corresponds to the initial estimation, and $Q_{aam}^{(0)} = Q_0$. Thus, once $\omega_a^{(1)}$ is known, a can be determined as follows,

$$a = -\ln \omega_a^{(1)} \tag{19}$$

3 Facial Animation

We create facial animation using a blendshape model. In our system, 39 blendshapes are used. b = [b_1,b_2,...,b_{39}] is the blendshape coefficient vector. To synthesize facial animation, the tracked facial motions need to be transformed to blendshape coefficients.

Since the face geometry of the target face model is usually different from that of the user, blendshapes of the target face model can not be directly fitted to the tracked feature points.

To accurately estimate the blendshape coefficients, blendshapes of the target character need to be adapted to a specific user. In [4], the user-specific blendshapes are generated based on a pre-built bilinear face model. Building the bilinear face model is expensive. In this paper, we estimate the blendshape coefficients based on a set of user-specific 3D key shapes.

Firstly, we capture a frontal neutral facial image of the user, and locate v ($v = 66$) feature points. Given the x,y-coordinates of the v feature points, we recover their z-coordinates using the method of [15]. The x, y, z-coordinates of these feature points are concatenated to form a 3D base shape K_0 for a specific user. The feature points are triangulated to create a mesh. The mesh is also referred as the user's face model. Next, we select v ($v = 66$) feature vertexes on the target face model. These feature vertexes are also triangulated according to the feature points. Finally, the deformations of the feature vertexes can be transferred to feature points by means of deformation transfer [16], and 39 3D key shapes K = [K_1,...,K_{39}] are obtained. Figure 3 shows an example of generated user's face model and the 3D key shapes.

Based on the 3D key shapes and the tracked facial shape s, blendshape coefficients are calculated by fitting the 3D key shapes to s [17]. Using a weak-perspective camera model, the fitting procedure is minimizing the following energy,

$$E = \sum_{i=1}^{v} \left\| \sigma R \cdot (K_0 + \sum_{j=1}^{39} b_j K_j)^{(i)} + T - s^{(i)} \right\|^2 \tag{20}$$

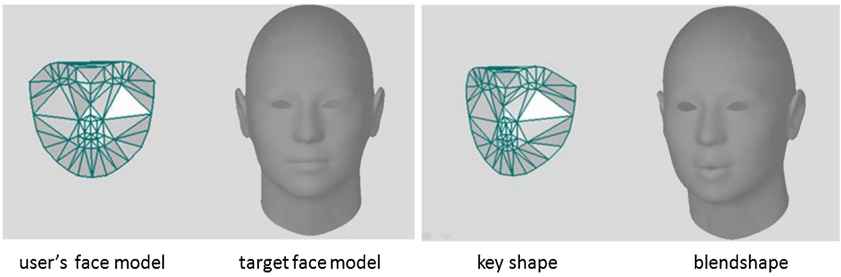

Fig. 3. From left to right: the user's face model, the target face model, an example of generated 3D key shapes, the corresponding blendshape

where σ is a scaling factor, R is the first two rows of a 3D rotation matrix and T is a translation vector. $s^{(i)}$ is the 2D position of the ith feature point in s.

Similar to [4], an animation prior is added to the fitting procedure to enhance temporal coherence. Given the blendshape coefficient vectors for h previous frames B_h=[$\mathrm{b}^{-1}, \mathrm{b}^{-2}, \cdots, \mathrm{b}^{-h}$], we combine B_h with the current frame's weight vector b into a single vector $(\mathrm{b}, \mathrm{B}_h)$. The probability distribution of $(\mathrm{b}, \mathrm{B}_h)$ is modeled with a Gaussian mixture model $N(\mathrm{b}, \mathrm{B}_h)$. The Gaussian mixture model is trained using pre-generated animation frames [18].

Finally, we estimate the blendshape coefficients b via

$$\arg\min \; E - \omega_1 \ln N(\mathrm{b}, \mathrm{B}_h) \tag{21}$$

where ω_1 is a constant. In our experiments, $\omega_1 = 0.5$, $h = 2$. We use an iterative two-step approach to solve the pose parameters (σ, R and T) and the blendshape coefficients b. Firstly, we fix the blendshape coefficients and solve for the pose parameters using Levenberg-Marquardt algorithm [19]. Secondly, we fix the pose parameters and solve for the blendshape coefficients using the gradient projection algorithm based on BFGS solver [20].

4 Experiments

Our system is implemented in C++ and parallelized using OpenMP. The openCV library is used for image processing. In our experiments, we first evaluate the fitting performance of the improved CLM. Then we show some generated animations.

Training. More than 800 frontal facial images are used to train our model. The improved CLM requires training a shape model, a set of patch experts. It also requires training a texture model. The shape model and the texture model are obtained by principal component analysis. The patch experts are set to (11×11)-pixels in size, and trained using positive examples and negative examples. Positive examples are obtained from the images patches centered at the

true feature points, while negative examples are obtained by sampling patches shifted away from the ground truth.

Figure 4 shows an example of the response maps obtained from the trained patch experts. We can see that the response map centered at the eye corner has good discriminative ability. The response map centered at the mouth corner has multiple peaks. This makes the local detector difficult to find the ground truth feature point. By incorporating global texture into CLM, it is possible to accurately locate the feature points as more texture information is provided. To handle large head rotations, we also built two models for left and right profile images.

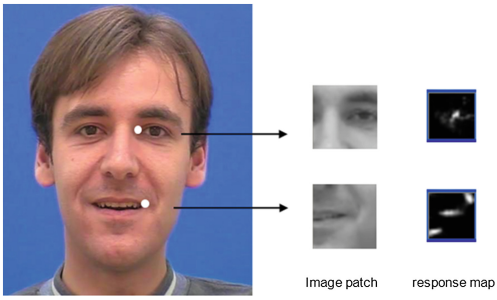

Fig. 4. An example of the response maps obtained from the trained patch experts

Determining the Weight ω_a . The weight ω_a for the global texture is determined experimentally. Firstly, we set ω_a to 0, 0.1, 0.2,...,0.9, 1.0. In each case, we evaluate the fitting accuracy using 180 testing images. The fitting accuracy is measured by the root mean square error (RMSE) between the search shape and the manually labeled shape. The RMSE is calculated for each testing image, and then the average RMSE is calculated on all testing images. Figure 5 shows the

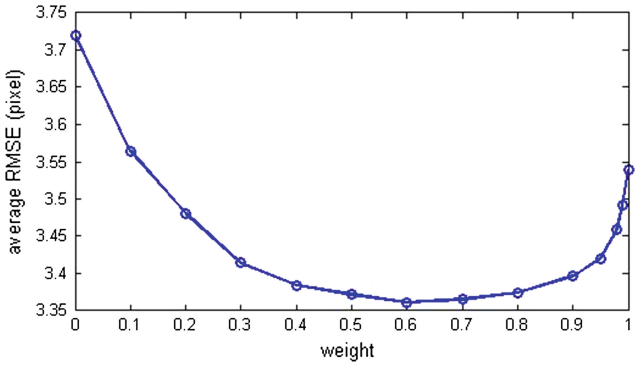

Fig. 5. The average RMSE for different values of ω_a.

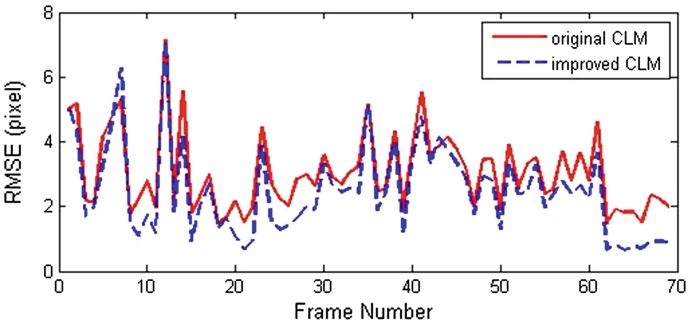

Fig. 6. Comparison between the original CLM and the improved CLM.

average RMSE at each ω_a. We can see that the RMSE reduces significantly after incorporating global texture. It is also shown that the smallest average RMSE is 3.36 pixels with $\omega_a = 0.6$. Thus, to adaptively control the weight, we can set $\omega_a^{(1)}$ to 0.6, and Eq. (19) gives $a = 0.51$. Then we evaluate the fitting accuracy for the case where adaptively controlled weight is used. The resulting average RMSE is 3.22 pixels. We can see that the adaptively controlled weight leads to a further decrease of average RMSE at about 4.17 %.

Tracking. We have also evaluated the performance of the improve CLM on FGNet talking face video (http://www-prima.inrialpes.fr/FGnet/data/01-TalkingFace/talking_face.html). The results are shown in Fig. 6. It is shown that the fitting error of the improved CLM is much smaller than that of original CLM [12]. Figure 7 shows some selected tracking results from FGNet talking face video and another video.

Fig. 7. Some tracking results of the proposed method from two videos.

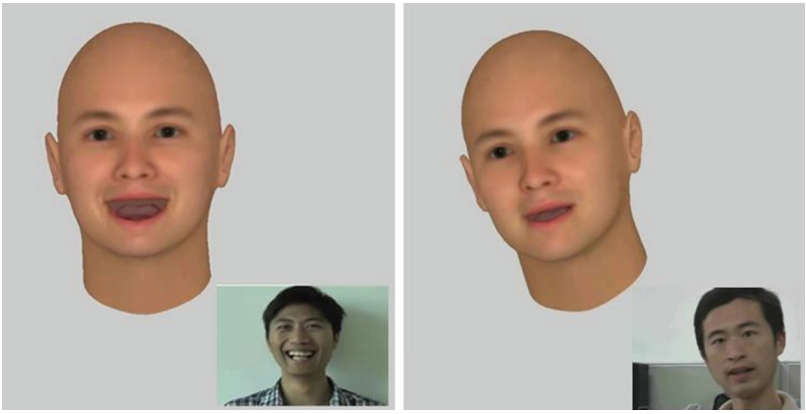

Fig. 8. Face tracking and animation for different users.

Animation. In the setup stage, a frontal facial image with neutral expression is captured for the user. Based on this image, the system automatically generates a set of user-specific 3D key shapes. At run time, facial features are tracked using the improved CLM. To speed up the tracking process, the weight ω_a can stop updating after one or two iterations. The tracked feature points are then used to compute blendshape coefficients. Our system runs at about 6 fps. Figure 8 shows the results of two users controlling and animating the facial expressions of an avatar. It is shown that the animations are quite realistic.

User Study. A user study is carried out to evaluate our system. Several animations are first generated from original facial video files using our system. The animations are then presented to 20 human subjects. The subjects are generally not connected with the topic. For each animation, the subjects are asked

Table 1. Questions in the questionnaire and the mean score for the subjective test

Questions	Mean score
1. Do the facial movements of the avatar look natural and realistic?	8.2
2. Do you recognize the avatar's facial expressions?	7.6
3. To what extent do the synthesized facial movements follow those in the original videos?	7.8

Table 2. General impression of our system from a few tested subjects

1. The system is very interesting.
2. I fell comfortable while interacting with the avatar. The avatar looks alive.
3. Sometimes the movements of eyes and lips are not well transferred to the avatar.
4. Except human avatars, non-human avatars should be added to the system.

to answer previously prepared questions (see Table 1). In these questions, the score is graded on the scale from 10 to 1. The higher score corresponds to more positive answers. The mean scores for these questions are also shown in Table 1.

In addition, the subjects are asked to interact with our system and then give the general impression of our system. Some interesting remarks from the tested subjects are shown in Table 2. These remarks point out our next goals.

5 Conclusion

We describe a system for video based face tracking and animation. To improve the tracking accuracy, global texture is incorporated into a constrained local model. To create facial animation, tracked face motions are transferred the target face based on a set of user-specific 3D key shapes. Experimental results show that the proposed system is effective for creating video based facial animation.

Acknowledgements. This work was supported by National Natural Science Foundation of China (61472393).

References

1. Rhee, T., Hwang, Y., Kim, J., Kim, C.: Real-time facial animation from live video tracking. In: Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (2011)
2. Zhang, L., Snavely, N., Curless, B., Seitz, S.: Spacetime faces: high resolution capture for modeling and animation. ACM Trans. Graph. **23**(3), 548–558 (2004)
3. Jiang, J., Zeng, M., Liang, B., Liu, X.: High quality binocular facial performance capture from partially blurred image sequence. In: IEEE International Conference on Computer-Aided Design and Computer Graphics (CAD/Graphics), pp. 196–203 (2013)
4. Cao, C., Weng, Y., Lin, S., Zhou, K.: 3D shape regression for real-time facial animation. ACM Trans. Graph. **32**(4), 41:1–41:10 (2013). Article No. 41
5. Luo, C., Yu, J., Wang, Z.: Synthesizing performance-driven facial animation. Acta Automatica Sinica **40**(10), 2245–2252 (2014)
6. Gross, R., Matthews, I., Cohn, J., Kanade, T., Baker, S.: Multi-pie. Image Vis. Comput. **28**(5), 807–813 (2010)
7. Chai, J., Xiao, J., Hodgins, J.: Vision-based control of 3D facial animation. In: Eurographics/SIGGRAPH Symposium on Computer Animation (2003)
8. Decarlo, D., Metaxas, D.: Optical flow constraints on deformable models with applications to face tracking. Int. J. Comput. Vis. **38**(2), 99–127 (2000)
9. Zhang, W., Wang, Q., Tang, X.: Real time feature based 3-D deformable face tracking. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part II. LNCS, vol. 5303, pp. 720–732. Springer, Heidelberg (2008)
10. Chen, Y., Yu, F., Ai, C.: Sequential active appearance model based on online instance learning. IEEE Sig. Process. Lett. **20**(6), 567–570 (2013)
11. Gao, X., Su, Y., Li, X., Tao, D.: A review of active appearance models. IEEE Trans. Sys. Man Cyber. Part C **40**, 145–158 (2010)

12. Wang, Y., Lucey, S., Cohn, J.: Enforcing convexity for improved alignment with constrained local models. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 1–8 (2008)
13. Cao, C., Hou, Q., Zhou, K.: Displaced dynamic expression regression for real-time facial tracking and animation. ACM Trans. Graph. **33**(4), 43:1–43:10 (2014). Article No. 43
14. Saragih, J., Lucey, S., Cohn, J.: Deformable model fitting by regularized landmark mean-shift. Int. J. Comput. Vis. **91**(2), 200–215 (2011)
15. Saragih, J., Lucey, S., Cohn, J.: Real-time avatar animation from a single image. In: IEEE International Conference on Automatic Face and Gesture Recognition, pp. 117–124 (2011)
16. Summer, R., Popovic, J.: Deformation transfer for triangle meshes. ACM Trans. Graph. **22**(3), 399–405 (2004)
17. Chuang, E., Bregler, C.: performance driven facial animation using blendshape interpolation. Technical report, Stanford University (2002)
18. Weise, T., Bouaziz, S., Li, H., Pauly, M.: Realtime performance-based facial animation. In: Proceedings SIGGRAPH (2011)
19. More, J.: The levenberg-marquardt algorithm: implementation and theory. In: Watson, G.A. (ed.) Numerical Analysis. Lecture Notes in Mathematics, vol. 630, pp. 105–116. Springer, Heidelberg (1978)
20. Byrd, R., Lu, P., Nocedal, J., Zhu, C.: A limited-memory algorithm for bound constrained optimization. SIAM J. Sci. Comput. **16**, 1190–1208 (1995)