REVEALING SYNTHETIC FACIAL ANIMATIONS OF REALISTIC CHARACTERS

Duc-Tien Dang-Nguyen, Giulia Boato, Francesco G.B. De Natale

Department of Information and Computer Science - University of Trento
via Sommarive 5, 38123 Trento - Italy
{dangnguyen, boato}@disi.unitn.it; denatale@ing.unitn.it

ABSTRACT

Given the recent development of advanced multimedia techniques able to support the creation of realistic computer generated characters, there is the parallel need of automatic tools allowing users to verify the source of the multimedia data they are observing, thus discriminating between artificial and natural information. In this paper, we focus on video representing human beings and we propose a novel method to identify computer generated characters by analysing the evolution of the face model in chronological order. Experimental results show that photorealistic facial animations, which are usually performed following fixed patterns, can be distinguished from natural ones, which follow much more complicated and various geometric distortions.

Index Terms— Computer Generated Animations, 3D Face Modelling, Video Forensics

1. INTRODUCTION

With the development of innovative multimedia technologies, the realism of computer generated (CG) characters has achieved a very high quality level. Non-existing subjects or situations can be easily generated. This is good for film and games production, but in a daily life context, it raises the need of advance tools supporting users in the identification of artificial data which may not represent reality.

Since 2005, the research community on multimedia forensics put a lot of effort in developing methods to distinguish CG and natural data, focusing mainly on images: by estimating statistical differences in wavelet-based decomposition [1][2]; by analysing physical variances, e.g., local patch statistics, fractal and quadratic geometry, surface gradient [3]; by evaluating the noise introduced by the recording device [4]; or by combining various data in a hybrid approach [5].

The application of these techniques to the problem of identifying CG characters does not allow to achieve very good performances. Thus, recently a couple of methods were developed by exploiting face-specific information. In 2012, a geometric approach supporting the distinction of CG and real human faces has been presented in [6] based on the idea that natural faces are more asymmetric with respect to synthetic ones, thus exploiting face asymmetry as a discriminative feature. Another method to distinguish between CG and real characters in videos by analyzing facial expressions has been proposed in [7]. This is based on the idea that facial expressions in CG characters follow a repetitive pattern, while in natural faces the same expression is usually produced in similar but not equal ways. However, this approach requires the analysis of multiple similar expression instances, which are not easy to be present in video of real situations. Moreover, since it works only on 2D information, only a limited number of face instances (closed to frontal view) can be exploited.

Given the difficulties of current approaches, we propose here a model-based method which allow to deal with normal behaviours of represented facial animation, where characters are moving, i.e., turning and rotating their faces. This analysis of the 3D model allow to deal more easily with human faces, which are various, deformable and can appear in multi ways depending on expression, lightning condition, poses, etc. In particular, we propose to study the evolution in chronological order of the 3D model of the analysed character, demonstrating that its variations allow to reveal synthetic animation. Indeed, facial animation following fixed patterns can be distinguished from natural ones which follow much more complicated and various geometric distortion, i.e., bigger variations in the 3D model deformation.

The rest of this paper is organized as follows: the proposed method is described in section 2, experimental results are reported in section 3, while section 4 draws some conclusions.

2. THE PROPOSED METHOD

In order to generate realistic facial animations, facial properties are changed by following some specific rules or movements. The idea behind this work is to distinguish synthetic characters in videos from natural ones by exploiting the analysis of such patterns.

In [7], a CG character could be identified by analyzing the variations of facial feature points comparing similar expressions, since variations in CG characters are smaller with respect to real persons performing the same animation. However, it is uneasy to perform such a measurement. Differences of facial feature points strongly depend on the selected animations, e.g., differences on feature points on the lips in a sad expression are not comparable to the differences in a happy expression, thus requiring the analysis of different sets of points for each single expression. Moreover, extracting robust facial feature points when characters are changing their poses is a nontrivial problem.

Here, we propose a novel way to analyze the evolution in time of the selected face, by estimating the 3D face model and studying its variations during the video. The reasons behind this solution are: (i) The face model is less dependent on changes of the face pose; (ii) Meaningful information about the whole face is taken into account instead of distinct feature points; (iii) The face model reconstruction does not require all facial feature points, which are not always visible due to occlusion or lighting condition, and can be computed for many more instances of the face within the video.

The proposed method contains 3 main steps:
(A) Video normalization: the video sequence is brought to standard parameters in terms of resolution and frame rate, so as to minimize possible alterations of the model caused by different video formats;
(B) Face model reconstruction: facial feature points, which represent the face shape, are extracted via Active Shape Model (ASM) and a face form in 3D is reconstructed by modelling a neutral shape to best approximate the extracted ASM;
(C) CG characters identification: the analysis of variations in following frames of the 3D face model, represented by exploiting the Principal Component Analysis (PCA), will lead to the final decision.

2.1. Video Normalization

In order to apply the proposed method to different kinds of video sources, we normalize the source to a standard format to avoid variations in the model due to the attributes of the video. The frame rate is therefore reported in the range 10-12 frames/second, which are largely sufficient to capture all the significant animation variations in a human face (noticed that the human visual system can process 10 to 12 separate images per second [8]).

In particular, a distance measure $D_f(F_i, F_j)$ between face models in frame F_i and frame F_j is defined and computed. $D_f(F_i, F_j)$ is computed exploiting three special feature points,

$$D_f(F_i, F_j) = \frac{1}{3} \sum_{k \in K} ||\rho_i^k - \rho_j^k|| \quad (1)$$

where ρ_i^k and ρ_j^k are the spatial coordinates of point k on the face in frame F_i and frame F_j, K = {left-eye inner corner, right-eye inner corner, philtrum}, and $|| \cdot ||$ is the Euclidean distance. These points are automatically extracted from each frame by using Luxand FaceSDK [9]. They are stable under different expressions and lighting conditions [10]. Thus, distances of these points can be considered as a measure of speed of the head movement, and hence can be used for this video normalization step. Particularly, for every second, if $D_f(F_i, F_j)$ is smaller or equal to a threshold T, M frames between F_i and F_j are grabbed. In our experiments, with T equal 8, 10, and 12, the number of frames grabbed M are 10, 11, and 12, respectively (there are no videos with $D_f(F_i, F_j) > 12$ in our experiments). As to the spatial resolution, each face is analyzed in a resolution of 400×400.

After this step (A), an input video is encoded as a series of face instances of the same person, adjusted by chronological order.

2.2. Face Model Estimation

In order to reconstruct the face model from a 2D input image, we apply the method from [11]. After building a reference 3D model, this method adapt this reference model to the 2D image through an optimization procedure.

To build the reference 3D model, Algorithm 1 is applied on a training set of 3D images, to construct a normalized mean shape $\overline{S}^{3D}$ which can be considered as a general 3D face model, and the corresponding eigenvectors matrix φ^{3D}, which can be used to transform a given 3D shape into the 3D face model. This normalized mean shape $\overline{S}^{3D}$ and the eigenvectors matrix φ^{3D} are called 3D Point Distribution Model (PDM). Notice that the PDM is built only once and can be applied to different faces in different videos.

Given a PDM, we now have to approximate it to all instances of faces output of step (A). In order to reconstruct the face model from a 2D input image, we have to project the 3D PDM into 2D space. This

Algorithm 1 Compute 3D Point Distribution Model (PDM)

Input: n different shapes of faces $\{s_1^{3D}, s_2^{3D}, \ldots, s_n^{3D}\}$, where $s_i^{3D} = \{x_i^1, y_i^1, z_i^1, x_i^2, y_i^2, z_i^2, \ldots, x_i^d, y_i^d, z_i^d\}$, where d is the number of ASM points and (x_i^k, y_i^k, z_i^k) is the spatial position of point k^{th} on face i^{th}.
Output: A normalized mean shape $\overline{S}^{3D}$ and the corresponding eigenvectors matrix φ^{3D} of the training faces .
Method: (inspired from [11])

1: Normalize all face shapes: all the points are scaled into [-1, 1]: $S_i^{3D} \leftarrow Normalize3D(s_i^{3D})$, $i = 1, 2, ..., n$.
2: Compute the mean shape: $\overline{S}^{3D} \leftarrow \frac{1}{n} \sum_{i=1}^{n} S_i^{3D}$
3: **repeat**
4: **for each** normalized shape S_i^{3D} **do**
5: Find rotation matrix R_i and translation vector t_i to transform S_i^{3D} into $\overline{S}^{3D}$.
6: $S_i^{3D} \leftarrow R_i(S_i^{3D}) + t_i$.
7: **end for**
8: Re-compute the mean shape: $\overline{S}^{3D} \leftarrow \frac{1}{n} \sum_{i=1}^{n} S_i^{3D}$
9: **until** convergence.
10: Apply Principal Component Analysis (PCA) to all normalized shapes S_i^{3D}, $i = 1, 2, ..., n$ to have the eigenvectors matrix φ^{3D}.

Algorithm 2 Extract pose and face parameters (from [11])

Input:

- A shape in 2D: $s^{2D} = \{x^1, y^1, x^2, y^2, \ldots, x^d, y^d\}$, where d is the number of ASM points and (x^k, y^k) is the spatial position of point k^{th} on the input face.
- A PDM ($\overline{S}^{3D}$ and φ^{3D}).

Output: The face model p^{3D}, rotation matrix R and translation vector t.
Method:

1: Normalize the face shape: all the points are scaled into [-1, 1]: $S^{2D} \leftarrow Normalize2D(s^{2D})$
2: $p^{3D} \leftarrow 0$.
3: **while** p^{3D}, R, and t do not converge **do**
4: Compute R and t by solving

$$Err = \left|\left| S^{2D} - P(R(\overline{S}^{3D} + \varphi^{3D} p^{3D}) + t) \right|\right|_2 \quad (2)$$

using Zhang's method [12], P is the projection transformation. A 3D point $(x_i, y_i, z_i)^T$ is projected by P into 2D space as follows:

$$P \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} = \frac{f}{z_i} \begin{pmatrix} x_i \\ y_i \end{pmatrix} + \begin{pmatrix} o_x \\ o_y \end{pmatrix} \quad (3)$$

where f is the focal length, and (o_x, o_y) is the principal point location on 2D image.
5: Compute the new face: $S^{*3D} \leftarrow R(\overline{S}^{3D} + \varphi^{3D} p^{3D}) + t$
6: Generate the ideal 3D shape S'^{3D}: $S'^{3D} \leftarrow x$ and y values from S^{2D} and z values from S^{*3D}.
7: Recompute p^{3D} from S'^{3D}:
$p^{3D} = (\varphi^{3D})^T (R^{-1}(S'^{3D} - t) - \overline{S}^{3D})$.
8: **end while**

could be done through an optimization procedure, which is summarized as Algorithm 2. The main idea is to perform the optimization process on a single instance each time: face pose is estimated based on the generated shape, then based on the new computed pose, the new face shape is re-estimated, and so on, i.e., either shape or pose is estimated each time based on the other information (in our experiments, the convergence is reached after 20 - 25 iterations). Thus, step (B) will produce all the information needed to map the set of 2D faces into the corresponding set of 3D face models.

Notice that differently from [11], the ASM points for each 2D face s^{2D} are extracted by using Luxand FaceSDK [9], which is able to extract 66 ASM points for each face in still images. Camera intrinsic parameters $(f, (o_x, o_y))$, the rotation matrix R and the translation vector t in equation (2) and (3), are represented as a single camera projection matrix, and hence can be jointly approximated. They can be decomposed from the camera projection matrix by using the method in chapter 6, section 6.3.2 in [13].

2.3. Computer Generated Character Identification

As an output of step (B) we have p^{3D}, R and t for each analyzed 2D face representation. Now in this last step (C) we propose to analyze the evolution of such 3D face models p^{3D} during the video, demonstrating that facial properties are changed by following some specific rules or movements and analyzing its variations allow to reveal synthetic animation. Thanks to PCA, exploiting the face model p^{3D} allows us to work on a space where information about the whole face is encoded but also somehow compressed in the first components which contain the most important parts of the signal. Furthermore, we study the evolution of the model not only during the whole video, but also on non-overlapping windows which can highlight particular animations and expressions of the represented character.

Let us assume each window W_j of length of l, and a face model p_i^{3D} extracted and encoded using PCA as $p_i^{3D} = (p_i^1, p_i^2, ..., p_i^n)$ for each frame i in W_j. In order to study how the 3D face model evolves in W_j and thus to measure the complexity of the geometric distortion during the animation, we extract the following properties:

1. The mean values μ_j^c, $c \leq n$, in W_j.

$$\mu_j^c = mean(||p_2^c - p_1^c||, ||p_3^c - p_1^c||, ..., ||p_l^c - p_1^c||) \quad (4)$$

2. The standard deviations of differences σ_j^c, $c \leq n$, in W_j.

$$\sigma_j^c = sdv(||p_2^c - p_1^c||, ||p_3^c - p_1^c||, ..., ||p_l^c - p_1^c||) \quad (5)$$

3. The average lengths of the trajectories τ_j^c, $c \leq n$, in the first components.

$$\tau_j^c = \frac{1}{l-1} \sum_{i=2}^{l} ||p_i^c - p_{i-1}^c|| \quad (6)$$

4. The average length of the combination of trajectories T_j^c, $c \leq n$, of the first components.

$$T_j^c = \frac{1}{l-1} \sum_{i=2}^{l} \sqrt{\sum_{k=1}^{c} ||p_i^k - p_{i-1}^k||^2} \quad (7)$$

(note that $T_j^1 = \tau_j^1$).

Here, μ_j^c contains the mean of the differences between the models and σ_j^c measures the spread of the models (from μ_j^c). τ_j^c and T_j^c evaluate the amount of changes of the models over time. Considering this problem as the analysis of a point moving in the space, Eq. (4) and Eq. (5) represent the mean position and variance, while Eq. (6) and Eq. (7) describe the length of the path of the moving point.

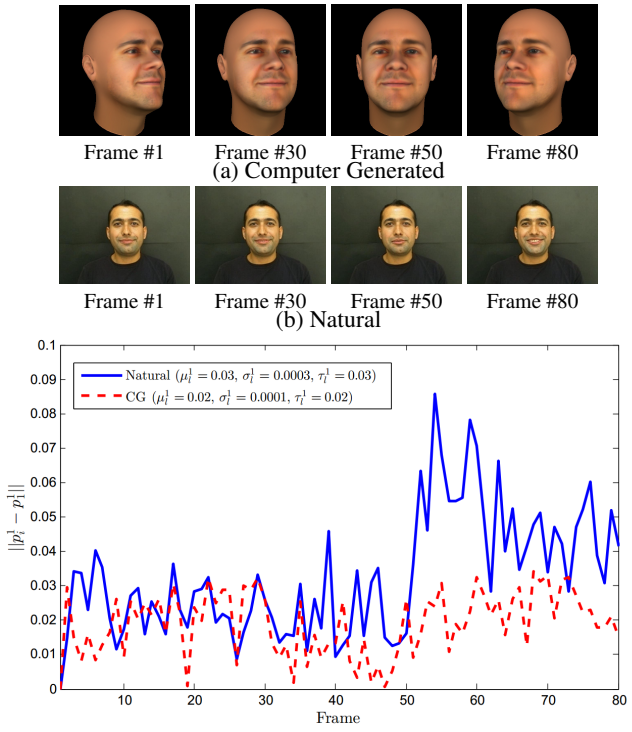


Fig. 1. Example of properties from Eq. (4) to Eq. (7) for a CG (a) and a natural (b) video sequences with $l = 80$ and $c = 1$.

These set of features are extracted from each window W_j. Another set of features is extracted in the same way over the whole video, i.e., $l = N$, where N is the number of frames. Since such feature set computed on the whole video is a fundamental information for videos with a main single expression while the average computed on sets corresponding to W_j, $\forall j$, is critical when we deal with more complicated videos with complex animations, both these properties are extracted and analyzed to lead to the final decision.

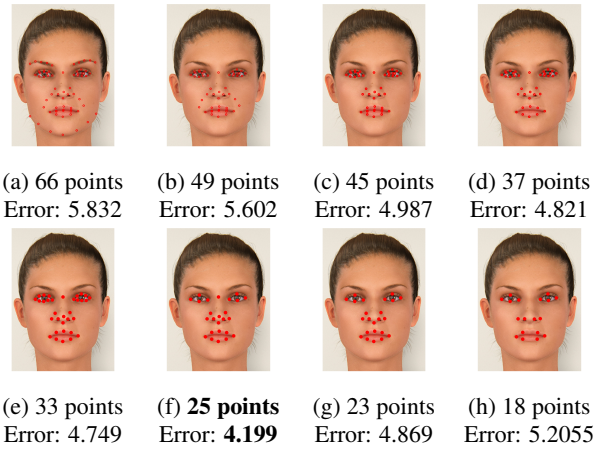
Shown in Figure 1 are the differences of the first component p_i^1 of each frame i from the p_1^1 of the first frame (see Eq. (4)) of (a) a CG happy face and (b) a natural happy face. In this example, the first 80 frames were extracted and analyzed. The mean μ_l^1, the variance σ_l^1 and the average length τ_l^1 (= T_l^1), $l = 80$, for (a) are smaller than the corresponding (b) values, as reported in Figure 1.

With all the properties extracted as described from Eq. (4) to Eq. (7), support Vector Machine (SVM) is used to differentiate between CG and natural face animations. We use a polynomial kernel with Sequential Minimal Optimization (SMO) method.

3. EXPERIMENTS

Two published datasets have been used in our experiments:

- **CASIA-3D FaceV1** [14]: which consists of 4624 scans of 123 people using the non-contact 3D digitizer. For each person, they collected 37 - 38 images, both in 3D and 2D, in different poses, expressions, and lighting conditions.
- Boğaziçi University Head Motion Analysis Project Database (**BUHMAP-DB**) [15], which contains 440 videos of 11 people (6 females, 5 males) performing 5 repetitions of 8 different gestures. Each video lasts about 1 - 2 seconds.



(a) 66 points Error: 5.832 (b) 49 points Error: 5.602 (c) 45 points Error: 4.987 (d) 37 points Error: 4.821

(e) 33 points Error: 4.749 (f) **25 points** Error: **4.199** (g) 23 points Error: 4.869 (h) 18 points Error: 5.2055

Fig. 2. Different setups on facial landmark positions and the correspondent errors.

We also created and collected faces and videos via FaceGen software [16] (created from the images in BUHMAP-DB and CASIA-3D FaceV1) and internet, which are:

- **Set 1**: 10 face models in 3D, rotated with different poses and stored in 2D to have 60 images in total.
- **Set 2**: 24 sets of synthetic characters performing different expressions, (2 males and 2 females with 6 expressions for each person). For each set, 100 images has been created as a video of 10 seconds with the frame rate of 10 fps.

3.1. 3D Face Reconstruction

Since the proposed method depend deeply on the extracted points from Luxand FaceSDK [9], it is necessary to perform experiments to measure the performance of this application. Hence, we used images of 30 people from CASIA-3D FaceV1, 20 images for each person with different poses and lighting condition for testing. The result was obtained with the average difference of 5.83 pixels, when comparing the reconstructed 66 ASM points with the manually marked points (the resolution of the image is 400×400). However, the position of points at eyebrows or chin is often not very accurate due to occlusion or illumination conditions. Therefore, reducing the number of selected points usually allows to improve the performance of the reconstruction step (B). Hence, we performed the test not only on the setup with 66 ASM points but also with other configurations, ranging from 66 points to 18 points.

An illustration of all setups is shown in Figure 2 which also shows that setup (f) with 25 points provides the best solution with an average error of 4.2 pixels (approximate 1%, comparing to the size of the image). This setup also provides the best solution on synthetic faces from Set 1, with an average error of 4.1 pixels. Hence, we used this configuration for all following experiments.

3.2. CG Facial Expression Identification

In this experiment, we ran our method on the BUHMAP-DB dataset and compare the results with [7]. Notice that in this case animations will mainly consist of single expressions like happiness or sadness. Windows size $l = 4$ and the number of analyzed components $c = 3$ are chosen, the LOO (Leave One Out) cross validation was used. The accuracy achieved by the proposed method outperforms results in [7] as shown in Table 1.

Table 1. Comparision between the proposed approach and the method in [7].

Animations	Happiness	Sadness
Method in [7]	67.5%	72.5%
Proposed approach	**97.5%**	**87.5%**

3.3. Synthetic Animation Identification

In this experiment, we used 24 animations from BUHMAP-DB and 24 synthetic animations from Set 2. Notice that in this case, animations are more complicated, consisting of both expressions and other gestures of the faces.

We ran our proposed approach with different sets of features, i.e., different values of l and c, in order to determine the best configuration. SVM was used as a binary classification and LOO cross validation was applied in the test. The proposed method obtained the best result with the accuracy of 93.75% on the windows length $l = 4$ with the features extracted from the first 3 components, i.e., $c = 3$. The details are shown in Table 2 where columns are the numbers of components c and rows are the length l of the analyzed windows.

Table 2. Accuracy performance of the proposed method on different configurations. Columns are the numbers of components and rows are the length of the analyzed windows. Accuracies are displayed in percentage.

l \ c	1	2	3	4	5	6
2	62.50	72.92	68.75	64.58	47.92	45.83
3	70.83	85.42	89.58	77.08	60.42	50.00
4	70.83	89.58	**93.75**	81.25	58.33	50.00
5	68.75	85.42	85.42	70.83	56.25	47.92

4. CONCLUSIONS

In this paper we introduced a robust technique to differentiate between CG and natural faces in video. By analysing the evolution of the face model in time, the method can overcome difficulties caused by different face poses, occlusions, or lighting inconsistencies. Experimental results showed that synthetical animations can be identified since their model are normally recreated or performed in patterns, comparing to natural animations which follow more complicated geometric distortion. Larger datasets and more complex actions will be considered in further work.

Acknowledgement

This work was partially supported by the ICT COST Action IC1105 - 3D-ConTourNet - 3D Content Creation, Coding and Transmission over Future Media Networks.

Portions of the research in this paper use the CASIA-3D FaceV1 collected by the Chinese Academy of Sciences' Institute of Automation (CASIA).

5. REFERENCES

[1] S. Lyu and H. Farid, "How realistic is photorealistic?," *IEEE Transactions on Signal Processing*, vol. 53, no. 2, pp. 845–850, 2005.

[2] Y. Wang and P. Moulin, "On discrimination between photorealistic and photographic images," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2006, pp. II.161–II.164.

[3] T. T. Ng, S. F. Chang, J. Hsu, L. Xie, and M. P. Tsui, "Physics-motivated features for distinguishing photographic images and computer graphics," in *ACM Multimedia*, 2005, pp. 239–248.

[4] N. Khanna, G. T. C. Chiu, J. P. Allebach, and E. J. Delp, "Forensic techniques for classifying scanner, computer generated and digital camera images," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2008, pp. 1653–1656.

[5] V. Conotter and L. Cordin, "Detecting photographic and computer generated composites," in *SPIE Symposium on Electronic Imaging*, 2011.

[6] D.-T. Dang-Nguyen, G. Boato, and F. G. B. DeNatale, "Discrimination between computer generated and natural human faces based on asymmetry information," in *European Signal Processing Conference*, 2012, pp. 1234–1238.

[7] D.-T. Dang-Nguyen, G. Boato, and F. G.B. DeNatale, "Identify computer generated characters by analysing facial expressions variation," in *IEEE International Workshop on Information Forensics and Security*, 2012, pp. 252–257.

[8] P. Read and M.-P. Meyer, *Restoration of motion picture film*, Butterworth-Heinemann, 2000.

[9] "Luxand FaceSDK," http://luxand.com/facesdk/, June, 2013.

[10] Y. Liu, K. L. Schmidt, J. F. Cohn, and S. Mitra, "Facial asymmetry quantification for expression invariant human identification," *Computer Vision and Image Understanding Journal*, vol. 91, no. 1/2, pp. 138–159, 2003.

[11] S.-C. Chen, C.-H. Wu, S.-Y. Lin, and Y.-P. Hung, "2d face alignment and pose estimation based on 3d facial models," in *IEEE International Conference on Multimedia and Expo*, 2012, pp. 128–133.

[12] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, Nov. 2000.

[13] Emanuele Trucco and Alessandro Verri, *Introductory Techniques for 3-D Computer Vision*, Prentice Hall PTR, 1998.

[14] "CASIA-3D FaceV1," http://biometrics.idealtest.org/, June, 2013.

[15] O. Aran, İ. Ari, M. A. Güvensan, H. Haberdar, Z. Kurt, H. . Türkmen, A. Uyar, and L. Akarun, "A database of non-manual signs in Turkish sign language," in *Signal Processing and Communications Applications*, 2007, pp. 1–4.

[16] "FaceGen Modeller from Singular Inversions," http://www.facegen.com, 2004.