Volume 24 (2005), number 4 pp. 729–741

Adaptive Deformable Models for Graphics and Vision[†]

Siome Goldenstein[1], Christian Vogler[2], and Luiz Velho[3]

[1] Instituto de Computação - Universidade Estadual de Campinas - Caixa Postal 6176, 13083-971, Campinas, SP, Brasil
siome@ic.unicamp.br

[2] Gallaudet Research Institute - Gallaudet University - 800 Florida Ave. NE, Washington DC, USA 20002-3695
christian.vogler@gallaudet.edu

[3] IMPA–Instituto de Matemática Pura e Aplicada - Estrada Dona Castorina, 110, 22460 Rio de Janeiro, RJ, Brasil
lvelho@impa.br

ABSTRACT

Deformable models are a powerful tool in both computer graphics and computer vision. The description and implementation of the deformations have to be simultaneously flexible and powerful, otherwise the technique may not satisfy the requirements of all the distinct applications. In this paper, we introduce a new method for the deformable model specification: deformable fields. Deformable fields are conceptually simple, lead to an easy implementation, and are suitable for adaptive models. We apply our new technique to describe an adaptive deformable face, and compare three different adaptation strategies. We show how our technique is suitable to describe different individuals, how to construct a model based on information from a single image, and how it allows the tracking of the deformation parameters over a video sequence.

Keywords: adaptive deformations, deformation modeling, deformable object tracking

ACM CCS: I.3.5 Computer Graphics: *Curve, Surface, Solid and Object Representations*

1. Introduction

Deformable models are at the heart of many applications in both computer graphics and computer vision. These models can represent rigid parts, such as in CAD applications; articulated rigid bodies, such as in robots; or soft deformable surfaces, such as the human face. It is not an easy task to represent and manipulate all these different objects in a unified way.

In computer graphics, the models usually need more than just rigid transformations to convey realism, whereas in computer vision, model-based techniques use this same model to reduce the search space of inverse problems, such as tracking or fitting. A good model for the deformations will allow better results and more robust implementations.

In computer vision, deformable models reduce the dimensionality of an inverse problem, allowing us to look for solutions only spanned by the deformation parameters. Classic applications are shape fitting and tracking the dynamics of the model's parameters.

Concrete 3D models are usually represented by a finite number of vertices, whose connectivity is organized in a mesh, that is, a discretization of the surface. This discretization is critical to the application: too few vertices and the model approximation will be poor, with disastrous effects. As we show later in the paper, in Section 8, the level of detail has a large impact on the accuracy of tracking. In contrast, too many vertices make it computationally too expensive to perform all calculations since many algorithms have higher than linear complexity. In any case, the designer needs a good understanding of the problem beforehand, so as to choose the right model resolution, or we need a computer algorithm to automate the task.

In this paper, we address the question of adapting and deforming models simultaneously, and apply the results to both

[†]Based on "Adaptive Deformable Models", by S. Goldenstein, C. Vogler and L. Velho, which appeared in Proceedings of SIBGRAPI/SIACG 2004 (ISBN 0-77=695-2227-0). © 2004 IEEE.

Submitted March 2005
Revised April 2005
Accepted May 2005

computer graphics and computer vision tracking. First, we introduce the concept of *deformable fields* (Section 3). Deformable fields are a new method to describe deformations that are resolution independent. They are conceptually simple, lead to an easy implementation, and are suitable for adaptive models.

Then, we describe how a powerful adaptive mesh library can be plugged into the deformation description system (Section 4), and compare different mesh-refining criteria to achieve levels of detail best suited to the application at hand (Section 5).

In Section 6, we show how to describe a principal component decomposition of a face database as deformation maps. With that in hand, it is only natural to use PCA (Principal Component Analysis) techniques [3] to find three-dimensional models from model–image correspondences.

We present local deformations and animation of human faces as a more elaborate example of deformation fields (Section 7). We derive a simple set of deformations and show that they are indeed face independent, appropriate for immediate use after the fitting stage.

Finally, in Section 8, we use the deformable model to recover the rigid and non-rigid motion of a face from uncalibrated monocular images and without the aid of visual markers. For that to happen, we first have to fit it to represent the particular subject in a video sequence, and we also need to design the deformations that describe all the motion we want to capture. We perform a quantitative validation of the procedure's accuracy with different types of model adaptation, and investigate how the level of detail affects accuracy and runtime.

2. Related Work

In order for a computer graphics or computer vision application employ deformable models, it needs some type of geometry and deformation representation [1,3,14,19,30,34].

There exist a significant number of problem domains in which dynamic meshes are required. Consequently, the literature in this area is vast. Here, we will mention only a few illustrative examples. Bowden and colleagues [2] employ an inflating balloon model to reconstruct a surface from volumetric data. The dynamic mesh is based on refinement by edge bisection. Kobbelt and colleagues [18] propose a multiresolution shape representation based on geometry smoothing and dynamic meshes. The adaptation uses edge collapses, edge splits and edge flips.

Existing schemes for dynamic meshes are usually based on operations defined on edges because of their good adaptation properties. Multiresolution representations can be defined through global or local operations on a mesh [9]. In order to support adaptation, the multiresolution data structure has to be constructed using local operations. Progressive meshes [16] constitute one example of such data structure. Another example is the hierarchical 4K meshes [32]. In this kind of representation, different meshes can dynamically be extracted from the data structure. However, the local operations need to be explicitly stored.

For good fitting, stereo can be combined with shading [28], and anthropometric databases can be used to generate biometrically accurate models [6]. PCA decompositions are an extremely powerful way to fit, track and even recognize objects [5,22,3,24,26] at the cost of building a large database of examples beforehand. Additionally, a powerful deformable volumetric model has been used for fast face tracking [30] and subtle motion [35] capture.

Predictive filters [11], such as the *Kalman filter* [4,21] and *particle filters* [12,17,27], are usually used to complement and/or smooth out the observations made.

3. Deformations

Rigid models have their fair share of applications, but in many applications, such as animation and model-based tracking, we need models whose shape and attribute can change over time. Usually, these changes are controlled by a set of parameters, such that the entire model can fully be described by an instance of this, potentially large, parameter space.

As laser scanners become more affordable, it becomes increasingly easier to obtain detailed 3D rigid models of the surfaces of interest. Yet, the basic problem remains the same: how can we deform this surface according to our needs, such that it will work across different instances of this same object with minimal user interaction?

First, we need to fit the result of the scan to a "normalized" mesh, so as to have a basis from which to work. We apply a PCA-based method [3] to accomplish this fit. The result is a normalized 2D space representing the model's surface, called the u, v space, which describes the geometric features of the face (such as the corners of the eyes) as a function of u, v.

We now describe a novel method to describe deformations. It is a cascaded composition of *deformation fields* applied over the u, v space. Because this space is normalized, this method results in resolution and person-independent deformations.

3.1. Computation of model coordinates

In order to visualize, track, or process a model, we need to calculate the three-dimensional coordinates of each point $\vec{p}_i$, given the value of the deformation parameter vector $\vec{q}$. For every point p_i, in u_i, v_i there is a function $F_i = F_{u_i,v_i}$ such that

$$\vec{p}_i = F_i = F(\vec{q}, u_i, v_i) = F_{u_i,v_i}(\vec{q}). \qquad (1)$$

Conceptually, we represent the F_i as a cascade of basic mathematical operations, such as adding a vector to a point, scaled by a parameter [13]. Although simple, this representation is powerful enough to describe any number of complex deformations, and also connects well with the concept of deformation fields, which we describe in the following.

3.2. Deformation fields

Deformation fields are a resolution-independent way to describe deformations over the whole model. Such a field describes how deformations behave with respect to the continuous domain u, v of the model surface. This field can then be sampled at discrete u, v points, one per model point, to obtain the $F_{u,v}$; that is, the cascade of basic mathematical operations for the computation of the point's 3D coordinates.

There are three elements needed to define a deformation field: the type of deformation itself, a set of *vector fields*, and a set of parameters. Additionally, a deformation field might operate over the results of other deformations fields, allowing the compositions of results, which corresponds to the aforementioned cascade of mathematical operations for a model point.

A vector field V^f is a function $V^f : \mathbb{R}^2 \rightarrow \mathbb{R}^3$, defined over the u, v domain. It is used any time when a deformation field requires an u, v-dependent vector to represent its mathematical operation.

3.3. Two examples of deformation fields

Example 1. A *geometry image* [10] is a rigid model, which can be treated as a special case of a deformation field. We interpret the geometry image as a discretized description of a vector field. The *constant* deformation field takes only one vector field as a parameter, where its value at u, v provides us with the 3D position of the corresponding model point.

$$p_{u,v} = V^{\text{geo.image}}(u, v), \tag{2}$$

where $V^{\text{geo.image}}$ is the continuous vector field, obtained from the discretized geometry image through interpolation.

Example 2. A description of a PCA face [3] is also easily accomplished through a cascade of deformation fields. We describe the mean positions with a constant deformation field

$$p^0_{u,v}(\vec{q}) = V^{\text{mean}}(u, v), \tag{3}$$

where V^{mean} behaves as in the previous example. For every principal component k, we can define a vector field V^{pc_k}. We create a cascade of *AddVector* deformation fields, where deformation field k will take into account all PCA components $\leq k$. The kth deformation field takes the $k - 1$th deformation field and adds a parameter times a vector field. We describe this cascade as

$$p^k_{u,v}(\vec{q}) = p^{k-1}_{u,v}(\vec{q}) + q_k \cdot V^{pc_k}(u, v), \tag{4}$$

where q_k is the parameter responsible for the kth PCA component.

3.4. Deformable faces with deformation fields

A very simple deformable face, yet powerful and useful for tracking, can be defined as a cascade of a few carefully designed affine transformations; that is, AddVector deformation fields (Equation 4).

We start with a representation of the face at rest, which is in no way an easy task, and a fertile area of study in its own right [23,6,3]. First, we define the eyebrow deformations using a vector field, which is zero-valued outside the influence area of the eyebrows. The vectors are assigned monotonically decreasing magnitudes from a maximum at the eyebrows themselves, to zero at the border (Figure 1, top center). This deformation can be broken into two separate ones, if we require asymmetric eyebrow movements in the face. This deformation simulates the effect of the *frontalis* face muscle.

Simple lip movements can be modeled by the use of two different muscles — the *zygomatic major*, which pulls the lips and cheeks in the direction of the temples, and the *risorius*, which pulls lips and cheeks in the direction of the ears. Again, each deformation can be split into two for asymmetric movements (Figure 1, top right and bottom left).

Finally, jaw openings can be modeled by a constant opening area (chin), superior and inferior lips (that creates the elliptic opening of the mouth) and decreasing values up to the border (Figure 1, bottom center).

The cascading of these simple affine transformations, although with fairly complex vector fields, provides us with a deformable face model that is sufficiently powerful even for computer vision tracking from a single uncalibrated camera [14,15]. Our technique and framework allow us to use representations constructed from captured data [1], to design deformations by hand [14], and even to use hybrid approaches.

We should point out that for an arbitrary cascade of deformation fields, there are values of the parameter vector that lead to invalid models. Indeed, the valid subset of parameter vector values creates models that have no self-intersection. This property is strongly tied to the behavior of Jacobians and u, v gradients of the underlying deformation fields.

4. Adaptive Mesh

Recall that the deformable model is a discretization of the target surface. One of the most common discretization methods

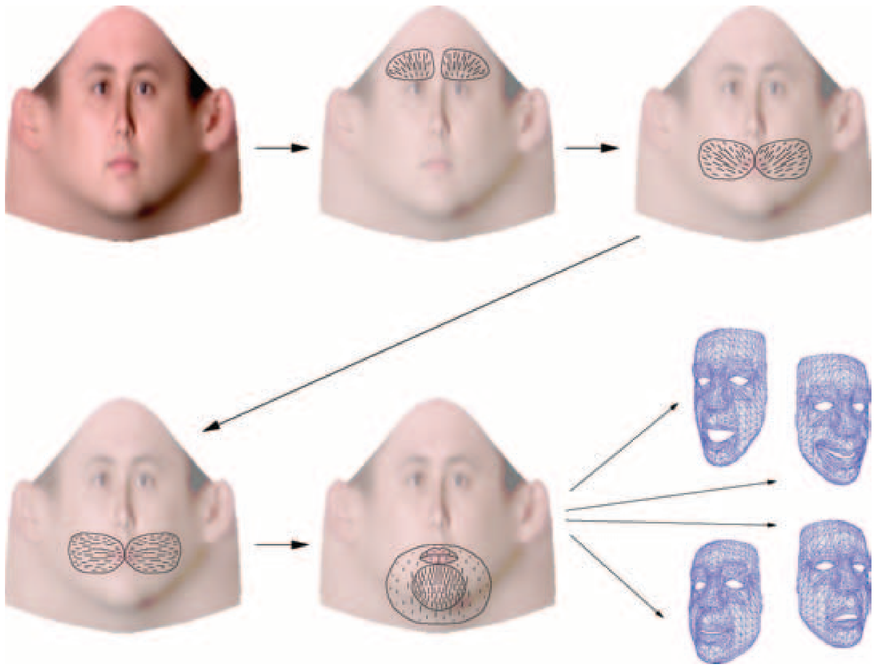

Figure 1: *Cascade of deformable fields that generate a simple face.*

is through a polygonal mesh that decomposes the base domain and gives a piecewise linear approximation of various properties defined over the domain — including the geometry in the case of a parametric surface.

In applications that employ deformable models it is desirable to have all the computations structured independently of a particular surface discretization. We have achieved one half of this requirement through the use of deformation fields over the u, v domain described in the previous section. The other half consists of developing methods to make the piecewise linear approximation sufficiently accurate for the purposes of the application.

For this reason, it is desirable to have a mesh library that can provide the application with a dynamic adaptive discretization of the surface based on problem-dependent criteria. In this way, it is possible to cleanly separate the application specific computations from the representation.

In the following, we describe a dynamic adaptive mesh library based on stellar operators that addresses most of the requirements of applications dealing with deformable models. This library encapsulates all the functionality for supporting the mesh representation. The implementation is robust, computationally efficient and economical in terms of memory usage. The library API is easy to use and provides the right level of abstraction for the application.

This library has the following features:

- The mesh is based on the half-edge, a standard topological data structure. Because the half-edge is widely adopted, there are many applications that can benefit from this library.
- The mesh has an underlying semi-regular multiresolution structure. As a result, no additional storage beside the current state of the mesh is required in the representation.
- The mesh dynamically changes its resolution based on general adaptation criteria specified by the application. This is implemented through refinement and simplification methods that maintain a restricted multiresolution.
- The library API includes operators for mesh creation, dynamic mesh adaptation and topological query operators.

The library is based on results from stellar subdivision theory and the notion of binary multitriangulation. We now give an overview of these concepts and describe the implementation of the library.

4.1. Splitting and welding

The key to an adaptive mesh representation is an underlying multiresolution structure. A multiresolution mesh, $H = (M_0, M_1, \dots, M_k)$, is a monotonic sequence of triangulations of a domain U, such that, $|M_i| \leq |M_j|$, for $i < j$, where $|M|$ is the number of triangles of a mesh M.

In general, the meshes $M_i \in H$ are related by operators that perform refinement or simplification to change the mesh resolution. It is desirable that these operators affect the mesh only *locally*. This property makes it possible to create many multiresolution sequences by piecing together independent local modifications.

Stellar subdivision theory provides suitable local operators to modify the resolution of a mesh. The stellar subdivision operators for two-dimensional triangle meshes are *the face split* and its inverse, the *face weld*; and the *edge split* and its inverse, the *edge weld* (Figure 2).

The main theorem of stellar theory asserts that these operators can transform between any two equivalent triangulations [20]. Moreover, stellar subdivision operators define the most localized atomic changes to a triangulation. We use stellar subdivision operators on edges as the building blocks for multiresolution meshes.

4.2. Semi-regular BMTs

The adaptive mesh maintains the multiresolution structure of a semi-regular binary multitriangulation. A *binary multitriangulation* (BMT) is a multiresolution structure formed by applying edge splits to an initial mesh, $\overline{M}$, called the *base mesh*, and producing a final mesh, $\underline{M}$, called the *full mesh*. The BMT can be thought of as a directed acyclic graph (DAG) describing all possible paths of local changes to the mesh [32,7]. In this DAG, arrows are labeled with stellar subdivisions on edges, and the vertices represent sub-meshes. Any cut in the DAG that separates $\overline{M}$ from $\underline{M}$ represents a valid mesh, called the *current mesh*.

A *regular binary multitriangulation* (RBMT) is a binary multitriangulation that satisfies the following conditions:

1. The base triangulation is the union of *basic blocks*.
2. Edge splits are only applied to interior edges of *basic blocks*.

A *basic block* is a pair of triangles with a common edge, called the *internal edge* of the block. The other edges are called *external edges*. In an RBMT, when the internal edge of a basic block is subdivided, new blocks are formed with the adjacent triangles and some additional edges may need to be subdivided. In that way, external edges of previous blocks become internal edges of new blocks. This process is called *interleaved refinement* and produces a *restricted quad-tree* [31] in which adjacent triangles differ at most by one resolution level.

The regularity of the RBMT allows us to store only the current mesh, yet allows us to refine or simplify the mesh according to the multiresolution structure.

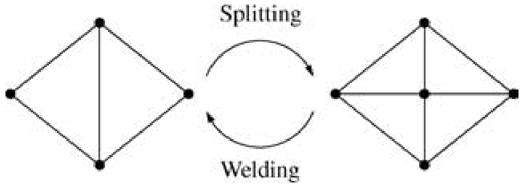

Figure 2: *Splitting and welding operations.*

5. Adaptive Refinement and Simplification

The mesh adaptation is implemented in the library by enforcing a RBMT structure using operators for restricted refinement and simplification.

The library assumes that shape information is known to the application independently of the mesh. More specifically, the following requirements need to be satisfied:

- There is a base domain in which the surface geometry and other properties are defined;
- It is possible to take and compute samples of points on the surface.

The application provides four functions to the library through a surface object: a procedure to construct a polyhedron that defines the geometry and topology of the base domain; a sampling function to evaluate the geometry of the surface at a vertex of the mesh; and refinement and simplification tests to determine where the mesh needs further subdivision or coarsening.

The library API for mesh construction and adaptation is composed of a mesh constructor, that takes as a parameter of the surface object, and methods for adaptive refinement and simplification of the mesh. These methods use the adaptation test functions and the sampling function of the surface.

The functionality of the mesh library is sufficient to create adaptive meshes of dynamic surfaces for our application of deformable models. In this context, some important specific aspects are:

- The base mesh incorporates all the important features of the surface;
- The surface is defined by a parametric function;
- The criteria for mesh adaptation are derived from the deformations driven by tracking, discussed in the following sections.

5.1. Obtaining the base resolution

One of the secrets for this method to work well is a properly built base resolution mesh. Ideally, the coarsest model should already have the appropriate topological structure for the adaptive mesh. Additionally, the model triangles should have approximately the same surface area, for a visually pleasant refining process.

We satisfy the first requirement by enforcing compliance of the base mesh with the topological requirements of the adaptation scheme. The mesh library provides functionality to bring the base mesh into the required form through a one-time application of carefully selected edge splits.

Initial similar-sized triangles, on the other hand, imply a good balance between minimal detail selection, and the variance of the edge sizes. So far we have maintained this balance through manual selection of points in the u, v space, and triangulation of this space with constraints (with the help of triangle (`http://www-2.cs.cmu.edu/~quake/triangle.html`) to find the base connectivity. We then use the deformation fields themselves to generate the appropriate base geometry.

In Figure 3(a) we show a hand-crafted base model constructed from coordinates in u, v, and in Figure 3(b) we show the model after making it compliant with the RBMT structural requirements.

5.2. Adapting the mesh to the model

Once we have the base model at the coarsest possible resolution, we have to decide by how much to refine the mesh to obtain a good-looking model. The optimal level of detail depends on both the facial characteristics of the person, to whom the mesh is being adapted, and which area of the face we are looking at. An additional concern is the complexity of the refined model — as the number of triangles increases, so does memory consumption, and computational overhead for the animation of the model. In the following, we compare three different refinement strategies.

5.2.1. *Unconditional refinement*

This strategy is the simplest one possible: repeatedly refine a model a fixed number of times (three times in the examples presented in this paper). The advantages are that this refinement method is very fast, because there are no time-consuming tests, and that it is guaranteed to yield a good-looking model with a high level of detail. Offsetting the advantages, however, is the indiscriminately high number of vertices and triangles in the resulting model, with all the complexity disadvantages mentioned earlier.

5.2.2. *Normals-based refinement*

The idea behind this strategy is to obtain a smooth model by breaking up sharp ridges and valleys, and jagged angles at the model edges. Ridges and valleys are defined by the angle between the triangle normals of any two triangles that share a common edge. If it falls above a threshold, we refine the edge. Similarly, jagged angles at the model edges occur when the triangle orientations at the endpoints differ greatly from one another. To compute the difference, we average the triangle normals of the triangles surrounding each respective endpoint. If the angle between the averages falls above a threshold, we refine the edge in question.

This method is computationally inexpensive, and results in reasonable approximations of the model surface contour. The number of vertices and triangles in the resulting model is much lower than with the unconditional refinement strategy. On the other hand, if there are indeed sharp ridges in the contour, such as around the eyes and the lips, this method keeps refining unnecessarily without a corresponding increase in the appearance of the model.

5.2.3. *Surface contour error-based refinement*

This strategy directly computes the quality of the polygonal approximation to the true contour of the model. We define the *error* of a polygon *Poly* at a point (u, v) to be

$$\varepsilon_{Poly}(u, v) = |p_{u,v} - P_{Poly}(u, v)|, \tag{5}$$

where $p_{u,v}$ is the 3D position of the contour and $P_{Poly}(u, v)$ is the 3D position of the polygon at coordinates (u, v).

Then the total error in the approximation of the surface contour with the polygon is

$$\varepsilon_{Poly} = \int_{Poly} \varepsilon_{Poly}(u, v)\, du\, dv. \tag{6}$$

The average error is obtained by dividing ε_{Poly} by the area of the polygon. If it falls above a threshold, we refine this polygon. In addition, we compute the maximum error

$$\max_{u,v}\{\varepsilon_{Poly}(u, v)\}, \tag{7}$$

which helps capturing peaks and troughs in the model surface contour, and lets us refine accordingly.

This method yields a visually pleasing refinement of the model with far fewer vertices and triangles than with either of the two other methods (Figure 3(e)), at a higher computational cost. Evaluating this criterion requires a discrete approximation of the integral in Equation (6), via sampling the polygon. For the models shown in Figure 3, this process is three times slower than normals-based and unconditional refinement. Therefore, this criterion is best suitable for applications where we change the model parameters frequently, and refine the model only infrequently. Additionally, this method requires us to measure, even if only approximately, the true contour of the model. Such a contour can be provided by a *geometry image* giving a discrete sampling of the underlying u, v space.

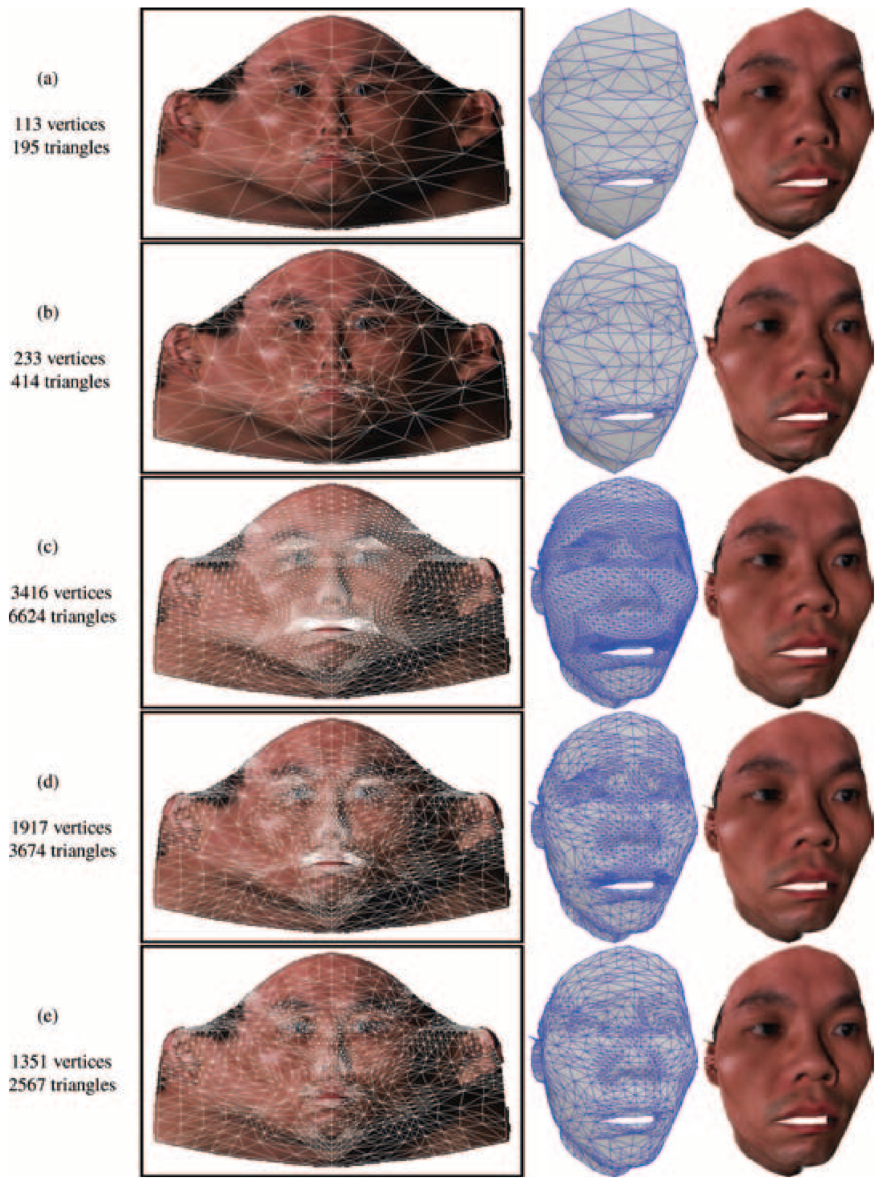

Figure 3: *(a) Original mesh, (b) mesh in RBMT form, (c) unconditional refinement to constant depth, (d) normals-based refinement and (e) surface contour error-based refinement. Left to right: Mesh in u–v space, polygonal mesh, face with texture applied.*

6. Deformations Fields for Face Fitting

A lot of research has gone into creating a three-dimensional model from one or more images [3,8,24]. Among these techniques, PCA-based decompositions of models have proven effective and well behaved when single images are the source of information. Of course, PCA can only be used once enough data have been collected to find the principal components.

In this paper, we cast a PCA decomposition of faces as a series of deformation maps that affect a basic shape (Equation 4). The basic shape is the mean of the PCA decomposition, and represents a neutral, genderless model.

For fitting, we manually create correspondences between some points in the coarse discretization of the face and their desired locations in the image. Then, we search for the values of the parameters (Equation 4) that best align the model to the

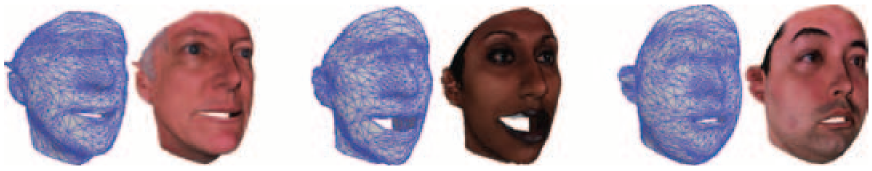

Figure 4: *Different faces under deformations.*

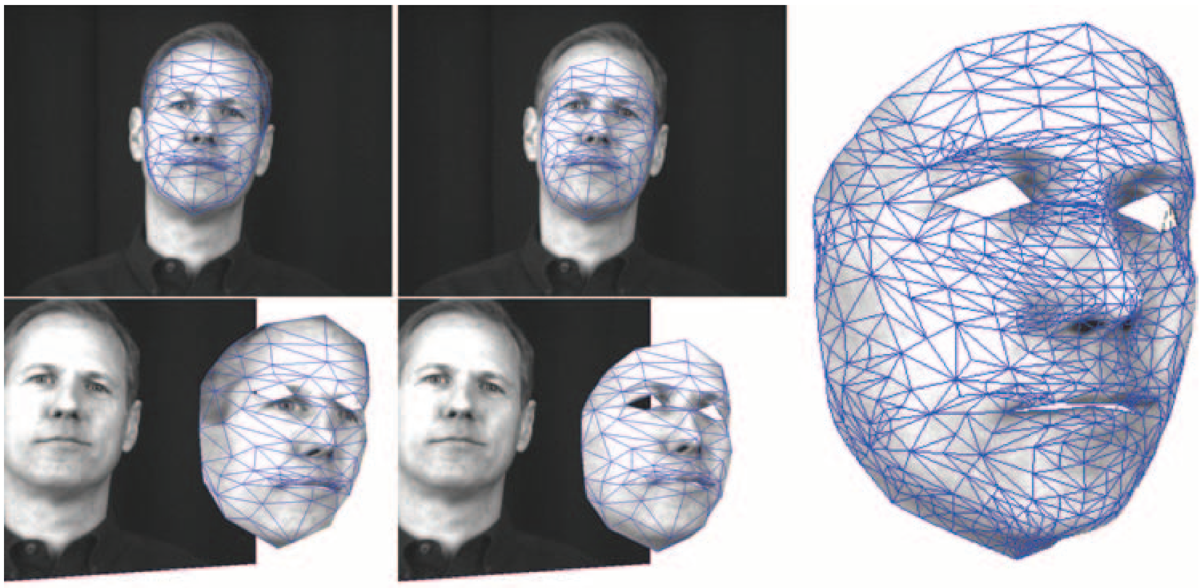

Figure 5: *Fitting of a subject, used in the facial expression tracking of a native American sign language signer telling a story. Left to right: Initial shape for fitting process, an intermediate stage and the final refined shape using the contour criterion.*

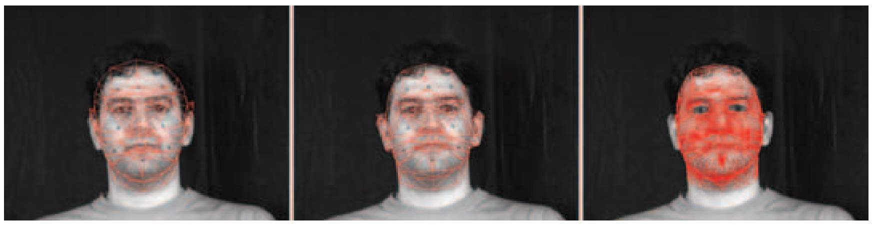

Figure 6: *Fitting for the subject used in the quantitative validation of our tracking technique. Left to right: Initial shape for fitting process, an intermediate stage and the final refined shape using the contour criterion.*

image. Because there are rotations and projections involved, this search is a nonlinear optimization. We look for $\vec{q}$ that minimizes

$$error = \sum |p_{u,v}\left(\vec{q}\right) - corr_{u,v}|^2, \qquad (8)$$

where $corr_{u,v}$ is the correspondence, the goal position, of $p_{u,v}$. In Appendix A, we explain in more detail how to perform this optimization.

In our implementation, we used a semi-automated tool to allow us to adjust initially guessed 2D positions based on the coarse orientation/translation of the face. For fitting, between 80 and 160 correspondences usually yield a good result for a 100-dimension PCA space.

In Figures 6 and 5, we show the initial generic model (the mean of the PCA decomposition), an intermediate state during the optimization procedure, and the final fit after refinement using the contour-based criterion for two different subjects. After establishing the desired model–image

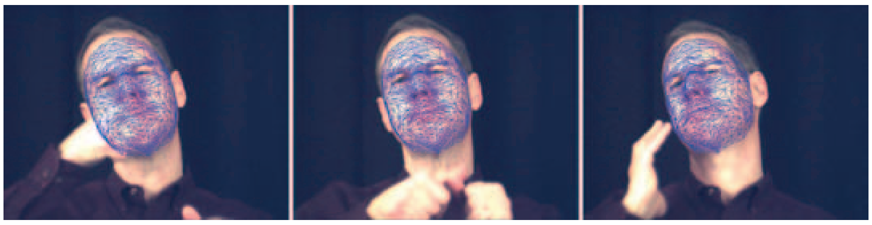

Figure 7: *Three snapshots from the ASL storytelling sequence with partial occlusions. The sequence was signed by a native signer and used a model based on the contour criterion.*

correspondences, the optimization process takes under 5 s to find the solution. To help the nonlinear optimization, we manually give an initial approximate orientation and translation as a starting point.

7. Deformable Retargetable Faces

We have coupled the library described in Section 4 with an implementation of the deformation fields in Section 3 and the adaptation strategies in Section 5 to our deformable model simulation and tracking system [14,15]. Figure 3 shows that our description with deformation fields is resolution independent. At this level of detail, animation of the facial parameters can be done in real time on a Pentium 4 system running at 2.4 GHz. The refinement of the base mesh takes 0.7 s for the unconditional and normals criteria, and 2.5 s for the contour-based criterion.

Because the u, v space is normalized across different human faces (Section 3), in theory the deformation maps should apply to any human face that can be represented in this normalized space. To test this hypothesis, we applied the deformations to a few of the subjects of the dataset PCA dataset of [3], seven in total. An example selection of faces is shown in Figure 4. This figure, as well as the supplemental movie, show that our method indeed makes retargeting feasible with minimal effort.

8. Deformable Model Tracking

The goal of deformable model tracking is to estimate the value of the parameters that best describe the model for every frame of a video sequence. Tracking is an inductive process, and requires a good estimate of the model at the first frame. The basic shape comes from the technique described in Section 6, and the deformation parameters are also obtained in a similar way. The tracking technique we use in this section has been thoroughly explained in [14,15,33], and here we only validate the use of our new model description methodology.

We use low-level computer vision techniques (such as a point tracker, edge tracker and optical flow) to find image correspondences between the current and the new images. Since we know where each model point is in the current image, we can establish model–image correspondences as in Equation (8) [14]. Since displacements are usually small across subsequent frames, the current value of the parameters is a good initial estimate for the nonlinear optimization (Appendix A).

Different low-level algorithms exploit distinct local characteristics of the image. A point tracker is only effective over areas rich in texture [29], while an edge tracker behaves well around zeros of the second derivative [14]. Since we ultimately look for model–image correspondences, we only evaluate image points that are vertices of our model. This means that the more points a model has, more candidates for good correspondences it will have. Unfortunately, more points also require more processing time and memory capacity, and decrease the performance of tracking. Thus, there is a compromise between performance and accuracy, and it is necessary to look for the best possible refinement of the model for tracking.

In the following, we discuss two experiments, which tracked a realistic data sequence, and a validation sequence for quantitative evaluation. The first experiment consisted of tracking the face of a native American sign language (ASL) storyteller during a narration. We obtained a video sequence featuring Dr. Benjamin Bahan from the National Center for Sign Language and Gesture Resources (data collected from native signers at Boston University under the supervision of Carol Neidle and Stan Sclaroff. Available online at `http://www.bu.edu/asllrp/ncslgr.html`) captured in full color at a resolution of 640 × 480 at 60 frames per second. Tracking such videos is close to being as difficult as face tracking can ever get, because of fast movements and frequent occlusions of the face by the hands. Successful tracking requires a detailed model, and thus provides an ideal test bed for adaptive deformable models. Figure 7 shows some snapshots from tracking this sequence with the contour-refined model from Figure 5.

The second experiment consisted of quantitative validation. We collected a special sequence of images, where using makeup, we drew markers (black dots) on the face of the subject before capture. Later, using a semi-automated procedure, we determined the 2D position of these markers on every frame of the sequence. The sequence was a bit longer

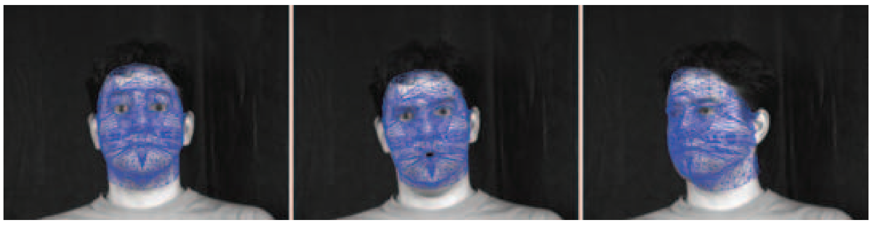

Figure 8: *Three snapshots from the validation tracking sequence with contour-based refinement.*

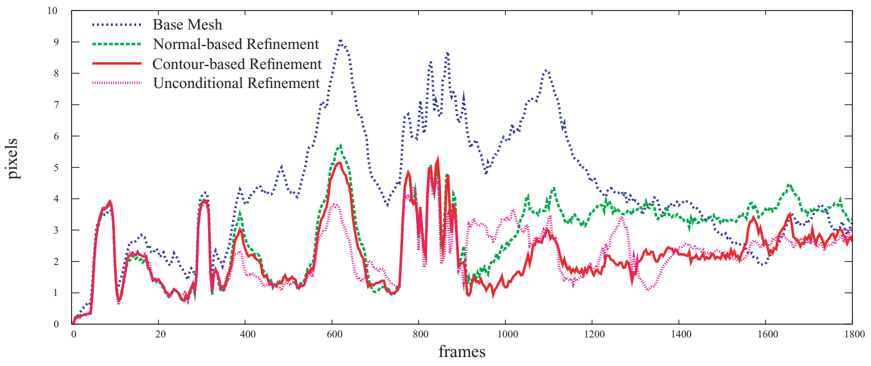

Figure 9: *Accuracy comparison of tracking with different refinements against the ground truth.*

than 30 s (3600 frames), and was captured at 60 Hz. In order to make these experiments closer to what we can expect from web cam images, we downsampled the image to a resolution of 320 × 240, and used only one in every four frames (effectively tracking at a frame rate of 15 Hz).

In the tracking validation, we disallowed the selection of any points by the low-level vision algorithms that were too close to the markers, so that the markers would not affect the tracking process. We also specified the points on the model that corresponded to each marker, and projected them into image space at every frame, so as to estimate the 2D positions of the markers. The distance between our estimate and the real 2D position of the markers determined the error of the tracking.

In Figure 8, we show three snapshots of tracking this sequence, and in Figure 9, we compare the accuracy of the tracking process against ground truth. We show the error plot across the sequence of the tracking using the base model, using the normal-based refinement, using the contour-based refinement and using the unconditional refinement.

The results in Figure 9 clearly show that the base mesh has the worst behavior. This is explained by the small number of vertices of the model. Tracking with a mesh obtained through the normal-based refinement yields a smaller error than the base mesh, and tracking using a mesh obtained with the contour-based refinement yields an even smaller error. The contour-based refinement mesh is as good as the mesh obtained with an unconditional refinement, showing that the extra details are not necessary for good tracking.

Thus, intelligent refinement of the model improves the accuracy of the tracking, while still saving a large amount of computation time compared to unconditional refinement. Generally, the fewer nodes, the faster is the tracking. On an AMD Opteron 246 system normals-based refinement (1200 nodes, 2289 triangles) took 0.4–0.5 s per frame, and contour-based refinement (1900 nodes, 3600 triangles) took 0.5–0.6 seconds per frame. In contrast, unconditional refinement (4400 nodes, 8400 triangles) took more than twice as long.

9. Conclusions and Future Work

In this paper, we have described a unified approach for adaptive deformable models, with human faces as a case study. This approach is based on representing the model geometry in a normalized u, v space, and defining deformations via deformation maps and vector fields operating on this space. This method is sufficiently powerful to make the definition of models and deformations both resolution independent and retargetable.

The framework for mesh adaptation described in this paper is easy to use. The logic for when to refine and simplify the mesh can be encapsulated in just two functions. We have shown by way of three different refinement strategies that it is easy to obtain just the right level of detail for the application.

The quality of the refined models is closely related to the underlying structure of the coarsest level of the mesh. Constructing this base mesh is not a straightforward task. The model has to be simple, while characterizing important landmarks. In addition, all neighboring triangles should have similar sizes. In this paper, we have built the base mesh manually, and an automated method for doing so remains an open problem.

Applying these techniques to computer vision tracking allows us to strike a good balance between tracking accuracy and computational cost. Future work will consist of making the face adaptive across frames during the tracking, so that at all times the best image features can be chosen and closely matched to model points. Doing so will require a working set of view-based and texture-based adaptation criteria. We are also investigating possible techniques to learn deformations from real data in an automated manner.

Appendix A: Solving the Nonlinear Optimization

Deformable model tracking is the inverse of our computer graphics application. Instead of manipulating the model vertices via the parameters $\vec{q}$, we would like to estimate $\vec{q}$ such that the model best fits the observations from the video images. Because the F_i in Equation (1) are highly nonlinear, we cannot solve for $\vec{q}$ directly. Instead, we find the solution iteratively through a gradient descent optimization procedure.

This optimization is based on estimating the 2D displacements between selected points of the projection of the model into the current image, and where these points should be. These displacements $\vec{f}_i$, also called *image forces*, can be estimated from frame to frame through 2D image processing algorithms, such as point tracking edge tracking, and optical flow, and are described in more detail in [14]. Note that for the fitting problem described in Section 6, the image forces are the individual displacements in Equation (8). The optimization then consists of finding the $\vec{q}$ that minimizes $\sum_i ||\vec{f}_i||$.

The first step is to project all image forces into a single contribution in parameter space, called the *generalized force*.

$$\vec{f}_g = \sum_i B_i^\top \vec{f}_i, \tag{9}$$

where

$$B_i = \left.\frac{\partial Proj}{\partial \vec{p}}\right|_{\vec{p}_i} J_i \tag{10}$$

is the projection of the Jacobian matrix J_i of a model vertex $\vec{p}_i$ into image space. This Jacobian matrix is defined as follows:

$$J_i(\vec{q}\,) = \frac{\partial \vec{p}_i}{\partial \vec{q}} = \begin{bmatrix} | & | & & | \\ \frac{\partial \vec{p}_i}{\partial q_1} & \frac{\partial \vec{p}_i}{\partial q_2} & \cdots & \frac{\partial \vec{p}_i}{\partial q_n} \\ | & | & & | \end{bmatrix}. \tag{11}$$

Using the generalized force $\vec{f}_g$, we solve the dynamical system

$$\dot{\vec{q}} = K\vec{q} + \vec{f}_g \tag{12}$$

(where K is a stiffness matrix, which can be zero) via small Euler steps:

$${}^{k+1}\vec{q} = {}^{k}\vec{q} + \delta \sum_i B_i^\top\big|_{{}^k\vec{q}}\ f_i, \tag{13}$$

where δ is the Euler step size. Choosing δ properly is highly application dependent; in our case, we use a value of 10^{-3}. Conceptually, Euler integration of this system is equivalent to gradient descent. It typically converges within 600 iterations.

In its basic form, this system is vulnerable to outliers in the image forces. We therefore insert an extra step before calculating the $\vec{f}_g$ in Equation (9), which gets rid of any outlying $\vec{f}_i$. This step, described in [33], projects the image forces individually into parameter space via calculating $B_i^\top \vec{f}_i$, then computes their mean and covariance via the robust minimum covariance determinant statistical estimator [25], and rejects all forces whose Mahalanobis distance to the robust mean falls above a threshold.

Acknowledgements

The research in this paper was supported by NASA Cooperative Agreements 9-58 with the National Space Biomedical Research Institute, CNPq, FAPESP, FAEPEX-Unicamp 1679/04, research scientist funds by the Gallaudet Research Institute, and NSF CNS-0427267.

References

1. V. Blanz, C. Basso, T. Poggio and T. Vetter. Reanimating faces in images and video. In *Proceedings of EUROGRAPHICS*, 2003.

2. R. Bowden, T. Mitchell and M. Sahardi. Real-time dynamic deformable meshes for volumetric segmentation and visualization. In *Proc. BMVC 11*, pp. 310–1319, 1997.

3. V. Blanz and T. Vetter. A morphable model for the synthesis of 3D faces. In *SIGGRAPH* pp. 187–194, August 1999.

4. G. Bishop and G. Welch. An introduction to the kalman filter. In *SIGGRAPH 2001 Course Notes*, 2001.

5. M. Dimitrijevic, S. Ilic and P. Fua. Accurate face models from uncalibrated and ill-lit video sequences. In *Proc. of IEEE Computer Vision and Pattern Recognition*, pp. 1034–1041, 2004.

6. D. DeCarlo, D. Metaxas and M. Stone. An anthropometric face model using variational techniques. In *SIGGRAPH* pp. 67–74, 1998.

7. L. Floriani, P. Magillo and E. Puppo. Efficient implementation of multi-triangulations. In *IEEE Visualization '98,* Ebert D., Hagen H., Rushmeier H., Eds., pp. 43–50, 1998.

8. P. Fua. Regularized bundle-adjustment to model heads from image sequences. *International Journal of Computer Vision*, 38(2): 153–171, 2000.

9. M. Garland. Multiresolution modeling: Survey & future opportunities. In *Eurographics '99, State of the Art Report (STAR)*, 1999.

10. X. Gu, S. Gortler and H. Hoppe. Geometry images. In *SIGGRAPH* pp. 355–361, 2002.

11. S. Goldenstein. A gentle introduction to predictive filters. *Revista de Informatica Teorica e Aplicada RITA XI*, 1, 61–89, 2004.

12. N. Gordon, D. Salmon and A. Smith. A novel approach to nonlinear/nonGaussian bayesian state estimation. In *IEEE Proc. Radar Signal Processing*, vol. 140, pp. 107–113, 1993.

13. S. Goldenstein, C. Vogler and D. Metaxas. Directed acyclic graph representation of deformable models. In *Proc. of IEEE workshop on motion and video computing*, 2002.

14. S. Goldenstein, C. Vogler and D. Metaxas. Statistical cue integration in DAG deformable models. In *IEEE Trans. on Pattern Analysis and Machine Intelligence 25*vol. 7, July 2003.

15. S. Goldenstein, C. Vogler and D. Metaxas. 3D facial tracking from corrupted movie sequences. In *Proc. of IEEE Computer Vision and Pattern Recognition*, 2004.

16. H. Hoppe. Efficient implementation of progressive meshes. *Computers & Graphics 22*, 1, pp. 27–36, 1998.

17. M. Isard and A. Blake. CONDENSATION: conditional density propagation for visual tracking. *International Journal of Computer Vision 29*, 1, 5–28, 1998.

18. L. Kobbelt, T. Bareuther and H. Seidel. Multiresolution shape deformations for meshes with dynamic vertex connectivity. *Computer Graphics Forum 19*, 3, 2000.

19. M. Kass, A. Witkin and D. Terzopoulos. Snakes: Active Contour Models. *International Journal of Computer Vision 1* 321–331, 1988.

20. W. B. R. Lickorish. Simplicial moves on complexes and manifolds. In *Proceedings of the Kirbyfest* vol. 2, pp. 299–320, 1999.

21. P. Maybeck. *Stochastic Models, Estimation, and Control*. Academic Press, New York, 1979.

22. H. Murase and S. K. Nayar. Visual learning and recognition of 3-D objects from appearance. *International Journal of Computer Vision 14*, 5–24, 1995.

23. F. Pighin, J. Hecker, D. Lischinski, R. Szeliski and D. Salesin. Synthesizing realistic facial expressions from photographs. In *SIGGRAPH*, pp. 75–84, 1998.

24. F. Pighin, R. Szeliski and D. Salesin. Modeling and animating realistic faces from images. *International Journal of Computer Vision 50*, 2, 143–169, 2002.

25. P. J. Rousseeuw and K. V. Driessen. A fast algorithm for the minimum covariance determinant estimator. *Technometrics 41*, 212–223, 1999.

26. A. Romdhani and T. Vetter. Efficient, robust and accurate fitting of a 3D morphable model. In *Proc. of Intl. Conf. of Computer Vision*, pp. 59–66, 2003.

27. A. Smith and A. Gelfand. Bayesian statistics without tears: A sampling-resampling perspective. *American Statistician 46*, 2, 84–88, 1992.

28. D. Samaras, D. Metaxas, P. Fua and Y. Leclerc. Variable albedo surface reconstruction from stereo and shape from shading. In *Proc. of IEEE Computer Vision and Pattern Recognition*, pp. 480–487, 2000.

29. J. Shi and C. Tomasi. Good features to track. In *Proc. of IEEE Computer Vision and Pattern Recognition*, pp. 593–600, 1994.

30. H. Tao and T. Huang. Visual estimation and compression of facial motion parameters: elements of a 3D model-mased video coding system. *International Journal of Computer Vision 50*, 2, 111–125, 2002.

31. B. Von Herzen and A. H. Barr. Accurate triangulations of deformed, intersecting surfaces. In *Computer Graphics SIGGRAPH '87 Proceedings*, vol. 21, pp. 103–110, July 1987.

32. L. Velho and J. Gomes. Variable resolution 4-k meshes: Concepts and applications. *Computer Graphics forum 19*, 195–212, 2000.

33. C. Vogler, S. Goldenstein, J. Stolfi, V. Pavlovic and D. Metaxas. Outlier rejection in high-dimensional deformable models. *Image and Vision Computing*, 2005 (in press).

34. A. Witkin, M. Gleicher and W. Welch. Interactive dynamics. *Computer Graphics*, vol. 24, pp. 11–21, 1990

35. Z. Wen and T. Huang. Capturing subtle facial motions in 3D face tracking. In *Proc. of Intl. Conf. of Computer Vision*, pp. 1343–1350, 2003.