Eurographics Italian Chapter Conference (2010)
E. Puppo, A. Brogni, and L. De Floriani (Editors)

Facile: a System for Artificial Face Dataset Generation

A. Brambilla[†1] and A. Colombo[‡1]

[1]DISCo, Università degli studi di Milano Bicocca, Italy

Abstract

We present Facile, a system for the generation of artificial face datasets aimed to assist the training and evaluation phases of face related algorithms. Facile is able to generate, starting from a set of textured 3D faces with neutral expressions, a new dataset containing synthesized facial expressions, multiple viewpoint positions and lighting conditions. The system is featured by a facial animation algorithm based on a mass-spring system. Muscles, defined on a reference face, modify springs contractions when activated. Dense correspondence with the reference face is computed for each input face: in this way muscles are automatically positioned and the desired facial expression is generated through physical simulation. Output images can be rendered with different lighting conditions and viewpoint positions depending on the task requirements.

Categories and Subject Descriptors (according to ACM CCS): Face Modeling; Mass spring systems; Facial expressions.

1. Introduction

One of the most characteristic feature of the face is that, among biometrics, is one of the most influenced by various sources of variability. The appearance and shape of a 2D or 3D image of the face can be altered by lighting and make-up (in case of 2D images), pose, occlusions, aging, expressions, beard, hairstyle and many other factors. Great efforts are spent by the research community in order to design algorithms able to deal with such variations. However, though several competitions have been organized for evaluation and benchmarking (FERET, FRVT200, FRVT2002, FRVT2006, FRGC), the in-depth analysis of face recognition systems requires much larger databases and more sophisticated tools. If we consider a set of 1.000 subjects, a database covering all the cited source of variability (including combinations) rapidly increase in size becoming impractical in terms of time, money and human-work. Consider, for example, ten facial expressions with three different intensities, five different lighting conditions, nine different orientations and four different occluding objects; even without taking into account aging, this database should be composed of more than five millions acquisitions. If we consider that in this example each source of variability has been poorly sampled on a very small population, it is clear that the problem of properly train and test face recognition systems cannot be underestimated.

The use of synthetic samples is not a new topic in biometrics. In [CMM02] a system for the generation of synthetic fingerprints (SFinGe) has been presented. The system has been used for generating a database used in international competitions (FVC2000, FVC2002, FVC2004 and FVC2006) obtaining results close to those obtained on real databases, proving that the real inter-class and intra-class variations of fingerprints were very well captured.

In [OPC03] a system for generating synthetic textured 3D face models with neutral expressions has been adopted to test a commercial face recognition system. Results has been compared against those obtained in the FRVT 2000 competition proving the effectiveness of artificial imagery.

In Colombo et al. previous works [CCS09, CCS08] artificial occlusion generation has been adopted to test a 3D face detection and recognition pipeline able to deal with partially occluded faces. The use of synthetic occlusions allowed an automatic in-depth analysis of the pipeline preserving a great

[†] andrea.bram@gmail.com
[‡] colomboal@disco.unimib.it

DOI: 10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2010/073-079

amount of human work which would have been necessary to label occluded and non-occluded pixels.

We believe that an automatic face generation system would be undoubtedly a precious tool for the vision research community. Real world datasets require a lot of time to be captured and considerable time and efforts to annotate each image. In fact, a face database should contain the ground truth for feature points and other characteristics such as subjects information (sex, race, age), occlusions, facial expressions and so on. Up to date, there are no public database in which image pixels are annotated as occluded or non-occluded. A complete synthetic face generation tool would resolve all these problems, generating a huge amount of annotated images with little effort.

One of the most difficult aspect in synthetic face generation is facial expressions synthesis and transfer. Recently, Blanz et al. [BBPV03] proposed the use of morphable models to accomplish these tasks. Facial expressions are encoded as directions on the morphable model face space. Their system is also able to reanimate faces in images and videos using an analysis–synthesis approach. Sibbing et al. [SHK10] combined surface editing and computer vision techniques in order to transfer the facial expression of a human actor to a predefined template mesh; they use a morphable model to make the reconstruction and tracking phases more robust, and to have better control over facial smoothness and features fold-back.

In this paper we presents our first steps toward the implementation of a system (Facile) for the generation of synthetic faces, aimed to training and test face-related algorithms. In particular, we present our efforts to generate artificial facial expressions. Starting from a set of 3D textured models of different subjects captured with neutral expression, Facile is able to generate a new dataset enriched with facial expressions, different viewpoints and lighting conditions. Differently from other state of the art approaches [BBPV03, SHK10], we have chosen to adopt a physical model of skin and muscles since we want to avoid a statistical parameterization, which does not maintain a precise link to muscles activations, jaw rotation and, in general, to spatially localized physical activators. Physics-based parameterizations are undoubtedly more useful and easier to use in case of algorithm analysis and testing, especially if associated with high-level coding systems, like the Facial Action Coding System (FACS) [EFH78].

2. System Overview

In Figure 1 is depicted the main diagram of the system. The input is a triangular textured 3D mesh representing a human face. The mesh should be normalized, i.e. registered in a predefined position. This can be done manually or using an automatic 3D face detection and normalization algorithm, such as [CCS09]. The first component of the system is the registration module, which has the purpose of establishing a dense correspondence with the mean reference face (computed averaging 100 facial 3D images). The reference face is parameterized in 2 dimensions through cylindrical projection. The output of the correspondence module is a warped cylindrical projection of the input mesh where each position (h,ϕ) is in correspondence with the reference face. The module is built upon a simplified version of the morphable model registration approach presented in [BPV06].

Once the correspondence is computed, facial muscles can be directly transferred from the mean face to the input face. The following animation module is able to animate the neutral face based on the specified muscle contraction parameters and a jaw rotation angle. The module uses a mass-spring mesh and a physical simulator. The final images are rendered specifying lighting conditions and viewpoint parameters.

3. Registration

The registration module computes a dense correspondence between the input face mesh and the mean reference face. We decided to adopt the same approach used in [BPV06]: the reference and the input face are mapped on a 2D domain using cylindrical projection:

$$I(h,\phi) = [r(h,\phi), R(h,\phi), G(h,\phi), B(h,\phi)] \tag{1}$$

where r is the radius and R,G,B are the color components. In this way, correspondence is treated as a 2D image problem. As in [BPV06] we used a modified optical flow algorithm for computing a dense vector field $v(h,\phi) = [\Delta h(h,\phi), \Delta\phi(h,\phi)]$ which indicate the spatial offset of a point on the first image respect to its corresponding point on the second image. Briefly, the algorithm is based on the assumption that entities in a generic image sequence $I(x,y,t)$ move across the image at velocity $(v_x,v_y)^T$ conserving their brightness:

$$\frac{dI}{dt} = v_x\frac{\delta I}{\delta x} + v_y\frac{\delta I}{\delta y} + \frac{\delta I}{\delta t} = 0 \tag{2}$$

If the image sequence is composed of two images showing different objects (such as two different faces), correspondences can be still successfully computed. If the velocities v are assumed to be constant on a neighborhood $R(h_0,\phi_0)$, they can be obtained minimizing at each point (h_0,ϕ_0):

$$E(h_0,\phi_0) = \sum_{h,\phi\in R} \|v_h\frac{\delta I(h,\phi)}{\delta h} + v_\phi\frac{\delta I(h,\phi)}{\delta\phi} + \Delta I(h,\phi)\|^2. \tag{3}$$

The solution is computed at each point solving a 2x2 linear system. The algorithm adopt a coarse to fine strategy through a Gaussian pyramid. Additional details can be found in [BPV06] and [BH90].

The image function $I(h,\phi)$ may be augmented with additional quantities in order to improve results. In our case, we obtained the best result using :

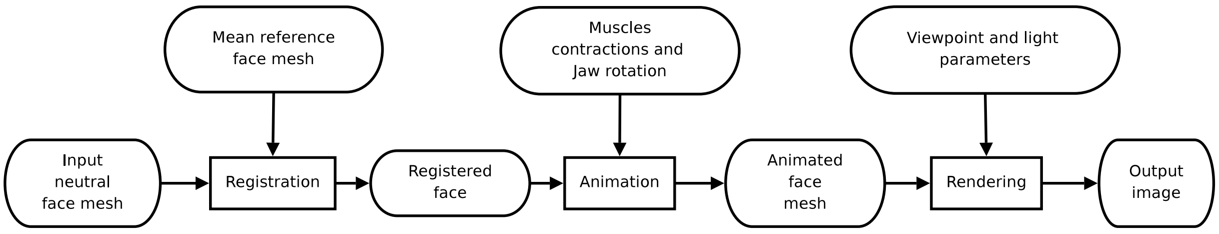

Figure 1: *Facile pipeline*

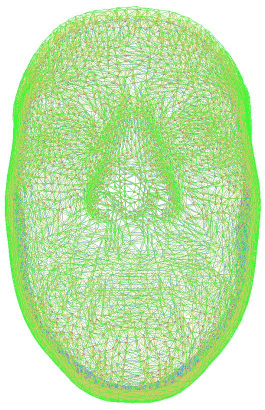

Figure 2: *Three-layered physical mesh used by the mass-spring simulator applied on the reference face.*

$$\|I(h,\phi)\|^2 = w_{N_x}N_x^2 + w_{N_y}N_y^2 + w_{N_z}N_z^2 + \\ + w_R R^2 + w_G G^2 + w_B B^2 + \\ + w_H H^2 + w_K K^2; \qquad (4)$$

where N_x, N_y, N_z are the components of the surface normal, RGB are the color components and H,K are the mean and Gaussian curvature. The weights w_i are chosen in order to adjust the different variations and to weight each component contribution.

4. Animation

Once a face has been registered, the physical simulator maps a predefined three-layered mass-spring mesh and the muscles on the registered face. Similar to [LTW95], our elastic mesh represents the fixed skull layer, the middle fascia layer and the external skin layer; the initial spring stiffness is computed using the method proposed in [VG98]. The jaw is considered part of the skull but is able to rotate by the specified angle. In order to limit computational timings, we use a mesh with less than 4000 vertices per layer (see Figure 2).

Parameters	Values
skin thickness	5
skin stiffness	20
skin-fascia stiffness	40
fascia stiffness	60
fascia-skull stiffness	5
m_i	0.5
γ_i	25
k_s	30
k_v	2

Table 1: *The values of parameters used for the construction of the mesh and for the physical simulation.*

4.1. Elastic mesh structure

First of all, given a facial 3D scan, a mass-spring mesh needs to be computed. As mentioned before, we model the epidermal tissue using a triangle-based mass-spring mesh composed by three layers:

- the most external one represents the skin;
- the mid-layer represents the *fascia*, which is the fibrous tissue that covers the underlying muscle layer;
- the last one is immovable and represents the skull.

The skin and the fascia are interconnected by both vertical and diagonal (cross-shaped) springs of a certain length (depending on the thickness of the skin), while fascia and skull points are actually overlapped and connected by springs of length 0. Springs of different layers have different stiffness, as showed in Table 4; moreover, the stiffness of the springs lying on the skin layer and on the fascia layer are corrected using the algorithm described in [VG98].

Defining and building the elastic mesh is rather simple: the edges and vertices of the skin layer are defined over the cylindrical projection of the mean face, the actual three-dimensional data is determined only when a model is loaded: taking advantage of the precomputed dense correspondence, the mesh is automatically and dynamically adapted to the selected face. Using the same principle, some additional data is specified over the mean face, i.e. the distinction between skull and jaw, the axis of rotation of the jaw and whether the fascia is attached or not to the skull. Fascia and skull layers are obtained shifting inward the skin layer: currently we use constant skin thickness over the whole face but we plan to create a more accurate model.

4.2. Muscle System

Muscles are manually defined on the reference mean face and are subdivided in two categories: piecewise linear and sphincter. The contraction parameters define how much a muscle contracts toward its attachment point (*piecewise linear muscles*) or toward its center (*sphincter muscles*). Muscle contractions are translated in net spring stiffness and relaxed length alteration. In this phase, muscles drive the animation and are, at the same time, influenced by skin: here we differ from state of the art algorithms (for example [KHpS01]) in which muscles are not affected by skin. In this way we are able to automatically handle muscle interactions and the case of the *orbicularis oris* which is pulled by the skin when the jaw drops. The physical simulator computes the equilibrium state solving a system of differential equations describing the motion of masses affected by the net springs.

More in detail, muscles are defined with a set of control points $p_i, i \in 1,..n$. Piecewise linear muscles contract toward their attachment point p_0, while sphincter muscles contract themselves toward their center. Note that, differently from [KHpS01], the contraction center is part of the muscle and it is not an external point. In Facile, we model the orbicularis oris with multiple sphincter muscles since we do not allow sphincter muscles to form loops. For each muscle, the user specify a contraction factor $c \in [0,1]$, where 0 means no contraction and 1 mean full contractions. The contracted control points positions, q_i, are computed in the following manner:

- for each p_i, we compute a parameter $b_i \in [0,1]$ which represents the normalized distance along the muscle from the contraction origin. For piecewise linear muscles we have:

$$b_i = \begin{cases} 0, & \text{if } i = 0; \\ \frac{\Sigma_{j=1}^{i} \|p_j - p_{i-1}\|}{\Sigma_{j=1}^{n-1} \|p_j - p_{i-1}\|}, & \text{otherwise.} \end{cases} \quad (5)$$

  while for sphincter muscles $b_i \in [-0.5, 0.5]$ since it refers to the center of the muscle:

$$b_i = \begin{cases} -0.5, & \text{if } i = 0; \\ \frac{\Sigma_{j=1}^{i} \|p_j - p_{i-1}\|}{\Sigma_{j=1}^{n-1} \|p_j - p_{i-1}\|} - 0.5, & \text{otherwise.} \end{cases} \quad (6)$$

- The contraction is applied scaling each b_i by the contraction factor c, obtaining d_i:

$$d_i = (1-c)b_i \quad (7)$$

  for the piecewise linear case, and

$$d_i = 0.5 + (1-c)b_i \quad (8)$$

  for sphincter muscles.

- For each parameter d_i the index j_i of the segment containing d_i is found:

$$j_i = v : b_v < d_i < b_{v+1} \quad (9)$$

- The final contracted positions q_i are computed through linear interpolation:

$$q_i = p_{j_i} + (p_{j+1} - p_{j_i}) \frac{d_i - b_i}{b_{i+1} - b_i} \quad (10)$$

The Facile muscle system, instead of modifying vertices positions, modify the fascia springs stiffness and relaxed length. In a preprocessing phase, in fact, the system determines which fascia springs are interested by the muscle, checking if both spring ends are crossed by the muscle, i.e. the distance from muscle segments is less than the muscle width. Interested vertices are contracted along the muscle; briefly the steps are the following:

- the position of vertex v is projected on the nearest segment, determining the normalized distance $\tilde{b_v}$ from the contraction origin (in the same way b_i's are computed) and its coordinates relative to the segment reference system;
- $\tilde{b_v}$ is scaled proportionally to the contraction c towards the origin and the containing segment is determined;
- the final position is computed transforming the vertex coordinates in the contracted segment reference system.

The new spring length is equal to the distance between the new positions of its ends. The stiffness of contracted springs is simple multiplied by a factor S: we found that a value of $S = 20$ allowed muscle springs to effectively pull the layers producing good results. In case of piecewise linear muscles, springs connecting the skull layer and the first segment of the muscle are also increased in stiffness by a factor S since, in this case, muscles are attached to the skull and must pull toward the attachment point. The system also checks if a spring is affected by multiple muscles: in this case the final rest length is computed weighting each muscle contribution by its contraction parameter.

4.3. Physical Simulator

Once the parameters of the springs have been modified according to the specified muscle contractions, a physical simulator determines the resulting deformation of the elastic mesh. The core of the simulator is the well-known Velocity Verlet algorithm [SABW82], conveniently revised to work with variable-size steps and to handle collisions.

4.3.1. Equation of motion

The physical simulation is based on the Lagrange equation of motion: for each node i of the mesh, we have

$$m_i \frac{d^2 \underline{x}_i}{dt^2} + \gamma_i \frac{d\underline{x}_i}{dt} + \underline{g}_i(t) + \underline{s}_i(t) + \underline{v}_i(t) = \underline{f}_i \quad (11)$$

where m_i and $\underline{x}_i$ are the node mass and position respectively, $\underline{g}_i$ is the net spring force exerted by the elastic mesh on node i, while γ_i is a damping coefficient. Following the idea proposed in [LTW95], for each skin element e we take into account two additional terms:

$$\underline{s}_i = k_s * (\underline{p}_i^e - \underline{\tilde{p}}_i^e) \tag{12}$$

$$\underline{v}_i = k_v * \underline{n}^e (V^e - \tilde{V}^e). \tag{13}$$

$\underline{p}_i^e$ and $\underline{\tilde{p}}_i^e$ are the current and rest nodal coordinates of node i with respect to the center of element e: the resulting force $\underline{s}_i$ is called *shape preservation force* and tries to maintain the shape and orientation of the skin element unaltered. Similarly, V^e and $\tilde{V}^e$ are the current and rest volume of element e, $\underline{v}_i(t)$ is the *volume preservation force* and its purpose is conserving the element volume after skin deformations caused by the actions of muscles. Determining the exact volume of skin elements is actually too expensive, so only an approximated value is computed; the direction of the force is given by $\underline{n}^e$, the normal of the lower triangle of e (the one on the fascia). The strength of these two forces depends on the scaling k_s and k_v.

Note that, since muscle actions are converted to alterations of springs parameters, the external forces $\underline{f}_i$ are always zero.

4.3.2. Collision detection

The great majority of the existing mass-spring systems try to avoid skull penetration (i.e. nodes of the skin that fall below the skull surface) adding a proper term to Equation 11: this approach, other than being usually inaccurate, has the annoying drawback of hindering nodes movements towards the skull, even movements that will not cause skull penetration. For this reason we decided to handle skull penetration explicitly: when a node falls below the skull, it is moved back to the skull surface. Moreover, for nodes close to the skull (within a certain threshold), the inward component of the resulting force is nullified. To speed up the collision detection process, a cylindrical projection of the skull is initially computed: during the simulation every point of the fascia is projected in the same way, then its position is checked against the corresponding skull point. Two separated skull pieces are used, one for the jaw and one for the rest of the skull.

This approach has also the advantage of allowing the correct simulation of the jaw movements: when the jaw is opened or translated, the system directly computes the final position of the corresponding skull points, then, during the simulation, the collision detection algorithm moves fascia and skin points accordingly. Actually, when the jaw is moved, in order to improve the convergence of the simulation algorithm, fascia and skin points are moved too; then the physical simulator computes the final equilibrium positions.

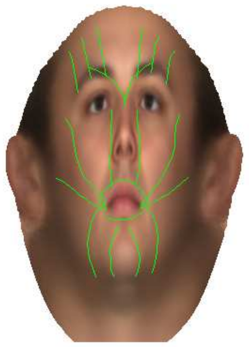

Figure 3: *Muscles actually modeled by facile displayed on the cylindrical projection of the reference face.*

5. Results

We tested Facile with almost 50 3D scans of human faces. Figure 4 shows some example of the physical simulator in action. As it can be seen, the same facial expression applied to the mean face is successfully transferred to the other faces through our dense correspondences module. We developed a small tool which allows us to define muscles on the cylindrical projection of the reference face. Actually we modeled the following muscles (see Figure 3): orbicularis oris, zigomatics, risourius, frontalis, caninus, corrugators and mentalis. Facial expressions can be generated specifying, for each muscle, its contraction factor in the range $[0,1]$. Although not all face aspects are actually modeled by Facile (such as eyes movements, muscle bulging, physically correct skin reaction), we are able to generate a huge number of facial expressions with minimal effort.

In Figure 5 are depicted a set of examples of rendered output images generated with a standard Whitted raytracer. The user specifies the number and the characteristics of light sources and selects the viewpoint. We plan to adopt a more sophisticated rendering engine and to model materials more accurately.

6. Conclusions and future work

We presented Facile, a system for the generation of artificial face datasets starting from a set of neutral 3D textured acquisitions. The generated dataset contains synthesized facial expressions, different viewpoints and lighting conditions, allowing face-related algorithms to explore face space variations. Facile is featured by a facial expression generator based on a physical simulator. Muscles are defined on the reference mean face and are automatically positioned on novel faces through dense correspondence. The proposed muscle system presents several advantages. First of all, muscles are embedded in the spring-mass system and muscle collisions do not need to be handled explicitly during the

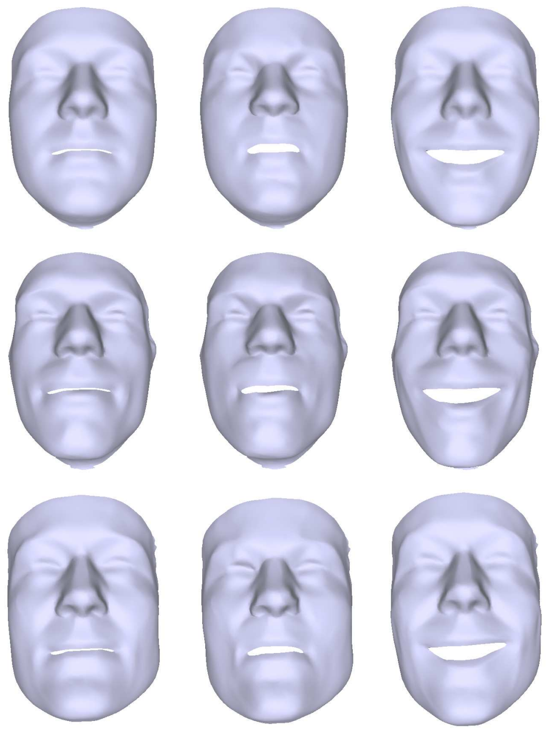

Figure 4: *Examples of animated faces. Left column: neutral expression. Middle column: angry expression. Right column: smiling expression. On the top row animations are applied to the reference face, while the last two rows shows the same expressions applied to two other faces taken from our database.*

simulation. Second, skin is affected by muscles and vice-versa: when the jaw is rotated the skin pulls the orbicularis oris automatically.

We plan to extend the physical simulator including eyes movements and enriching the model with teeth, tongue and a complete model of the head. Further improvements involve the adoption of a more accurate physical simulator based, for example, on the finite elements approach. This will allow an accurate modeling of shape variations and therefore a possible adoption of Facile for training and testing 3D face-related algorithms. Facile is ready to be integrated with generative models of faces, such as the morphable model approach proposed in [BV99].

We conjecture that ultra-realistic modeling of faces is not necessary to train and test 2D face-related algorithms. This is supported by the fact that small images are often used for training purposes: for example [VJ01] adopts 24x24 pixels images. We will soon start experimenting Facile for training and testing a 2D face detection algorithm and a feature points localizer in order to verify our conjecture.

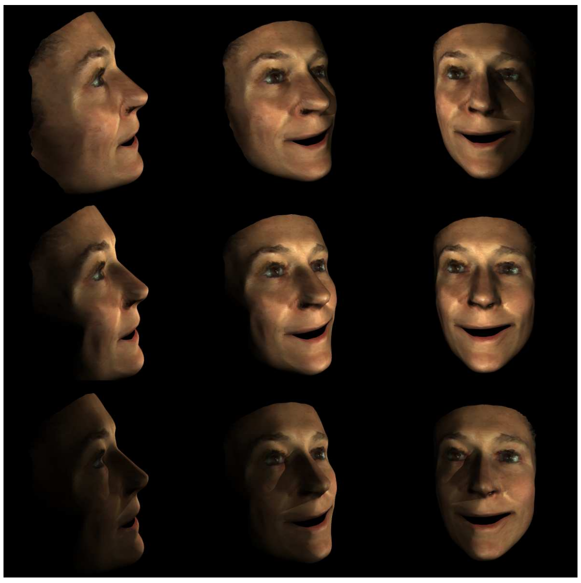

Figure 5: *Examples of facile output images. The same subject with a smiling facial expressions is depicted. Viewpoint and illumination conditions are varied allowing face related algorithms to explore image variations.*

References

[BBPV03] Blanz V., Basso C., Poggio T., Vetter T.: Reanimating faces in images and video. In *Computer Graphics Forum* (2003), vol. 22, pp. 641–650.

[BH90] Bergen J. R., Hingorani R.: *Hierarchical motion-based frame rate conversion*. Tech. rep., David Sarnoff Research Center Princeton NJ 08540, 1990.

[BPV06] Basso C., Paysan P., Vetter T.: Registration of expressions data using a 3d morphable model. *Automatic Face and Gesture Recognition, IEEE International Conference on 0* (2006), 205–210.

[BV99] Blanz V., Vetter T.: A morphable model for the synthesis of 3D faces. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (1999), ACM Press/Addison-Wesley Publishing Co., pp. 187–194.

[CCS08] Colombo A., Cusano C., Schettini R.: Recognizing faces in 3d images even in presence of occlusions. In *Biometrics: Theory, Applications and Systems, 2008. BTAS 2008. 2nd IEEE International Conference on* (29 2008-Oct. 1 2008), pp. 1–6.

[CCS09] Colombo A., Cusano C., Schettini R.: Gappy pca classification for occlusion tolerant 3d face detection. *Journal of Mathematical Imaging and Vision 35*, 3 (2009), 193–207.

[CMM02] Cappelli R., Maio D., Maltoni D.: Synthetic fingerprint-database generation. In *ICPR (3)* (2002), pp. 744–747.

[EFH78] Ekman P., Friesen W., Hager J.: Facial Action Coding System.

[KHpS01] Kähler K., Haber J., peter Seidel H.: Geometry-based muscle modeling for facial animation. In *In Proc. Graphics Interface 2001* (2001), pp. 37–46.

[LTW95] LEE Y., TERZOPOULOS D., WATERS K.: Realistic modeling for facial animation. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1995), ACM, pp. 55–62.

[OPC03] ORLANS N. M., PISZCZ A. T., CHAVEZ R. J.: Parametrically controlled synthetic imagery experiment for face recognition testing. In *WBMA '03: Proceedings of the 2003 ACM SIGMM workshop on Biometrics methods and applications* (New York, NY, USA, 2003), ACM, pp. 58–64.

[SABW82] SWOPE W., ANDERSEN H., BERENS P., WILSON K.: A computer simulation method for the calculation of equilibrium constants for the formation of physical clusters of molecules: Application to small water clusters. *The Journal of Chemical Physics 76* (1982), 637.

[SHK10] SIBBING D., HABBECKE M., KOBBELT L.: Markerless reconstruction of dynamic facial expressions. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on* (2010), IEEE, pp. 1778–1785.

[VG98] VAN GELDER A.: Approximate simulation of elastic membranes by triangulated spring meshes. *Journal of graphics tools 3*, 2 (1998), 42.

[VJ01] VIOLA P. A., JONES M. J.: Rapid object detection using a boosted cascade of simple features. In *CVPR (1)* (2001), pp. 511–518.