Adaptive Deformable Models

SIOME GOLDENSTEIN[1], CHRISTIAN VOGLER[2], LUIZ VELHO[3]

[1]UNICAMP–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@visgraf.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 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. Additionally, we show how our technique is suitable to describe different individuals.

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 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.

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; too many vertices and it will be computationally too expensive to perform all calculations, since many algorithms have higher than linear complexity. Either, 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. 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).

We present the deformation and animation of human faces as a concrete application of the techniques described in this paper (Sections 6). We derive a simple set of deformations and show that they are indeed face independent.

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 [15, 24, 8, 4, 21, 11, 3]. In computer vision, these models have been used for applications such as tracking, fitting, recognition, and segmentation [15, 6, 2, 18]. Human faces are a particular type of object that attracts a lot of attention [4, 7, 20, 21, 11].

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. [5] employs an inflating balloon model to reconstruct a surface from volumetric data. The dynamic mesh is based on refinement by edge bisection. [16] proposes 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 [10]. In order to support adaptation, the multiresolution data structure has to be constructed using local operations. Progressive meshes [14] constitute one example of such data structure. Another example are the hierarchical 4-K meshes [22]. In this kind of representation, different meshes can be dynamically extracted from the data structure. However, the local operations need to be explicitly stored.

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 be fully 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 [4] 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 $\vec{q}$. For every point p_i, in u_i, v_i there is a function $F_i = F_{u_i,v_i}$ such that

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

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 [1]. 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* [13] 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 the discretized geometry image through interpolation.

Example 2. A description of a PCA face [4] 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 like 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 k^{th} deformation field takes the $k-1^{\text{th}}$ 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 k^{th} 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 [19, 8, 4]. 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 until 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 [11, 12]. Our technique and framework allow us to use representations constructed from captured data [3], to design deformations by hand [11], and even to use hybrids approaches.

4 Adaptive Mesh

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

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 multi-triangulation. 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, $H = (M_0, M_1, \ldots, M_k)$, 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

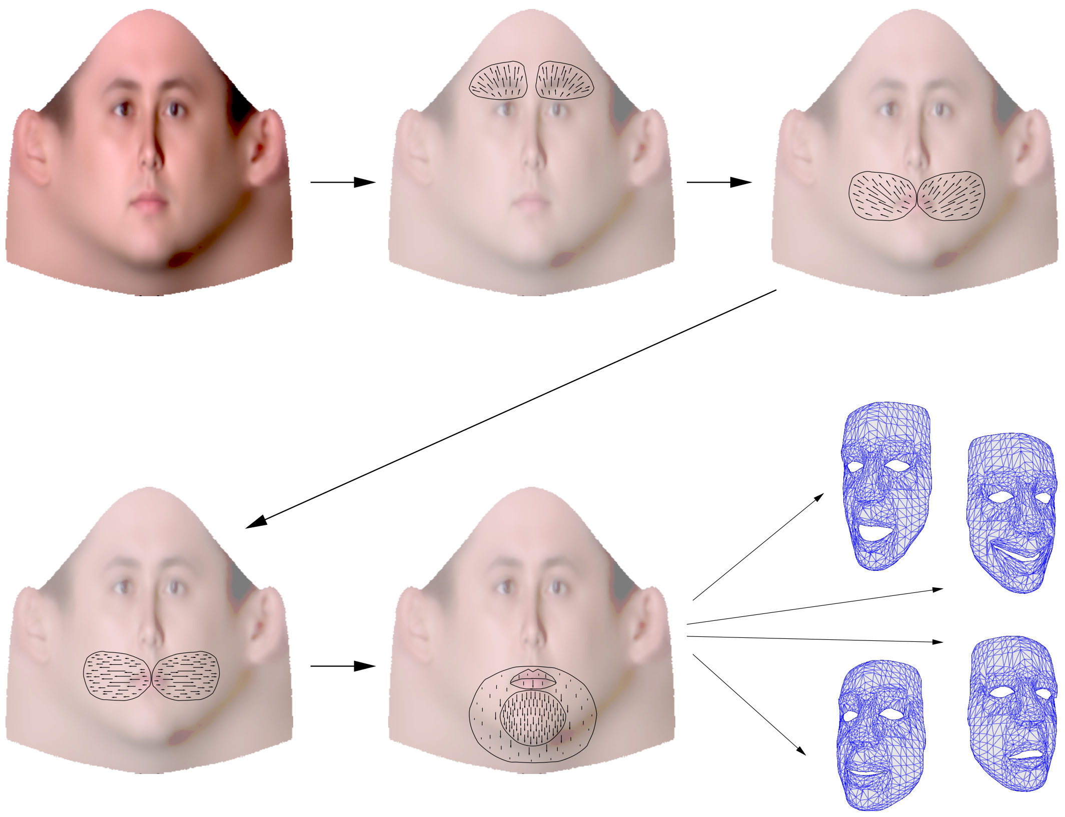

Figure 1: Cascade of deformable fields that generate a simple face.

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*.

The main theorem of stellar theory asserts that these operators can transform between any two equivalent triangulations [17]. 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 multi-triangulation. A *binary multi-triangulation* (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 [22, 9]. In this DAG, arrows are labeled with stellar subdivisions on edges, and the vertices represent submeshes. Any cut in the DAG that separates $\overline{M}$ from $\underline{M}$ represents a valid mesh, called the *current mesh*.

A *regular binary multi-triangulation* (RBMT) is a binary multi-triangulation 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* [23], 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.

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 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`[1]) to find the base connectivity. We then use the deformation fields themselves to generate the appropriate base geometry.

In Figure 2a we show a hand-crafted base model constructed from coordinates in u, v, and in Figure 2b 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

[1]`http://www-2.cs.cmu.edu/~quake/triangle.html`

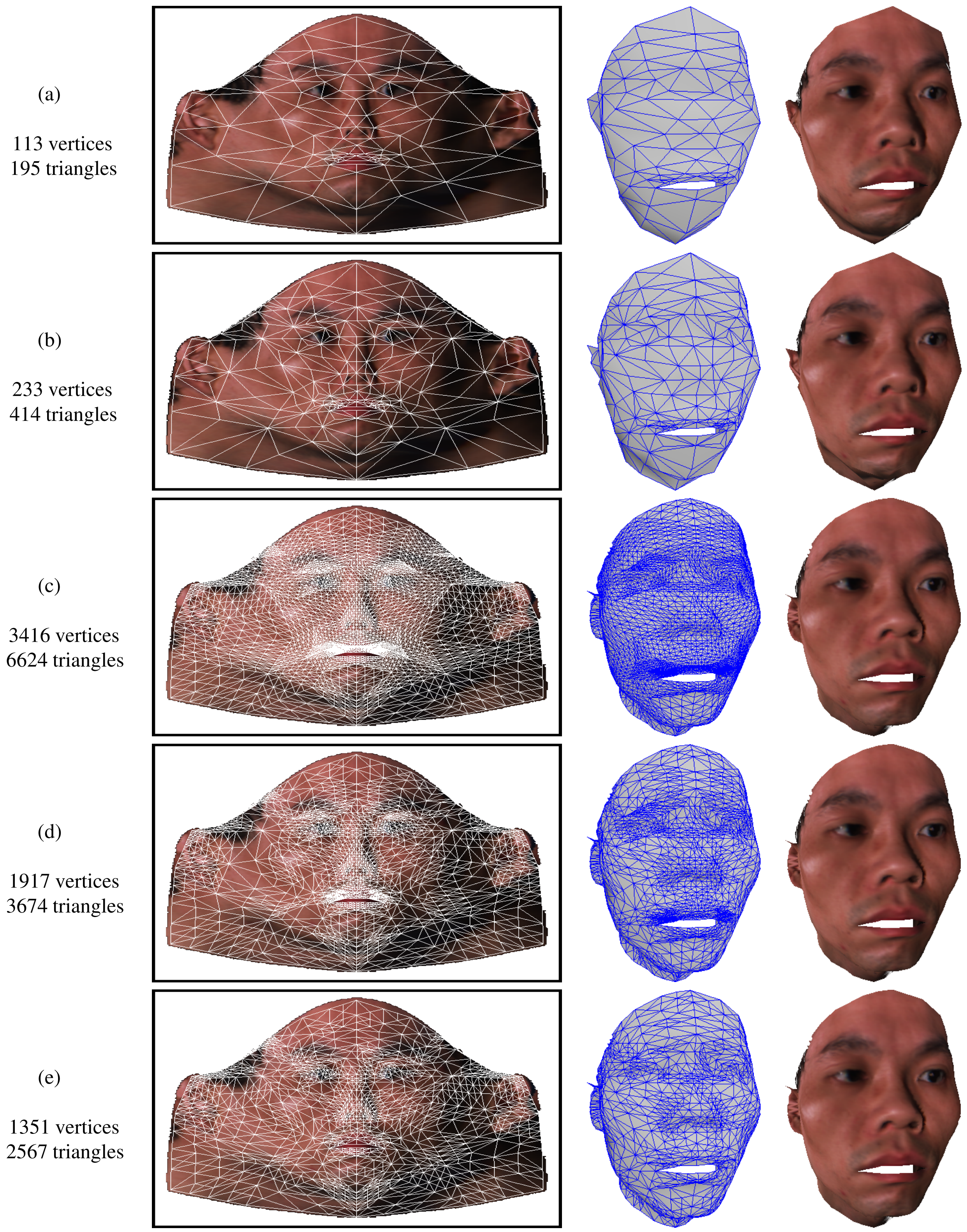

Figure 2: (a) Original mesh, (b) Mesh in RBMT form, (c) Unconditional refinement to constant depth, (d) Normals-based refinement, (e) Surface contour error-based refinement. Left to right: Mesh in u-v space, Polygonal mesh, Face with texture applied.

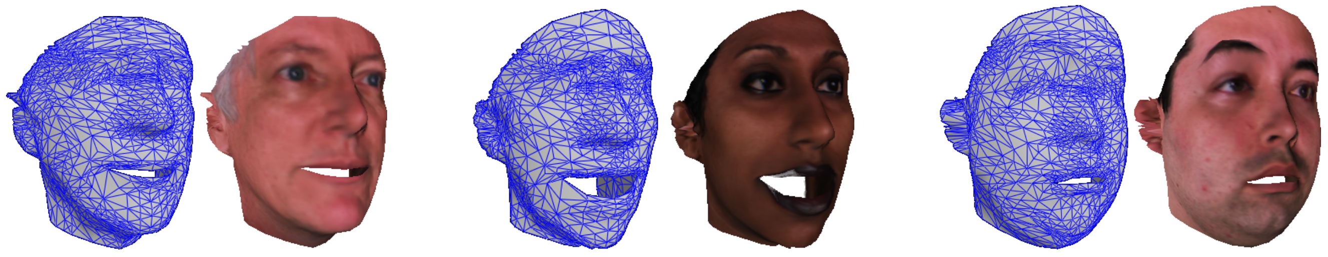

Figure 3: Different faces under deformations.

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:

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

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

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

which helps capture 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 2e), 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 2, 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.

6 Case Study: 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 [11, 12]. Figure 2 shows that our description with deformation fields is resolution-independent. At this level of detail, animation of the facial parameters can be done in realtime on a Pentium 4 system running at 2.4 GHz. The refinement of the base mesh takes 0.7 seconds for the unconditional and normals criteria, and 2.5 seconds 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 [4], seven in total. An example selection of faces is shown if Figure 3. This figure, as well as the supplemental movie, show that our method indeed makes retargeting feasible with minimal effort.

Hence, we have three immediate benefits: we can animate an arbitrary face, do model-based tracking of an arbitrary person (after proper fitting), and retarget facial motions.

7 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.

Currently, we are in the process of applying these techniques in computer vision, for deformable face tracking. For that we will need 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.

References

[1] Authors. Removed for blind review. In *Proc. of IEEE Workshop on Motion and Video Computing*, 2002.

[2] A. Blake and M. Isard. *Active Contours: The Application of Techniques from Graphics,Vision,Control Theory and Statistics to Visual Tracking of Shapes in Motion*. Springer-Verlag, 1998.

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

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

[5] R. Bowden, T. Mitchell, and M. Sahardi. Real-time dynamic deformable meshes for volumetric segmentation and visualization. *Proc. BMVC*, 11:310–1=319, 1997.

[6] T. Cootes and C. Taylor. Active shape models - their training and application. *CVIU*, 61(1):38–59, 1995.

[7] D. de Carlo and D. Metaxas. Optical flow constraints on deformable models with applications to face tracking. *IJCV*, 38(2):99–127, July 2000.

[8] D. DeCarlo, D. Metaxas, and M. Stone. An anthropometric face model using variational techniques. In *SIGGRAPH*, pages 67–74, 1998.

[9] L. Floriani, P. Magillo, and E. Puppo. Efficient implementation of multi-triangulations. In David Ebert, Hans Hagen, and Holly Rushmeier, editors, *IEEE Visualization '98*, pages 43–50, 1998.

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

[11] S. Goldenstein, C. Vogler, and D. Metaxas. Statistical Cue Integration in dag Deformable Models. *IEEE Trans. PAMI*, 25(7), July 2003.

[12] S. Goldenstein, C. Vogler, and D. Metaxas. 3D facial tracking from corrupted movie sequences. In *CVPR*, 2004.

[13] X. Gu, S. Gortler, and H. Hoppe. Geometry images. In *SIGGRAPH*, pages 355–361, 2002.

[14] H. Hoppe. Efficient implementation of progressive meshes. *Computers & Graphics*, 22(1):27–36, 1998.

[15] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active Contour Models. *IJCV*, 1:321–331, 1988.

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

[17] W. B. R. Lickorish. Simplicial moves on complexes and manifolds. In *Proceedings of the Kirbyfest*, volume 2, pages 299–320, 1999.

[18] D. Metaxas. *Physics-based Deformable Models: Applications to Computer Vision, Graphics and Medical Imaging*. Kluwer Academic Publishers, 1996.

[19] F. Pighin, J. Hecker, D. Lischinski, R. Szeliski, and D.H. Salesin. Synthesizing realistic facial expressions from photographs. In *SIGGRAPH*, pages 75–84, 1998.

[20] F. Pighin, R. Szeliski, and D. Salesin. Modeling and animating realistic faces from images. *IJCV*, 50(2):143–169, 2002.

[21] H. Tao and T. Huang. Visual Estimation and Compression of Facial Motion Parameters: Elements of a 3D Model-Based Video Coding System. *IJCV*, 50(2):111–125, 2002.

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

[23] B. Von Herzen and A. H. Barr. Accurate triangulations of deformed, intersecting surfaces. In *Computer Graphics (SIGGRAPH '87 Proceedings)*, volume 21, pages 103–110, July 1987.

[24] A. Witkin, M. Gleicher, and W. Welch. Interactive dynamics. In *Computer Graphics*, volume 24, pages 11–21, 1990.