International Journal of Computer Vision, 32, 111–142 (1999)
© 1999 Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.

General Object Reconstruction based on Simplex Meshes

HERVÉ DELINGETTE

Herve.Delingette@inria.fr

INRIA, B.P. 93, 2004 route des Lucioles,
06902 SOPHIA ANTIPOLIS Cedex, FRANCE

Received March 19, 1997; Revised December 7, 1998

Editors: ??

Abstract. In this paper, we propose a general tridimensional reconstruction algorithm of range and volumetric images, based on deformable simplex meshes. Simplex meshes are topologically dual of triangulations and have the advantage of permitting smooth deformations in a simple and efficient manner. Our reconstruction algorithm can handle surfaces without any restriction on their shape or topology. The different tasks performed during the reconstruction include the segmentation of given objects in the scene, the extrapolation of missing data, and the control of smoothness, density, and geometric quality of the reconstructed meshes. The reconstruction takes place in two stages. First, the initialization stage creates a simplex mesh in the vicinity of the data model either manually or using an automatic procedure. Then, after a few iterations, the mesh topology can be modified by creating holes or by increasing its genus. Finally, an iterative refinement algorithm decreases the distance of the mesh from the data while preserving high geometric and topological quality. Several reconstruction examples are provided with quantitative and qualitative results.

Keywords: Image Segmentation, Deformable Models, 3D Reconstruction, Medical Imaging, Range Images

1. Introduction

1.1. General Object Reconstruction

From laser range finders to volumetric medical imagery, the development of three-dimensional acquisition devices have stressed the need for general shape reconstruction techniques. Object reconstruction is a task that consists in building a geometric model from a tridimensional dataset obtained from a scanning device. Such tridimensional datasets may vary substantially in terms of accuracy, resolution, or data structure. In all cases, the reconstruction task should create a geometric model corresponding to the real object. The nature of this reconstructed model is dependent on the type of high level task performed a posteriori (see figure 1), such as visualization, object recognition, or scientific computing (mechanical or flow analysis).

In this paper, we address the problem of general object reconstruction as opposed to sensor-dependent reconstruction techniques. Ideally, a general reconstruction system should handle volumetric images as well as tridimensional range data, with varying noise level and resolution. Furthermore, it should reconstruct smooth objects as well as polyhedral shapes with a control over the mesh density and the closeness of fit.

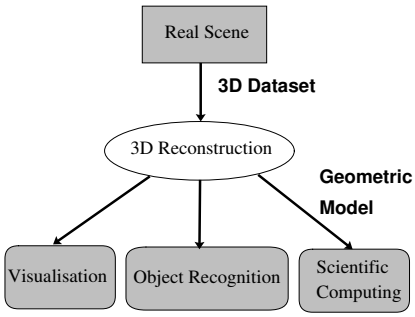


Fig. 1. The general reconstruction scheme

The figure recovery of objects from tridimensional datasets requires several tasks including:

Segmentation: the scene described in the tridimensional dataset, is usually made-up of several objects lying close to each other. The segmentation task consists in isolating the object points from data points corresponding to neighboring objects or outliers.

Filtering: real tridimensional datasets always include some amount of noise. Therefore, it is often necessary to smooth the reconstructed model in order to reduce the effect of noise without removing salient features such as sharp edges.

Extrapolation of missing data in many cases, 3D datasets do not entirely describe a given object. Therefore, it is necessary to handle those missing data points by extrapolating the surface in the most "intuitive" manner.

Mesh Density Control: when dense datasets are provided, it is often required, especially for visualization purposes, to greatly reduce the amount of information stored in the geometric model. On the contrary, when sparse data is provided, it is necessary to increase the mesh density by refining the geometric model.

Mesh Quality Control: many scientific computation algorithms (continuum mechanics, flow analysis,...) require the definition of finite elements on meshes of high geometric quality. The quality of a mesh may be defined in several terms [2]. For instance, on triangulated models, the geometric quality may be measured using the minimum, median or average angle of triangles.

Few existing algorithms address the "general reconstruction problem". For instance, isosurfacing [20] is a commonly used technique for reconstructing surfaces from volumetric or range images [10]. However, because it does not handle missing and noisy data, it cannot be considered as a general reconstruction technique. Similarly, Delaunay triangulation[1, 3] is widely used for serial slice reconstruction, or reconstruction from unorganized points but it cannot extrapolate missing data points.

Deformable modeling is well-suited for general object reconstruction because it makes few assumptions about the shape to recover and it can deal with missing and noisy data. There are several existing frameworks for deformable models, but a common approach consists of formalizing the deformation as a variational problem involving an internal energy ensuring the geometric continuity of the model, and an external energy controlling the closeness of fit. Many researchers have proposed reconstruction systems based on deformable models [7, 13, 40, 23, 6] (for a survey of deformable models in medical image analysis see [25]). However, few systems address the problem of general surface reconstruction, including the five tasks previously listed.

1.2. Deformable Models

A key issue for a general reconstruction system is related to the choice of a "good" surface representation. More precisely, such representations should be well-suited for all surfaces independent of their geometry and topology. In this section, we review the existing geometric representations of deformable models and introduce our original surface representation.

1.3. The Parameterization Problem

Most reconstruction systems are based on parametric representations, such as splines or finite elements. A parametric representation provides a continuous transformation between a parameter space Ω embedded in the Euclidean plane $\mathbb{R}^2$ and a tridimensional surface. The existence of a continuous representation enables the definition of geometric quantities, such as a normal vector or curvature information, everywhere on the surface

model. This is often required for high level tasks such as CAD design.

However, parametric representations suffer from two problems. The first problem is related to the representation of complex shapes. Because parameterizing a shape is equivalent to mapping a subset of the Euclidean plane onto that shape, problems occur when the object is not of planar, cylindrical, or toroidal topology. For spherical shapes for instance, at least one degenerate point or pole is created by the mapping of a plane onto a sphere. This point leads to many problems in the surface deformation because normal vector and curvature cannot be computed in a stable manner at that vertex. A lot of work has been done in the field of computer aided geometric design to overcome these topological problems. A popular approach consists of sewing several parametric patches and ensuring that proper geometric continuity is realized between the patches. For instance Hoppe and Eck [15] use the construction scheme of Peters [29] to build G^1 continuous surfaces of arbitrary topology while other reconstruction systems such as Loop and De Rose [19] rely on different patch corners such as Sabin nets. An important constraint in order to represent efficiently general deformable models with spline patches is that the G^1 continuity equation across patches must be linear in the control points. Those continuity constraints across patches usually entail the addition of a large number of control points and furthermore tend to break the homogeneity of the mesh [18].

The second problem lies in the nature of the parameterization which greatly influences the deformation scheme. The influence of the parameterization originates from the nature of the fairness functionals that are usually defined in terms of parametric-dependent quantities such as partial derivatives. This is the case, for instance, of the widely used thin plate functional, based on first order derivatives. When the deformation is constrained by parametric-dependent functionals, the reconstructed geometric model is dependent on the nature of the initial parameterization.

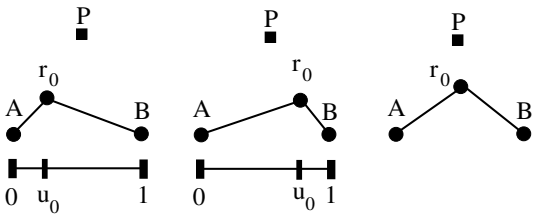


Fig. 2. The parameterization problem : (left) The curve minimizing the approximation functional of equation 1 with $u_0 = 0.2$ and $\lambda = 2.0$; (center) Same as (left) with $u_0 = 0.8$ (right) The curve minimizing the intrinsic approximation functional of equation 2 with $\lambda = 2.0$.

In figure 2, we demonstrate the influence of parameterization in a simple case where we are minimizing the following criterion:

$$E(\mathbf{r}) = \int_0^1 \| \frac{d\mathbf{r}}{du} \|^2 du + \lambda \| P - \mathbf{r}(u_0) \|^2 \quad (1)$$

The choice of the parameter u_0 of the attachment point $\mathbf{r}(u_0)$ has a drastic effect on the resulting approximation, better visual results being obtained when the initial parameterization is close to arc length. On the other hand, when the smoothness functional is intrinsic, i.e. independent of parameterization, the solution of the approximation problem is no longer dependent of the choice of u_0. In figure 2(right), we use the length of the curve as the fairness functional of the approximation problem:

$$E_{\text{intrinsic}}(\mathbf{r}) = \int_0^1 \| \frac{d\mathbf{r}}{du} \| du + \lambda \| P - \mathbf{r}(u_0) \|^2 \quad (2)$$

Details of these two approximation problems are provided in appendix A.1. This example reveals the trade-off in choosing between parameter-dependent and parameter-independent smoothness functionals. Parameter dependent functionals are easier to handle numerically, but the quality of the approximation depends on the metric distortion of its parameterization. On the other hand, intrinsic functionals guarantee a high level of smoothness but lead to more complex nonlinear problems.

In most research papers, little attention is paid to the errors introduced by metric distortions between the parameter space and the surface in ap-

proximation problems. This is mainly because they limit themselves to objects with simple topology (planar or cylindrical). However, Eck and Hoppe [15] have addressed this reparameterization problem with the use of harmonic maps. Brechbuhler [5] reparameterizes closed surfaces in order to perform efficient object recognition.

As a conclusion, building parametric deformable models of arbitrary topology is a difficult task due to the constraints of having G^1 continuity across patches as well as keeping a parameterization with little distortion. It is not well suited for the task of general surface reconstruction due to the computational expense of performing topological changes such as creating holes or local refinement.

1.4. Non Parametric Deformable Models

To overcome the topological restriction of parametric deformable models, researchers have developed alternate representations. For instance, implicit deformable models, where the surface model is defined through an implicit function $f(x, y, z) = 0$, have the advantage of not restricting the topology, connectivity, and the geometry of the surface. In most cases, the function is defined over a regularly tessellated domain. For instance, Mc Inerney and Terzopoulos [24] defined topologically adaptable snakes through reparameterization on regular simplicial domain. Maladi and Vemuri [21] propose to deform a two dimensional and tridimensional implicit model by propagating fronts based on the formulation of Osher and Sethian. Muraki [27] and Gascuel et al. [38] propose implicit models based on radial functions to fit range or medical data. Taubin et al. [37] fit algebraic curves and surfaces of high degree on range data. Curless and Levoy [10] use distance maps to build a single model from multiple range images. If implicit model recovery has many advantages in terms of topological flexibility, it leads to a computationally expensive scheme since it deforms the embedding space rather than the manifold itself. Furthermore, implicit models cannot represent surfaces with holes such as surfaces of planar topology.

Unstructured meshes, where no underlying parameterization is defined, have been used as deformable surface representation as well. The problem with using unstructured meshes lies in the definition of a stable and meaningful internal energy, or internal force. Vasilescu and Terzopoulos [39] use non-linear springs and masses on triangulated meshes for the reconstruction of range images. Even if this representation allows for local refinement and the detection of discontinuities, the numerical stability of spring models with non-zero rest length, is very sensitive to the mesh topology. In particular, if not enough springs are attached a vertex, the system becomes under-constrained and several rest shapes are possible. On the contrary, if too many springs are attached a vertex, the system is over-constrained and the bending of the spring model is very limited.

Other methods for regularizing a triangulation have been proposed. A traditional method for fairing triangulations is to apply Laplacian smoothing where each vertex is moved towards the center of its neighbors. Taubin [36] reduces the shrinkage of Gaussian smoothing by applying a low-pass filter. Recently, Boyer [4] proposed to minimize locally the area of the triangles adjacent to a vertex. Mallet [22] uses a smoothness energy on a triangulation defined as a quadratic form of the neighboring vertex positions. These three internal energies provide intuitive deformation, but are only stabilizers of degree 1 and therefore tend to flatten the curved parts of the triangulation. Welch and Witkin [41] use more sophisticated intrinsic functionals on triangulations by fitting a local coordinate frame at each vertex, and then minimizing the functional locally. In fact, this method proposes new finite differences expressions on an unstructured mesh. Despite the generality of this approach, it does not seem to be well-suited for our purpose because it involves non linear optimization and numerical instability may occur when performing intensive refinements or large deformations.

Subdivision surface techniques [16, 33] allow to describe smooth surfaces by iteratively subdividing an unstructured mesh. However, since the limit surface cannot be recovered explicitly but only approximately as a function of the control points, it often leads to computationally expensive schemes for approximating real data.

Oriented particles proposed by Szelinski [35] do not make any assumptions about the connectivity,

or topology of the reconstructed model. However, it is not clear how robust is this method against outliers and narrow objects.

1.5. Simplex Meshes as Deformable Models

We propose an original surface representation, called simplex meshes as the basis of our reconstruction system. Simplex meshes can represent surfaces of all topologies, just as triangulation meshes. Furthermore, the geometry of simplex meshes enables to define at each vertex, discrete geometric quantities such as mean curvature or normal vectors. We have defined regularizing forces on simplex meshes allowing a high order of geometric continuity (C^2 continuity) in a simple and efficient manner. In addition, local and global topological operators are defined to refine a simplex mesh or change its genus, in a simple fashion. Because simplex meshes are not parametric models, it is not necessary to update a rigidity matrix or the parameterization mapping when performing topological transformations.

In this paper, we present a shape recovery system based on this surface representation. In particular, we discuss the problem of initializing a simplex mesh, often considered as the weak point of deformable models. Also, we propose a semi-automatic algorithm for changing the mesh topology by creating holes or increasing the surface genus. Finally, we describe a refinement algorithm based on the minimization of a geometric criterion based on the distance to the data or the local curvature.

The outline of the paper is as follows. In section 2, we briefly introduce the simplex mesh both in terms of topology as well as geometry. In section 3, we introduce the internal and external forces applied on these meshes in presence of range data or volumetric images. In section 4, we describe the various components of our reconstruction system including the initialization and topology control. In section 6, we present several examples of shape recovery and we conclude in section 7.

2. Simplex Meshes

The properties and definitions of simplex meshes are only summarized in this section. Some complementary definitions have been added as appendices and additional information can be found in [11]. Some preliminary results have been published in [12].

2.1. Definition of Simplex Meshes

The definitions of simplex meshes and triangulations are closely related. More precisely, their underlying graphs are dual of each other. Another important property of simplex meshes is their constant vertex connectivity. In this section, we only introduce the topological properties of simplex meshes. First, we give a general definition of k-simplex meshes embedded in a Euclidean space $\mathbb{R}^d$ of dimension d. Then, we consider 1 and 2-simplex meshes of $\mathbb{R}^3$ as representations of contours and surfaces.

2.1.1. Definition of Cells and Simplex Meshes

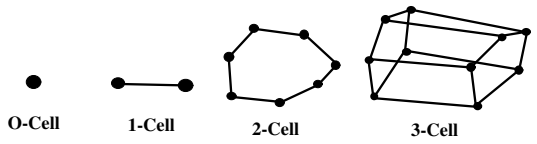


Fig. 3. Examples of p-cells $0 \leq p \leq 3$.

We define a k-simplex as a union of p-cells, $0 \leq p \leq k$. Since these cells are p-simplex meshes, the definition of a cell is recurrent:

Definition 1. We define a 0-cell of $\mathbb{R}^d$ as a point P of $\mathbb{R}^d$ and a 1-cell of $\mathbb{R}^d$ as an edge of $\mathbb{R}^d$, i.e. an unordered pair of distinct vertices (P, M). We recursively define a p-cell ($p \geq 2$) $\mathcal{C}$ of $\mathbb{R}^d$ as a union of $(p-1)$-cells such that:

1. Every vertex belonging to $\mathcal{C}$ belongs to p distinct $(p-1)$-cells.
2. The intersection of two $(p-1)$-cells is either empty or is a $(p-2)$-cell.

A 2-cell is therefore a set of edges that have one and only one vertex in common. It is therefore a closed polygonal line of $\mathbb{R}^d$. Examples of p-cells are shown in figure 3. 0-cells are called vertices, 1-cells edges and 2-cells faces.

A simplex mesh is simply defined as:

Definition 2. A k-simplex mesh $\mathcal{M}$ of $\mathbb{R}^d$ is a $(k+1)$-cell of $\mathbb{R}^d$.

A k-simplex mesh is therefore a union of k-cells that follow the properties of definition 1. Examples of 2-simplex meshes are shown in figure 4.

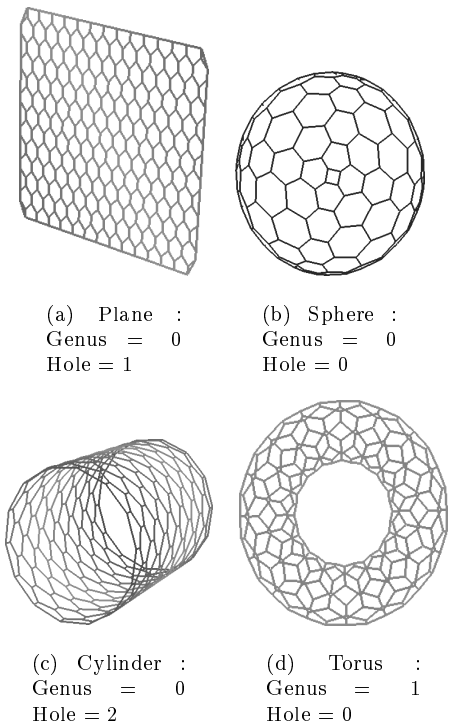


Fig. 4. Four generic 2-simplex meshes with different values of genus and number of holes.

Proposition 1. A k-simplex mesh is a $(k+1)$-connected mesh: each vertex has $(k+1)$ neighboring vertices.

The constant connectivity between vertices implies a simple relation between the number of vertices and the number of edges. Table 1 summarizes the connectivity between vertices, edges, faces, and cells of a k-simplex mesh. If a k-simplex mesh is $(k+1)$-connected, all $(k+1)$-connected meshes are not necessary simplex meshes. For instance, a 3-connected mesh where two faces intersect along two edges cannot be a 2-simplex mesh.

We write the set of n vertices of $\mathcal{M}$ as $V(\mathcal{M})$ and its connectivity function as $N(\mathcal{M})$. If

Table 1. The connectivity relations of a k-simplex mesh.

	Edges / Vertex	Faces / Vertex	Faces / Edge
k=1	2		
k=2	3	3	2

P_i is a vertex of a k-simplex mesh $\mathcal{M}$ then $(P_{N_0(i)}, P_{N_1(i)}, \ldots, P_{N_k(i)})$ are its $(k+1)$ neighbors.

2.1.2. Duality with Triangulations It is important to stress the dual nature between k-simplex meshes and k-triangulations. The k-triangulations, also called k-manifolds [32], are sets of p-simplices $(0 < p \leq k)$ that follow strict topological rules such as the Euler-Poincaré relation. The k-triangulations are actually a subset of more general sets of p-simplices, called k-simplicial complexes.

A k-triangulation is composed of p-simplices $(0 \leq p \leq k)$ which are the p-faces of the triangulation. 0-faces are the vertices, 1-faces the edges and 2-faces the triangles.

We can define a topological transformation that associates a k-simplex mesh to a k-triangulation. This transformation is pictured in figure 5 and considers differently the vertices and edges located at the boundary of the triangulation from those located "inside". Basically, this duality transformation associates a p-face of a k-triangulation with a $(k-p)$-cell of a k-simplex mesh.

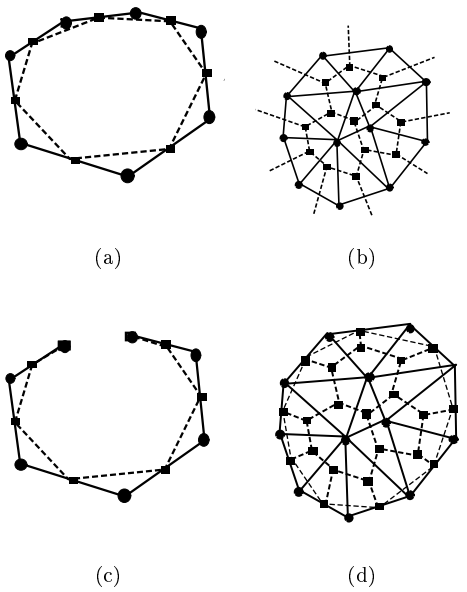


Fig. 5. Duality between k-triangulations ($k = 1, 2$) drawn with solid lines circles, and k-simplex meshes drawn with dashed lines and squares. The first two figures correspond to triangulations without boundaries whereas the last two have boundaries.

Table 2 summarizes the transformation between a p-face of a k-triangulation and a $(k-p)$-cell of a k-simplex mesh. For cells or faces belonging to a triangulation boundary or a simplex mesh boundary, the dual transformation applies differently. Vertices, edges, and triangles belonging to a k-triangulation boundary are associated with two cells of a k-simplex mesh: one $(k-p)$-cell and one $(k-p-1)$-cell $(0 \leq p < k)$ (see table 3).

Table 2. Duality between a k-triangulation and a k-simplex mesh for internal faces.

	$1-\text{Tr} \Longleftrightarrow 1-\text{SM}$	$2-\text{Tr} \Longleftrightarrow 2-\text{SM}$
$p = 0$	Vertex $\Longleftrightarrow$ Edge	Vertex $\Longleftrightarrow$ Face
$p = 1$	Edge $\Longleftrightarrow$ Vertex	Edge $\Longleftrightarrow$ Edge
$p = 2$		Triangle $\Longleftrightarrow$ Vertex

In a triangulation, there is a real notion of boundary since a triangle may not be surrounded by three triangles. In a simplex mesh, a hole is simply an "empty" cell, since each vertex is surrounded by $(k+1)$ k-cells.

There exists other duality transformations that have been defined on k-triangulations. The most commonly studied has been the duality between triangulations and cellular complexes through the duality between Delaunay triangulation and Voronoï diagrams [31]. Voronoï diagrams are cellular complexes and the duality relation with Delaunay triangulation is geometric because it depends on the position of its vertices. On the contrary, the duality between triangulations and simplex meshes is purely topological since there are no geometric bijections between simplex meshes and triangulations.

The inexistence of geometric duality can be proven by considering the relative number of degrees of freedom of simplex meshes and triangulations. The geometry of a non-degenerate k-simplex mesh or a non-degenerate k-triangulation is determined by the set of coordinates of their vertices. However, for $k > 1$ the number of vertices V_{sm} of a k-simplex mesh is different from the number of vertices V_{tr} of a k-triangulation. For $k = 2$ for instance, for a triangulation without

Table 3. Duality between a k-triangulation and a k-simplex mesh for boundary faces.

	1-Tr $\Longrightarrow$ $1-$SM	2-Tr $\Longrightarrow$ 2-SM	1-SM $\Longrightarrow$ $1-$Tr	2-SM $\Longrightarrow$ 2-Tr
$p = 0$	Vertex $\Longrightarrow$ Edge	Vertex $\Longrightarrow$ Face	Vertex $\Longrightarrow$ (nil)	Vertex $\Longrightarrow$ (nil)
	Vertex $\Longrightarrow$ Vertex	Vertex $\Longrightarrow$ Edge		
$p = 1$		Edge $\Longrightarrow$ Edge		Edge $\Longrightarrow$ (nil)
		Edge $\Longrightarrow$ Vertex		

holes having genus number g, the Euler relation gives:

$$V_{tr} - \frac{V_{sm}}{2} = 2(1-g)$$

Therefore, we cannot build an geometric homeomorphism between triangulations and simplex meshes since the vectorial space representing their space of possible configurations have different dimensions. Only for 1-simplex meshes, is it possible to build a geometric dual 1-triangulation since they have the same number of vertices. Geometric equivalence with triangulations can only exist if we consider degenerate k-simplex meshes, for instance by restraining to convex polytopes (convex simplex meshes with planar faces).

The inexistence of a geometric transformation between simplex meshes and triangulations implies that the geometric deformation of a simplex mesh is not equivalent to the geometric deformation of a triangulation. This is the reason why simplex meshes are a surface representation distinct from triangulations.

2.1.3. Contours Contours defined on 2-simplex meshes, are 1-simplex meshes, i.e. closed polygonal curves. They are simply defined as a set of neighboring vertices such that a contour vertex has two and only two neighbors belonging to the contour (see figure 6). Contours can be defined around any simplex-mesh faces especially faces corresponding to a hole.

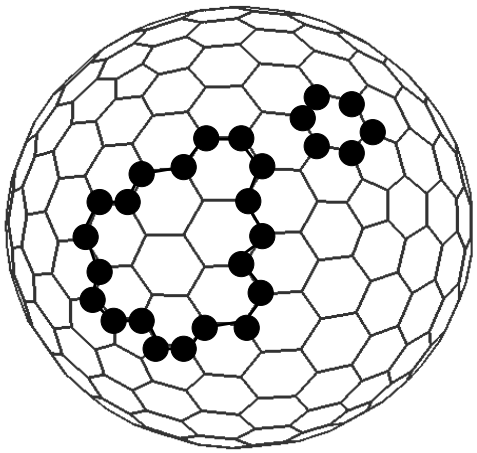


Fig. 6. Two contours defined on a 2-simplex mesh. The rightmost contour surrounds a face of the mesh.

Contours are considered as deformable models moving independently from the mesh. The surface mesh is thus attached to contour vertices which set the boundary conditions for the mesh deformation. The definition of contours allows a greater control of a simplex mesh shape.

2.1.4. Mesh Transformation Simplex meshes as triangulations are unstructured meshes and therefore can be locally refined or decimated. In addition, simplex meshes can be cut along contours and surface handles can be created.

We define four basic topological operators acting on a simplex mesh, $T_1^2, T_2^2, T_3^2, T_4^2$, described in figure 7. The first two operators, are Eulerian since they do not modify the mesh genus. On the other hand, T_3^2 and T_4^2 are meta-operators because they can break a mesh into two pieces, or change its genus (number of surface handles). All topological transformations can be decomposed into a set of those four operators.

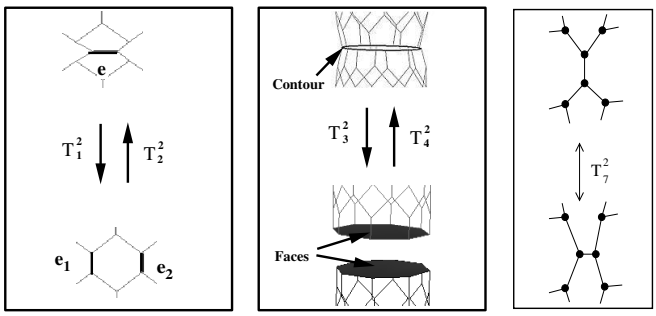


Fig. 7. (a) The two Eulerian operators T_1^2, T_2^2 defined on 2-simplex meshes; (b) The two meta-operators T_3^2 and T_4^2; (c) The edge swap operator T_7^2.

2.2. *Geometry of simplex meshes*

In this section, we present the main geometric relations existing in a 2-simplex mesh of $\mathbb{R}^3$. Similar results exist for 1-simplex meshes of $\mathbb{R}^3$ (i.e. tridimensional contours) and are described in appendix A.3. The main result consists of a simple equation (equation A1 and equation 7) giving the position of a vertex relatively to its neighbors and some geometric quantities: the simplex angle and the metric parameters.

On a tridimensional 2-simplex mesh $\mathcal{M} \in \mathbb{R}^3$, a vertex P_i is surrounded by three vertices $(P_{N_1(i)}, P_{N_2(i)}, P_{N_3(i)})$. These three points define the tangent plane at P_i with its normal vector $\mathbf{n}_i$:

$$\mathbf{n}_i = \frac{P_{N_1(i)} \wedge P_{N_2(i)} + P_{N_2(i)} \wedge P_{N_3(i)} + P_{N_3(i)} \wedge P_{N_1(i)}}{\| P_{N_1(i)} \wedge P_{N_2(i)} + P_{N_2(i)} \wedge P_{N_3(i)} + P_{N_3(i)} \wedge P_{N_1(i)} \|}$$

We introduce the circle S_1 of center C_i and radius r_i circumscribed to the triangle $(P_{N_1(i)}, P_{N_2(i)}, P_{N_3(i)})$. We introduce also the sphere S_2 of center O_i and radius R_i, circumscribed to the four vertices $(P_i, P_{N_1(i)}, P_{N_2(i)}, P_{N_3(i)})$. The simplex angle $\varphi_i = \angle(P_i, P_{N_1(i)}, P_{N_2(i)}, P_{N_3(i)})$ at P_i is defined by the two equations:

$$\begin{gathered} \varphi_i \in [-\pi, \pi] : \\ \sin(\varphi_i) = \frac{r_i}{R_i} \operatorname{sign}((P_{N_1(i)} - P_i) \cdot \mathbf{n}_i) \\ \cos(\varphi_i) = \frac{\|C_i - O_i\|}{R_i} \operatorname{sign}((C_i - O_i) \cdot \mathbf{n}_i) \end{gathered} \tag{3}$$

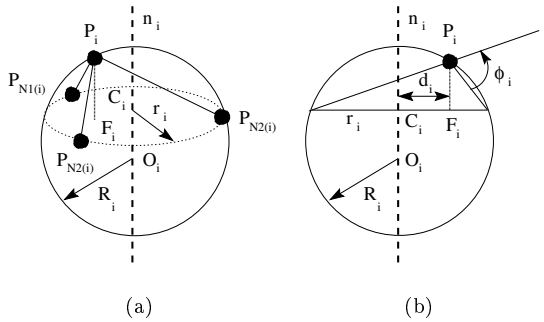


Fig. 8. (a) The circumscribed sphere S_2 of radius R_i and the circumscribed circle S_1 of radius r_i. (b) Projection of figure (a) onto the plane (O_i, C_i, P_i). The simplex angle can be interpreted as an angle of planar geometry.

The simplex angle has many original properties that are developed in [11]. In particular, there is a simple relationship between the simplex angle φ_i and the curvature $H_i = \frac{1}{R_i}$, also called the mean curvature at vertex P_i:

$$H_i = \frac{\sin \varphi_i}{r_i} \tag{4}$$

From a vertex P_i, we introduce the orthogonal projection F_i of P_i onto the neighboring triangle $(P_{N_1(i)}, P_{N_2(i)}, P_{N_3(i)})$. The metric parameters at a vertex P_i are the barycentric coordinates of F_i with respect to the triangle $(P_{N_1(i)}, P_{N_2(i)}, P_{N_3(i)})$:

$$F_i = \epsilon_{1i} P_{N_1(i)} + \epsilon_{2i} P_{N_2(i)} + \epsilon_{3i} P_{N_3(i)} \tag{5}$$

$$\epsilon_{1i} + \epsilon_{2i} + \epsilon_{3i} = 1 \tag{6}$$

The simplex angle with two metric parameters (since the three metric parameters are linked by equation 6) represent the position of P_i with respect to its three neighbors:

$$P_i = \epsilon_{1i} P_{N_1(i)} + \epsilon_{2i} P_{N_2(i)} + \epsilon_{3i} P_{N_3(i)} + L(r_i, d_i, \varphi_i)\mathbf{n}_i \quad (7)$$

where

- r_i is the radius of the circumscribed circle at the triangle $(P_{N_1(i)}, P_{N_2(i)}, P_{N_3(i)})$.
- d_i is the distance between F_i and the center C_i of the circumscribed circle.
- $L(r_i, d_i, \varphi_i)$ is defined as:

$$L(r_i, d_i, \varphi_i) = \frac{(r_i^2 - d_i^2)\tan(\varphi_i)}{\epsilon\sqrt{r_i^2 + (r_i^2 - d_i^2)\tan^2(\varphi_i)} + r_i} \quad (8)$$

with:

$$\epsilon = 1 \quad if \quad |\varphi_i| < \pi/2$$
$$\epsilon = -1 \quad if \quad |\varphi_i| > \pi/2$$

The simplex angle and the metric parameters define the local shape around a given vertex. The simplex angle controls the local mean curvature, i.e, the height of a vertex with respect to the tangent plane. The metric parameters control the vertex position in the tangent plane with respect to its three neighbors. In another words, the metric parameters change the local "parameterization" whereas the simplex angle changes the extrinsic curvature of the surface.

3. Deformable Simplex Meshes

3.1. Law of Motion

In this section, we describe the law of motion for simplex meshes. Unlike parametric deformable models, the deformation of simplex meshes is not based on the notion of partial derivatives, but on the relative position of a vertex with respect of its neighbors, i.e. in terms of the simplex angle and the metric parameters.

As in most deformable model schemes, all vertices of a simplex mesh are considered as a physical mass submitted to a Newtonian law of motion including internal and external forces:

$$m\frac{d^2P_i}{dt^2} = -\gamma\frac{dP_i}{dt} + \mathbf{F}_{int} + \mathbf{F}_{ext} \quad (9)$$

where m is the vertex mass and γ is the damping factor. $\mathbf{F}_{int}$ is the internal force assuring some level of geometric continuity. $\mathbf{F}_{ext}$ is the external force constraining the distance between the mesh and some tridimensional dataset.

We discretized the time t in order to compute the evolution of the simplex mesh under the law of motion (9). Using central finite differences with an explicit scheme, the law of motion is discretized as:

$$P_i^{t+1} = P_i^t + (1-\gamma)(P_i^t - P_i^{t-1}) + \alpha_i\mathbf{F}_{int} + \beta_i\mathbf{F}_{ext} \quad (10)$$

Both forces $\mathbf{F}_{int}$ and $\mathbf{F}_{ext}$ are computed at time t. This explicit scheme is conditionally stable and therefore the two parameters α_i and β_i must belong to a given interval to guarantee a stable scheme. In equation 10, the internal and external forces have the dimension of a displacement.

3.2. Internal Force Computation

We propose an original formulation of internal forces based on the geometry of simplex meshes. The large extent of shape control is the main motivation for using simplex meshes rather than triangulations as a deformable surface representation (see section 5).

The proposed internal force expressions are both geometrically significant and computationally efficient. However, they do not derive from the minimization of a global functional. Indeed, the geometric parameters (mean curvature, simplex angle, and metric parameters) at a vertex are related to the vertex position with a complex non-linear relationship. Therefore, the minimization of a global functional expressed in term of the geometric parameters, would lead to unnecessarily complex force expressions. We have chosen to use simplified regularizing force formulae at the expense of not having a global functional for guiding the minimization.

The internal force applied on the simplex mesh vertices can be decomposed into a tangential force $\mathbf{F}_{\mathrm{T}angent}$ and a normal force $\mathbf{F}_{\mathrm{N}ormal}$.

$$\mathbf{F}_{int} = \mathbf{F}_{\mathrm{T}angent} + \mathbf{F}_{\mathrm{N}ormal}$$

3.2.1. Tangential Internal Force The goal of the tangential force is to control the vertex position with respect to its three neighbors in the tangent plane. In general, it is preferable to have vertices uniformly spread over the surface of a simplex mesh as in figure 9 (a). In some other cases, it is advantageous to have a concentration of vertices in places with high curvature as in figure 9 (b).

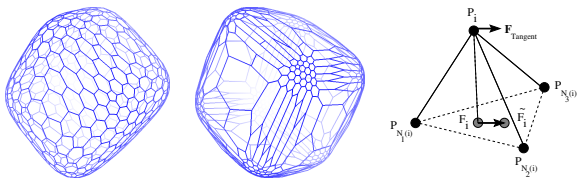


Fig. 9. (a) Deformable simplex mesh with reference metric parameters equal to 1/3; (b) Deformable simplex after adaptation of its reference metric parameter in order to increase the resolution in places with high curvature; (c) Definition of the tangential force as $(\tilde{F}_i - F_i)$.

At each vertex P_i, we have defined in section 2.2 the three metric parameters $(\epsilon_{1i}, \epsilon_{2i}, \epsilon_{3i})$. In addition, we define three reference metric parameters $(\tilde{\epsilon}_{1i}, \tilde{\epsilon}_{2i}, \tilde{\epsilon}_{3i})$ corresponding to the prescribed value of the metric parameters. These two sets of metric parameters correspond to the barycentric coordinates of the two points F_i and $\tilde{F}_i$. The tangential force $\mathbf{F}_{\mathrm{T}angent}$ acting on a vertex P_i is simply a spring force between F_i and $\tilde{F}_i$:

$$\mathbf{F}_{\mathrm{T}angent} = \tilde{F}_i - F_i = (\tilde{\epsilon}_{1i} - \epsilon_{1i})P_{N_1(i)} + (\tilde{\epsilon}_{2i} - \epsilon_{2i})P_{N_2(i)} + (\tilde{\epsilon}_{3i} - \epsilon_{3i})P_{N_3(i)}$$

To obtain uniformly spread vertices, the reference metric parameters should all be equal $\tilde{\epsilon}_{1i} = \tilde{\epsilon}_{2i} = \tilde{\epsilon}_{3i} = 1/3$. Otherwise, any strictly positive value of the reference metric parameters is possible for a stable deformation.

3.2.2. Normal Internal Force The force acting in the normal direction constrains the mean curvature of the surface through the simplex angle (see equation 4). The general expression of $\mathbf{F}_{\mathrm{N}ormal}$ is governed by the reference simplex angle $\tilde{\varphi}_i$:

$$\mathbf{F}_{\mathrm{N}ormal} = (L(r_i, d_i, \tilde{\varphi}_i) - L(r_i, d_i, \varphi_i))\,\mathbf{n}_i$$

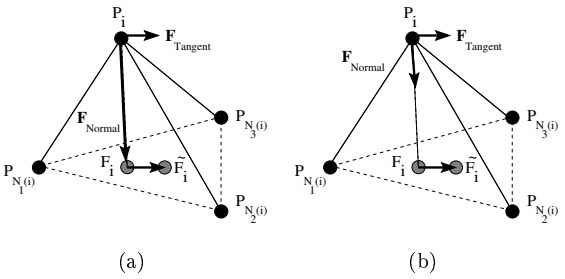


Fig. 10. (a) Geometric interpretation of the C^1 regularizing force: in this case the internal force can be written as $\mathbf{F}_{int} = \tilde{F}_i - P_i$; (b) Geometric interpretation of the C^2 regularizing force: the normal component is related to the variation of mean curvature in a neighborhood of vertex P_i.

The expression of $\tilde{\varphi}_i$ depends on the level of geometric continuity that should be enforced:

Position Continuity or C^0 constraint: We set $\tilde{\varphi}_i = \varphi_i$. In this case, $\mathbf{F}_{\mathrm{N}ormal} = \mathbf{0}$ and the surface can freely bend around vertex P_i. This is useful to represent surface orientation discontinuities.

Surface Orientation Continuity or C^1 constraint: We choose $\tilde{\varphi}_i = 0$, which corresponds to moving each vertex towards its center of curvature. The total internal force is then equivalent to the mean curvature motion corresponding to the minimization of the total surface area. By using equation (7), the internal force can be written as: $\mathbf{F}_{int} = (\epsilon_{1i}P_{N_1(i)} + \epsilon_{2i}P_{N_2(i)} + \epsilon_{3i}P_{N_3(i)} - P_i)$. When metric parameters are equal to 1/3, the vertex P_i is attracted towards the center of its neighbors and the internal force is equivalent to Laplacian Smoothing.

Shape Constraint: To constrain the local curvature of a vertex, we set $\tilde{\varphi}_i = \varphi_i^0$, where $= \varphi_i^0$ is a constant. When this constraint is applied on every vertex of a simplex mesh, this entails a global shape constraint up to a rotation, translation, and scale transformation.

Simplex Angle Continuity or C^2 constraint : $\tilde{\varphi}_i$ is defined as an average of the simplex angles at neighboring vertices:

$$\tilde{\varphi}_i = \sum_{j \in Q^{s_i}(P_i)} \varphi_j \qquad (11)$$

The neighborhood $Q^{s_i}(P_i)$ is recursively defined as the union of $Q^{s_i-1}(P_i)$ with the neighbors of $Q^{s_i-1}(P_i)$ (we set $Q^0(P_i) = \{P_i\}$ as shown in figure 11). The neighborhood size s_i corresponds intuitively to the notion of rigidity, or deformation scale. In a highly rigid model, the model is smoothed over a large scale, and therefore an external constraint would entail a small deflection but over a large extent of the surface. The rigidity also has an effect on the dynamics of the deformable model. This is because the scale of deformation influences the speed of constraint propagation on the surface. A flexible mesh with a small rigidity needs more iterations to reach its stable position than a mesh with high rigidity.

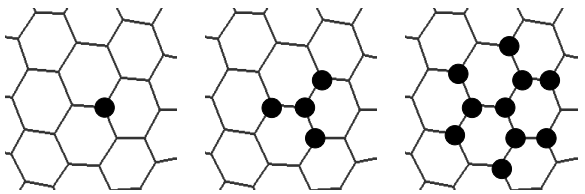


Fig. 11. Description of a neighborhood $Q^{s_i}(P_i)$ around a vertex P_i with $s_i = 0$ (see (a)) , $s_i = 1$ (see (b)) and $s_i = 2$ (see (c))

In practice, we are using the C^2 smoothness constraint for shape recovery. Indeed, with the C^0 smoothness constraint, the simplex mesh is equivalent to spring-mass models and therefore only guarantees that a vertex stays close to its three neighbors. The C^1 smoothness constraint entails a continuity of surface normals but shrinks the surface towards its center of curvature. This shrinking effect is a well known limitation of linear filtering ($\mathbf{F}_{int}$ is then a linear combination of the vertex positions) and results in a bias estimate of the surface reconstruction. Indeed, the shrinking effect implies that the mesh has a privileged direction of deformation. To limit this bias, several algorithms has been proposed. For instance, a balloon force [8] has been added that artificially pushes each vertex outwards. Another approach consists in creating an initial mesh that encloses the object to reconstruct. Instead, we propose to use a C^2 smoothness constraint for extrapolating missing data since it performs a good visual continuity of the mesh without any shrinkage.

The maximum value of α_i is related to the stability criterion of the differential equation (10). When applying the C^0 or C^1 constraint, equation 10 can be seen as a straightforward Jacobi relaxation scheme and the maximum value of α_i is 1.0. The expression of the C^2 internal force is non-linear and we have found experimentally that α_i value below $1/2$, ensures a stable iterative scheme.

3.2.3. Contour Deformation Contours defined on simplex meshes are deformable as well, and vertices belonging to those contours follow the same law of motion as in equation (9), but with a different expression of the internal force. The internal forces of tridimensional contours are described in appendix A.4 and are closely related to the formulae of section 3.2.1 and 3.2.2 ranging from position continuity to curvature continuity.

Because contour vertices are deformed independently of mesh vertices, they act as boundary conditions for the surface mesh. There are two types of surface-contour constraints:

Simplex Angle Condition: The simplex angles at contour vertices are set to a given value.

Tangent Condition: The angle between the mesh normal and the contour normal measured around the contour tangent vector, intuitively corresponds to the angle between the surface and the contour.

3.2.4. Examples The definitions of C^0, C^1, C^2 and shape constraints on simplex mesh vertices and contour vertices, provide a powerful framework for reconstructing tridimensional objects. The variational modeling ability of simplex meshes proved to be a key feature for reconstructing surfaces of complex geometry and topology.

In figure 12, we show a smooth junction between 4 four cylinders with C^1 and C^2 smoothness constraints. The junction has been created from a

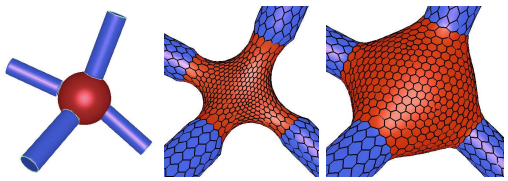


Fig. 12. Example of smoothness constraints applied to simplex meshes. We constrain a surface with four holes (left) to smoothly connect 4 cylinders with surface orientation continuity (center) and simplex angle continuity (right).

sphere with four holes. The holes have been manually connected with the extremities of each cylinder.

In figure 13, we show three examples of variational modeling based on simplex meshes and contours. The first figure is an example of a minimal surface (zero mean curvature everywhere) based on simplex meshes. In figures 13 (b) and (c), we demonstrate the different boundary conditions between simplex meshes and contours.

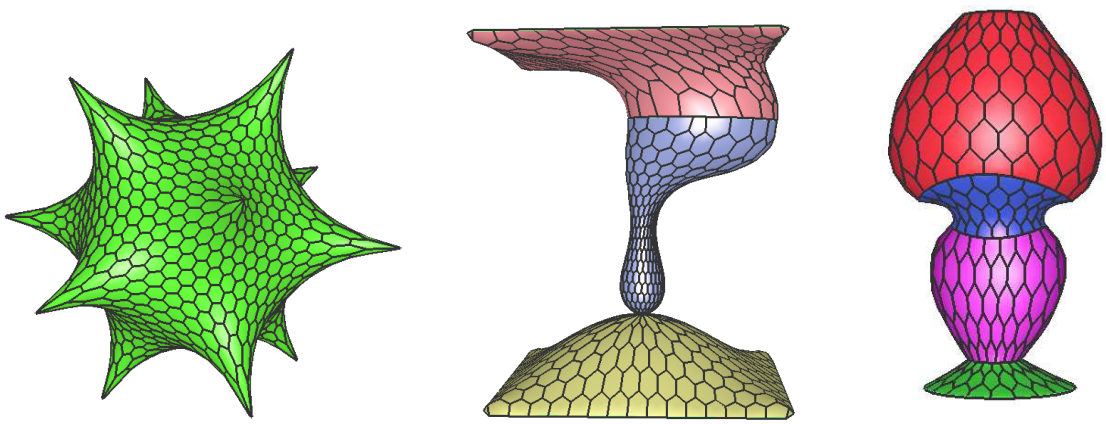
3.3. External Force Computation

The expression of the external force $\mathbf{F}_{ext}$ is dependent on the nature of the dataset and is given in sections 3.3.1 and 3.3.2 for range data and volumetric images. In all cases, the external force is directed along the normal direction $\mathbf{n}_i$ at the vertex P_i where the force is applied. The collinearity of $\mathbf{F}_{ext}$ with $\mathbf{n}_i$ is important for two reasons.

First, it entails smooth deformations with attraction forces varying smoothly along the mesh. Since we use an iterative process, it is important to guarantee a stable and smooth behavior over time. Indeed, tangential displacements could create vertices with negative metric parameters or even a self-intersecting mesh.

Second, the normal direction of the external force ensures the resulting mesh shape of the mesh will be smooth even in the presence of sparse data as shown in figure 14.

The complexity of the computation of $\mathbf{F}_{ext}$ is linear in the number of mesh vertices, unlike Metaxas [17] and Terzopoulos [23] who compute the external force of all vertices with a complexity linear with the number of data points since they compute the closest mesh vertex from each



(a) (b) (c)

Fig. 13. (a) a simplex mesh under C^1 continuity constraint (with null mean curvature everywhere) having the symmetry of a dodecahedron; (b) a simplex mesh under C^2 continuity constraint and four contours; the surface-contour constraints of the two intermediate contours correspond to curvature constraint (zero mean curvature) and tangent constraint; (c) a vase created from a cylinder and five contours;

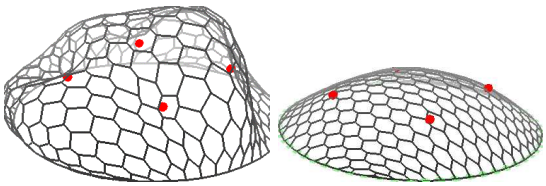


Fig. 14. (left) A simplex mesh is fit on sparse data represented by dark dots. The external force is not projected along the normal direction and tangential components of the force perturbates the smoothness constraint of the mesh. (right) The external force is directed along the normal direction resulting in a smooth shape.

data point. This method is not appropriate for general surface reconstruction since it makes the double assumption that all data points belong to the same object and that there are no outliers in the data. Furthermore, in presence of dense data, this method becomes too computationally expensive.

We propose an expression of the external force that deals with sparse as well as dense datasets and that can take into account outliers. We describe the computation of $\mathbf{F}_{ext}$ for range data and volumetric images.

3.3.1. External Force Computation on Range Data
The computation of the external force is dependent on the notion of closest points. For every vertex P_i of the mesh, we look for the closest data point $M_{Cl(i)}$ and the force $\mathbf{F}_{ext}$ is then computed as:

$$\mathbf{F}_{ext} = \beta_i G\left(\frac{(M_{Cl(i)} - P_i)\cdot \mathbf{n}_i}{D_{\text{ref}}}\right)\mathbf{n}_i \qquad (12)$$

where β_i is a weight parameter ($\beta \leq 1.0$), $\mathbf{n}_i$ is the normal vector at P_i, $G(x)$ is the derivative of

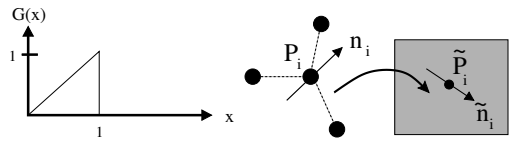


Fig. 15. (left) The function $G(x)$ corresponding to the derivative of an M-estimator; (right) Search for the closest point on a range image using the projection method; $\tilde{P}_i$ and $\tilde{\mathbf{n}}_i$ are the projection of P_i and $\mathbf{n}_i$ on the image plane.

an M-estimator function and D_{ref} is the cut-off distance.

The force $\mathbf{F}_{ext}$ is computed as the projection of the vector $M_{Cl(i)} - P_i$ in the normal direction. In order to have a robust estimation of the closest point, we use a cut-off distance D_{ref} with an M-estimator function $G(x)$. We have chosen a simple truncated linear function as the $G(x)$ function (see figure 15 (left)) which states that the force is null if $(M_{Cl(i)} - P_i)\cdot \mathbf{n}_i$ is greater than D_{ref}.

The cut-off distance D_{ref} is manually defined as a percentage of the overall size of the dataset, i.e. the radius of the sphere surrounding the dataset. During the first stage of the deformation, we choose a relatively large value of D_{ref} (up to 20% of the radius) in order to allow large deformations of the mesh. After few iterations, the value of D_{ref} is decreased (at most 8% of the radius) in order to limit the effect of outliers and to speed-up the search for the closest point.

The computation of $M_{Cl(i)}$ depends on the nature of the dataset and can be achieved with four algorithms:

1. Projection Method : On dense range images extracted from triangulation principles [34] or stereo-vision [14], the search for the closest point is theoretically of complexity $O(m^2)$ for an $m \times m$ image. However, when the calibration matrix is known, we propose a method in $O(1)$, that approximates the search for the closest point. Indeed, we restrict the search along a two-dimensional segment in the image, projection of the tridimensional line passing through P_i and directed along $\mathbf{n}_i$ (see figure 15 (right)). This segment is centered around the projection of P_i and is only a few pixels long, depending on the value of D_{ref} . Once the closest point along the segment has been found, we search for the closest point in a 5×5 window around that point.
2. Projection Method and Normal Orientation : We can further restrict the choice of the closest point by considering the normal orientation of the range data in addition to the Euclidean distance. We then check if the dot product between the normal $\mathbf{n}_i$ at the mesh and the normal at the range images is strictly positive.

3. KD-tree : When no calibration matrix is available, we compute the closest point with a kd-tree [31]. This data structure gives the closest point inside a sphere centered around P_i and of radius D_{ref}, in nearly constant time. Because of memory limitation, it is not always possible to store all data points inside a kd-tree. In the first stage of the deformation where D_{ref} is relatively large, we usually subsample the whole dataset in order to guarantee a fast computation of $\mathbf{F}_{ext}$. During the second stage, we discard all the data points that are far away from the mesh, and store all the remaining points in the kd-tree. The size of the kd-tree may still be large, but since the value D_{ref} is substantially lower, the computation time is still moderate.

3.3.2. External Force Computation on Volumetric Images On volumetric images, the reconstruction task usually consists in isolating regions of consistent intensity values. Therefore, gradient intensity is the main information on which the external force is based. As in [7] and [23], we combine both gradient intensity and edge information for the computation of $\mathbf{F}_{ext}$:

$$\mathbf{F}_{ext} = \mathbf{F}_{grad} + \mathbf{F}_{edge} \tag{13}$$

The gradient intensity is used for local deflection of the mesh towards the voxels of maximum variation of intensity. The edge information, on the other hand, corresponds to gradient maxima and entails larger deformations of the mesh. The edge image is a binary image extracted from the gradient intensity image through thresholding.

Previous work [23] has derived the computation of $\mathbf{F}_{grad}$ as the gradient of the potential field $\|\nabla I\|^2$, thus generalizing the approach of active contours. However, this formulation has the drawback of creating oscillations of the deformable model across the voxels with high gradient, due to the choice of the time step and the space discretization. Cohen [9] has proposed to normalize the gradient vector in order to limit those oscillations within a pixel. However, this has the drawback of limiting the range of influence of the gradient force to an arbitrary value.

We propose an expression of the gradient force that avoid all oscillations and that is not disturbed by local maxima of gradient intensity. The force $\mathbf{F}_{grad}$ at vertex P_i relies on the search in a neighborhood of P_i, for the voxel of maximum gradient intensity. If $\mathcal{V}$ is the voxel containing P_i, then we inspect all voxels in an $m \times m \times m$ window around $\mathcal{V}$ for the voxel center G_i, of highest gradient intensity (see figure 16 (a)). The force expression is then:

$$\mathbf{F}_{grad} = \beta_i^{grad}((G_i - P_i) \cdot \mathbf{n}_i)\mathbf{n}_i \tag{14}$$

where β_i^{grad} is a weighting parameter with $0 \leq \beta_i^{grad} \leq 1$. Since the force is proportional to the true deflection vector $G_i - P_i$ (and not the gradient vector), this force does not entail any oscillations of the mesh. The computation of this force is actually fast since we can pre-compute the voxel center G_i given the voxel $\mathcal{V}$ and a fixed window size.

The gradient force can be made more specific by incorporating additional constraints:

Gradient Direction Constraint: The voxel G_i must have the highest gradient intensity with a gradient direction consistent with the normal direction at P_i: $\nabla I(G_i) \cdot \mathbf{n}_i > 0$. This constraint is natural since the gradient is always directed inwards or outwards from a region of constant intensity value.

Intensity Value: The voxel G_i must have an intensity value in a given range value $[I_{min}, I_{max}]$. This constraint helps to discriminate against neighboring regions of high contrast with different intensity values.

The edge image is created by thresholding the maxima of the gradient norm in the gradient direction. The removal of small connected components may help to obtain reliable edge information. The traditional approach for the computation of the edge force $\mathbf{F}_{edge}$ has been to use distance maps built from the binary edge image [7, 23, 28]. Distance maps provide a simple algorithm for computing the direction of the edge force even at a long range. However, this approach has several limitations. First, it entails very large deformations away from the edge voxels, which causes an unstable behavior of the mesh model. Furthermore, distances are ambiguous at voxels equidistant from two edge voxels. Our method uses the normal orientation at vertex P_i to solve this ambiguity.

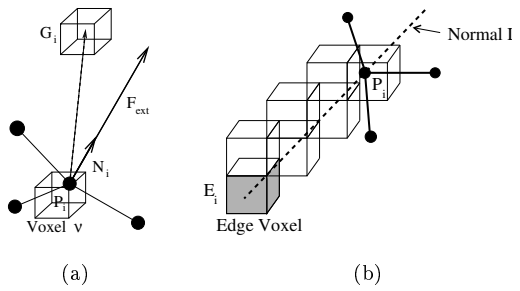


Fig. 16. (a) Computation of the gradient force $\mathbf{F}_{gradient}$: G_i is the center of the voxel with the highest gradient intensity in the neighborhood of voxel ν; (b) Search for the closest edge voxel along the normal line for the computation of $\mathbf{F}_{edge}$.

Our approach consists in finding the closest edge voxel in the normal direction of the mesh. At vertex P_i, we find the closest voxel $\mathcal{V}$, and a tridimensional line of voxels is scanned in the direction of $\mathbf{n}_i$ (see figure 16 (b)). The maximum number of voxels scanned is given by the reference distance D_{ref}, determined as a percentage of the overall radius of the edge image. In general, less than 30 voxels are scanned for each vertex. The tridimensional Bresenham line drawing algorithm is used to efficiently scan the voxel image along the line. If E_i is the closest edge voxel along the normal line, then the edge force is given by:

$$\mathbf{F}_{edge} = \beta_i^{edge}(E_i - P_i) \tag{15}$$

The vector $E_i - P_i$ is collinear with $\mathbf{n}_i$ and β^{edge} corresponds to the stiffness of the edge force, $0 \leq \beta_i^{edge} \leq 1$.

As with the gradient force, we can add constraints for finding the closest edge voxel :

gradient direction constraint: The voxel E_i must have the highest gradient intensity with a gradient direction consistent with the normal direction at P_i: $\nabla I(G_i) \cdot \mathbf{n}_i > 0$.

intensity value: The voxel E_i must have an intensity value in a given range value $[I_{min}, I_{max}]$.

4. Reconstruction Algorithm

4.1. Modeling Scheme

We propose a reconstruction scheme that is independent of the dataset type and that requires little a priori knowledge about the scene. The scheme is presented in figure 17 (a) and consists of two main stages.

The first stage corresponds to the mesh deformation from an initial position to a close approximation of the dataset shape. The mesh is initialized with a spherical, cylindrical, or planar topology. The user can choose between a manual initialization or an automatic initialization algorithm described in section 4.2, if the dataset has few outliers.

In all cases, the mesh topology can be changed after the first stage of deformation. It consists in creating holes, eventually followed by an increase of the surface genus, as seen in the diagram 17. All topological changes are performed semi-automatically with an automatic detection of topological problems, but a manual validation of all operations.

Once the mesh has the desired topology, the second stage of deformation consists in providing a more accurate geometric representation of the object by refining or adapting the mesh. Refinement and adaptation are two complementary tasks that respectively add vertices and move vertices towards parts of high curvature. Both tasks ensure that the final mesh will be at a given distance from the data while having good geometric and topological properties.

Only the rigidity parameter r_i and the external force parameter β_i are modified between the first and second stage. During the first stage, we set r_i to high values ($r_i \approx 10$) and β_i to low values ($\beta_i \approx 0.1$) in order to obtain smooth and large scale deformations on the mesh. However, during the second stage, we use low values of r_i ($r_i \approx 1$) and high values of β_i ($\beta_i \approx 0.5$) since a control of the fit is required.

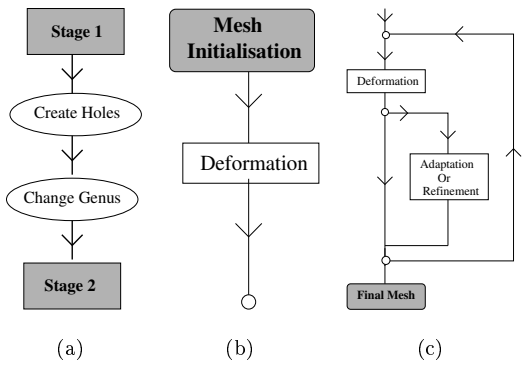


Fig. 17. (a) The general reconstruction scheme; (b) The first stage of deformation: the mesh is brought from its initial shape to a rough approximation of the data; (c) The second stage of deformation: the mesh is refined or adapted in order to create an optimal representation of the object.

4.2. *Initialization*

One of the main problems with reconstruction systems based on deformable models is that the recovered shapes depend on the initial position of the mesh. In general, these models need to be initialized in a close vicinity of the dataset. When modeling complex shapes, manual initialization may not give proper results, despite the ability of changing the mesh topology.

We propose an initialization algorithm that can only handle a limited number of topologies (three in the current implementation) but that performs well even in the presence of missing data. Initial models can be created from both range and volumetric images. The algorithm is not sensitive to noise but may be influenced by outliers.

The creation of a spherical mesh takes place by first computing the centroid G of the dataset, and then by finding the spherical surface centered on G enclosing the whole dataset. To do so, we consider a tessellated unit sphere centered on G, and then project each data point or voxel onto that sphere. We store on each surface element of the sphere, the maximum radial distance of projected points (see figure 18). When there are no points projecting on a sphere element, the mean value of the maximum projected radial distances is used instead. The variance of the maximum projected radial distances on the unit sphere determines the number of vertices in the initial simplex mesh.

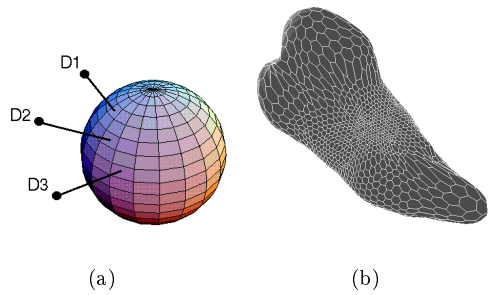


Fig. 18. (a) For initializing a spherical mesh, each data point is projected on a sphere; (b) The initial simplex mesh build from a parametric surface on spherical coordinates;

For cylindrical meshes, the axis of minimum inertia is first computed and then divided into n segments. Then, each data point is projected on a line segment and the centroid of the projected points on each intermediate planes defines the spine of the generalized cylinder (see figure 19 (a)). The cylinder's surface is built by considering the maximal radial distance of projected points on each slice. A simplex mesh of cylindrical topology is created based on this generalized cylinder with the extrapolation of missing parts. The variance of the radius determines the number of vertices in the initial mesh.

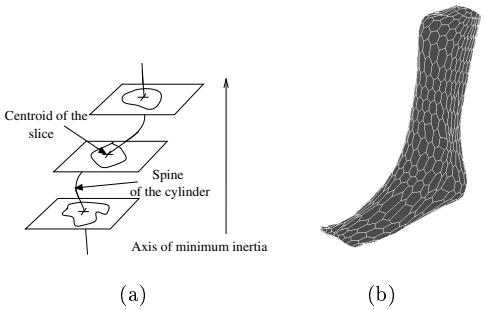


Fig. 19. (a) The axis of minimal inertia, the spine, and the slices used for the initialization of a cylindrical mesh; (b) The initial simplex build from the cylindrical surface;

Finally, the initialization of planar meshes is similar to the cylindrical case. Instead of the minimum inertia axis, we define the maximum inertia plane, passing through the centroid G and directed by the direction of maximum inertia (see figure 20). We project data points onto the plane, and compute for each planar element, the average height from the plane. Similarly, the variance of the height determines the number of vertices in the initial mesh.

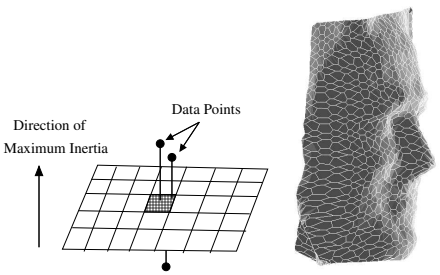


Fig. 20. (a) The initialization of a planar mesh is based on the projection of data points on the plane of maximum inertia; (b) The initial simplex mesh build from the elevation map;

4.3. Changing the mesh topology

After the initialization stage, it may occur that the mesh does not have the same topology as the object. Since topology of tridimensional surfaces are characterized by the number of their holes and their genus number, we propose two algorithms for changing those topological characteristics. Since initial meshes have, in general, a simpler topology than the existing objects, we only focus on building models with additional holes and increased genuses.

4.3.1. Hole Creation The main idea of this algorithm is to cut pieces of surface mesh where there are no corresponding data points. More precisely, we label all faces having vertices located at a distance greater than the reference distance D_{ref} from the dataset. Those faces are gathered into a set of zones that are cut with the topological operator T_3^2. A contour is created around each hole and after few iterations, each contour deforms under the influence of internal and external forces.

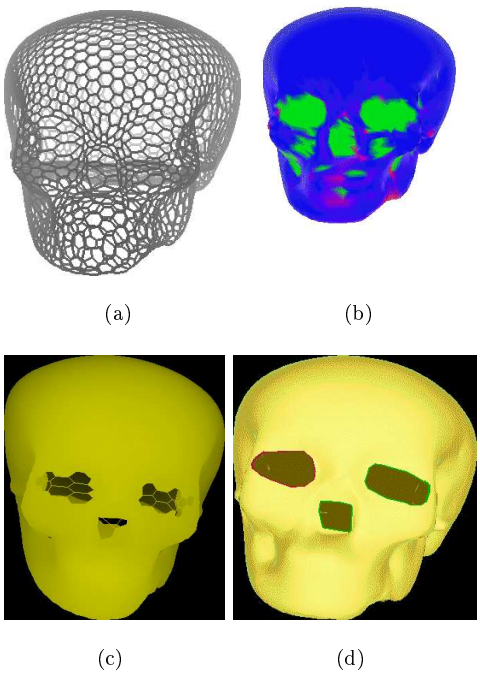


Fig. 21. Hole creation on a skull model: (a) Mesh after the first stage of deformation; (b) Color coding of the distance of mesh vertices to the dataset; (c) Zones to be cut; (d) Mesh after the hole creation algorithm;

We present two examples of hole creation with the reconstruction of a skull model (see figure 21) and face model (see figure 22). For the skull, holes has been created around the the orbits, the nose and the foramen. Other zones have not been cut due to their small size.

The face example shows why this procedure cannot be made fully automatic. Two zones have indeed been created: one around the upper part of the head and one at the neck level. The zone located at the hair level corresponds to missing data due to the limitation of the acquisition technology. On the other hand, the zone located around the neck does not correspond to any existing physical points and should be removed. This is why the hole creation algorithm requires some user interaction based on his a priori knowledge of the tridimensional scene.

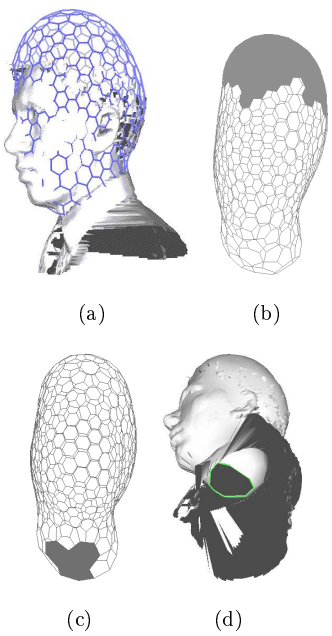


Fig. 22. Holes creation on a face model : (a) Simplex mesh after the first stage of deformation; (b) The zone corresponding to the missing data around the hair; (c) Zone corresponding to the neck; (d) Mesh after cutting the neck zone;

4.3.2. Genus Modification The genus of a surface is a topological characteristic defined as the number of surface handles. When dealing with incomplete data, the initialization stage proposed in section 4.2 provides meshes with zero genus (sphere, cylinder, or plane). It is therefore necessary to provide an algorithm that increases the mesh genus.

We propose an algorithm that consist in connecting, with the topological operator T_4^2, two contours surrounding two holes. Among all pairs of contours surrounding holes, we select those verifying the following criteria:

1. The centers of the two contours must be close to each other.
2. The diameters of the two contours must be similar.
3. The two contours must be located inside nearly parallel planes.

In figure 23, we initialize a mesh of spherical topology around a vertebra. The algorithm described in the previous section, automatically creates two holes on each side of the handle. If those contours verify the three criteria above listed, the genus is increased by applying the topological operator T_4^2. The mesh is subsequently refined to closely fit the model.

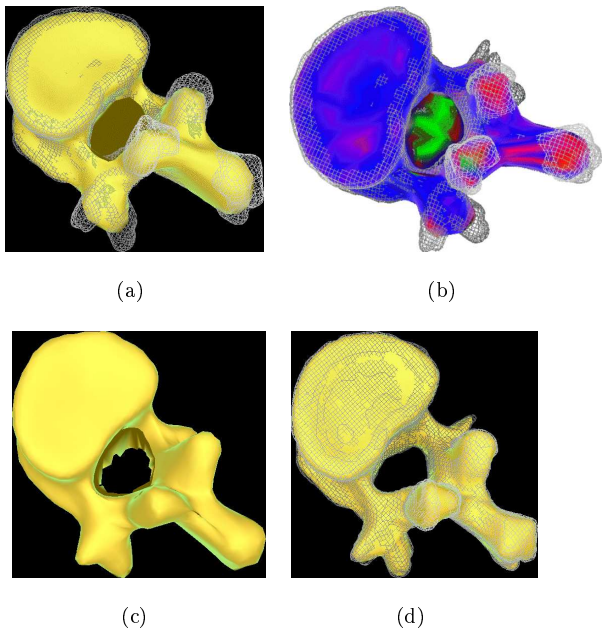


Fig. 23. (a) Deformation of a mesh initialized as a sphere on a vertebra dataset; (b) Color coding of the distance to the data; (c) The hole creation algorithm generates two holes; (d) The two contours are merged and the mesh has a toroidal topology;

4.4. Controlling the closeness of fit

4.4.1. The Adaptation Algorithm Since simplex meshes are discrete representation of surfaces, the mesh goodness of fit not only depends on the distance of vertices to the data, but also on the relative location of vertices on the surface object. A good strategy is to concentrate vertices towards parts of high curvature into order to optimize the shape description.

The reference metric parameters $\tilde{\epsilon}_i = \{\tilde{\epsilon}_{1i}, \tilde{\epsilon}_{2i}, \tilde{\epsilon}_{3i}\}$ control the relative position a vertex with respect to its three neighboring vertices. Our algorithm

updates the values of the reference metric parameters depending on the mean curvature values of neighboring vertices.

Basically, the algorithm operates as follows: vertices of low mean curvature migrate towards neighboring vertices of relatively higher mean curvature whereas vertices of constant mean curvature have metric parameters close to one third.

The reference metric parameters are updated every p iterations ($p \geq 1$) by the following relaxation algorithm:

$$\tilde{\epsilon}_i^{t+p} = \tilde{\epsilon}_i^t + \frac{1}{2}(\tilde{\epsilon}_i^{opt} - \tilde{\epsilon}_i^t))$$

where $\tilde{\epsilon}_i^t$ is the reference metric parameters at iteration t and $\tilde{\epsilon}_i^{opt}$ is the optimal metric parameter value based on the local variation of the mean curvature at P_i.

To get a stable estimate of the mean curvature variation at a given vertex, we first consider the mean value of the absolute mean curvature value $|\bar{H}_i|$:

$$|\bar{H}_i| = (|H_{N_1(i)}| + |H_{N_2(i)}| + |H_{N_3(i)}|)/3$$

We then compute the relative mean curvature deviation vector $\delta|H|_i$ as:

$$\delta|H|_i = \begin{pmatrix} \frac{|H_{N_1(i)}| - |\bar{H}_i|}{|H_i|} \\ \frac{|H_{N_2(i)}| - |\bar{H}_i|}{|H_i|} \\ \frac{|H_{N_3(i)}| - |\bar{H}_i|}{|H_i|} \end{pmatrix}$$

The vector $\delta|H|_i$ is related to the variation of absolute mean curvature between one vertex and its three neighbors. The vector norm is null if the absolute mean curvature is locally constant. The optimal reference metric parameter value is related to $\delta|H|_i$ with the following relationship:

$$\tilde{\epsilon}_i^{opt} = \frac{1}{3} + \gamma_i \delta|H|_i \qquad (16)$$

where γ_i is a constant that controls the extent of the vertex adaptation and is usually chosen between 0.03 and 0.25. However, since reference metric parameters $\tilde{\epsilon}_{ki}$ must belong to the interval $[0, 1]$, we may compute a value of γ_i substantially lower than 0.03. In practice, we choose to update the metric parameters every $p = 10$ iterations in order to stabilize the mesh before re-evaluating the mean curvature over the mesh. Example of mesh adaptation is provided in figure 9 (a) and (b) on a mesh representing a cube. Figure 9 (a) corresponds to a mesh deformation without adaptation (reference metric parameters equal to one third) whereas figure 9 (b) corresponds to a mesh adaptation with $\gamma_i = 0.20$.

4.4.2. The Refinement Algorithm Since the shape description of a simplex mesh is determined by the number of its vertices, we introduce a face refinement algorithm that controls the mesh resolution. Our algorithm is based on a refinement measure describing whether a face is a good candidate for refinement. Several geometric criteria may be combined to estimate the refinement measure.

More precisely, we define the following four geometric criteria as refinement measures:

The relative face area : It is computed as the ratio of a face area to the total mesh area.

The face elongation : It is computed as 1.0 plus the difference of length between the longest edge and the smallest edge of a face, divided by the median value of face edges length.

The face Gaussian curvature : It is evaluated as the area of the spherical polygon described by the face normal vectors on the Gaussian sphere.

The face distance to a 3D dataset : It is based on the ratio between the smallest distance to a 3D dataset with the cut-off distance D_{ref}.

The refinement algorithm works in an iterative manner (see figure 24 (a)) at every q iterations ($q = 5$ in general). We first sort all faces according to their refinement measure such that faces having high refinement measures are first processed. Each face having a refinement measure above a given threshold, is refined by creating a new edge with the topological operator T_2^2. The mesh is then locally smoothed around each newly created edge and neighboring edges are eventually swapped in order to keep the number of vertices per face close to six. Finally, we recompute the refinement measure for all refined faces and iterate until all faces have a refinement measure below the threshold.

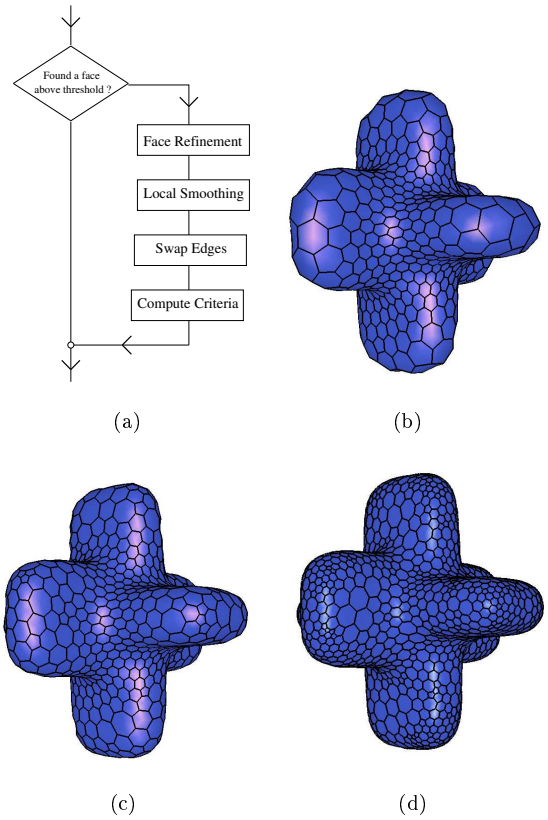


Fig. 24. (a) Flow chart of the refinement algorithm;(b) A simplex mesh fit on a synthetic cross without refinement (1620 vertices); (b) Mesh with a low level of refinement (1970 vertices); (c) Mesh with a high level of refinement (4020 vertices);

This algorithm is fairly efficient because all operations are performed locally. Figure 24 shows an example of mesh refinement based on the distance to a synthetic cross dataset.

We often choose the curvature and distance to 3D dataset as the adequate criteria of refinement. However, when the 3D dataset is very noisy, or has many outliers, the distance criterion should not be used since it would result in actually fitting the noise. Finally, it is often useful to restrict the refinement algorithm to given parts of the mesh in order to optimize the total number of vertices of the simplex mesh.

5. Simplex meshes versus Triangulated meshes

Simplex meshes and triangulated meshes are topologically dual of each other. This duality implies that all topological operations defined on a simplex mesh can be defined on a triangulation and inversely. However, in terms of data structure, simplex meshes are slightly easier to work with because of the constant vertex connectivity and because borders are considered as empty faces of a simplex mesh. Note that borders of triangulations do not correspond to triangles and therefore must be handled in a separate data structure.

The main difference between simplex meshes and triangulation is geometric in nature. On a vertex of a simplex mesh, surface normal, mean curvature, simplex angle, and metric parameters are unambiguously defined. We have devised a simple and efficient internal force on simplex meshes solely based on these geometric quantities. Efficiency is an important issue for shape recovery because it allows some level of user interaction. Furthermore, we have found that the C^2 smoothness constraint is the appropriate constraint for extrapolating missing data since the C^1 smoothness constraint tends to shrink surfaces towards their center of curvature.

Several simple and efficient schemes have been proposed for deforming triangulations under the C^1 continuity constraint (a good example can be found in [30]). However, as seen in section 1.4, enforcing the C^2 continuity constraint on non-parametric triangulated representations requires either sophisticated spline formulations [26] or subdivision surfaces [16, 33]. Those representations lead to more computationally intensive algorithms and are not well-suited for a fast update of their data structure after a local (mesh refinement) or global (genus increase) topological change. Indeed, a key advantage of simplex meshes lies in their internal regularizing force that is only based on the vertices position without any additional degrees of freedom. Therefore, all topological operations can be done efficiently with a simple data structure.

However, a disadvantage of using simplex meshes over triangulations is that they must be triangulated in order to be displayed or transfered to other high level tasks. In general, for triangulating a simplex mesh, the dual triangulation is

not a good choice because it strongly underestimates the vertex curvature. Instead, it is preferable to triangulate each face independently for instance by adding a central vertex.

6. Results

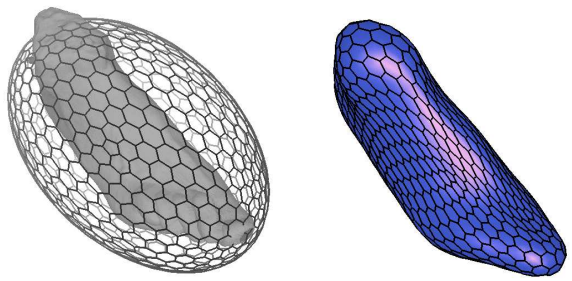
6.1. Quantitative Results

6.1.1. Influence of the Damping Factor Our deformable model framework is based on a Newtonian law of motion (see equation 9), that includes a damping factor in order to prevent oscillations of the system. Most deformable model systems, on the contrary, are based on a Lagrangian law of motion, where the speed of each vertex is equal to the sum of the internal and external forces.

In this section, we compare the efficiency of the Newtonian law of motion with the Lagrangian law of motion. We recall that the discrete Newtonian law of motion is:

$$P_i^{t+1} = P_i^t + (1-\gamma)(P_i^t - P_i^{t-1}) + \mathbf{F}_{int} + \mathbf{F}_{ext} \quad (17)$$

We note that if $\gamma = 1$, then equation 17 corresponds to a Lagrangian law of motion. As the damping factor γ decreases towards zero, the effect of the acceleration increases. For $\gamma = 0$, the deformable model behaves as a perfect oscillator. We have tested the effect of the damping factor on the deformation of a simplex mesh on a range image of a foot. The mesh is initialized as an ellipsoid and we perform 30 iterations of deformation. The initial mesh position and the deformed mesh with a value of $\gamma = 0.80$ are shown in figure 25.



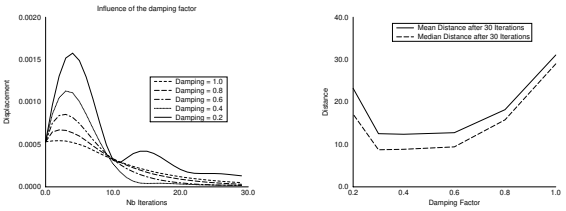
(a) (b)

Fig. 25. (a) Initialization of the mesh around the foot range data; (b) Mesh after 30 iterations with $\gamma = 0.80$

In figure 26, we have studied the displacement of vertices during the deformation for several values of the damping factor. The displacement is defined as the sum of vertex motion between two iterations. It appears that as γ decreases, the convergence speed increases. However, when γ is smaller than 0.20, large oscillations prevent the mesh from converging towards the shape of the range data. For $\gamma = 0.20$, we observe that the resulting mesh self-intersects.

Table 4. Accuracy and computational cost of the projection and kd-tree method.

	Accuracy Rate	Mean Error	Maximum Error	D_{ref}	Computation Cost of the Projection Method	Computation Cost of the KD-tree
Initial Mesh	33 %	36.93 mm	371 mm	380 mm	0.79 ms	27.20 ms
Deformed Mesh	95.2 %	0.38 mm	45.89 mm	95 mm	0.81 ms	2.21 ms
Refined Mesh	98.7 %	0.04 mm	14.32 mm	95 mm	0.81 ms	2.02 ms



(a) (b)

Fig. 26. (a) Chart of the mean vertex displacement as a function of the iteration number for several values of the damping factor; (b) The mean and median distance to the range data for several values of the damping factor after 30 iterations.

The results show that the Newtonian law of motion is more efficient than the Lagrangian law of motion. However, the choice of the damping factor must be governed by the trade off between efficiency and stability. If γ is too small, the mesh may not converge towards the right shape, especially when the mesh is far away from the data. If γ is too close from 1.0, then the efficiency of the deformation process is poor. In practice, we choose a value of $\gamma = 0.65$.

6.1.2. Performance Evaluation of Closest-Point Algorithms We have introduced two methods for finding the closest data point to a given vertex. The projection method uses the calibration matrices of camera-based range-finders to find the closest point along the normal line. This method does not guarantee that we will find the true closest data point, but it is fairly computationally efficient. On the other hand, the kd-tree data structure, gives the true closest point from a vertex, but at a higher memory and computation cost.

In this section, we compare the accuracy and the computational efficiency of these two methods at three different stages of deformation: initialization, the first deformation and after refinement. The three meshes are shown respectively in figure 25 (a) and figures 27 (a) and (b).

Since the kd-tree method gives the true closest point to the data, we evaluate the accuracy of the projection method in comparison with the distance given by the kd-tree method. All distances are measured along the vertex normal direction, in order to remove the effect of sparse data. In figure 4, we can see that the accuracy of the projection method, increases drastically as the mesh is closer to the range data. The maximum errors occur at parts where the range data is incomplete (such as the tip of the foot) and therefore does not have an important influence on the overall shape of the deformable model.

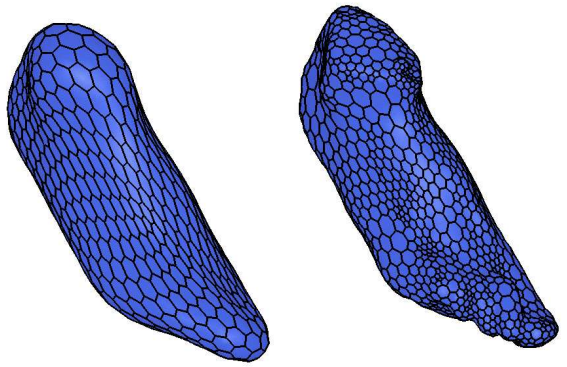
The computational cost of both methods are compared in figure 4. It appears that the efficiency of the kd-tree method is linked with the radius D_{ref} of search around the vertex which is not the case for the projection method. When the mesh is far from the data, the kd-tree method is 35 times slower than the projection method whereas it is only 2.5 times slower for lower value of D_{ref}. In all cases, the time needed to build a kd-tree from a 512×512 range image was $4.03s$.

From this experiment, we conclude that when the calibration parameter of the range sensor are known, the projection method is better-suited than the kd-tree method because of its low computational and memory requirement. Furthermore, the poor accuracy of the projection method when far from the data is balanced by the low value of the stiffness parameter β_i ($\beta_i = 0.1$).

6.1.3. Effect of the Refinement on the Distance Error In this section, we study the effect of refinement on the overall distance error. Starting from a rough estimate shown in figure 25 (a), the mesh position after applying the first stage of deformation is shown in figure 27 (a). This model has 1280 vertices and over-smoothes the shape of the original data.

We refine this mesh based on its distance to the dataset. In this example, we choose two different refinement levels set by two refinement measures. The former refinement measure has been automatically computed in order to refine 25% of the original number of faces whereas the latter corresponds to a refinement level of 40%. Since the refinement measure is linked to the maximum distance to the data, choosing a refinement measure is equivalent to choosing a maximum value for that distance. The first refinement measure corresponds to a maximum distance of 22.10 mm and the second to a distance of 14.25 mm. Note that

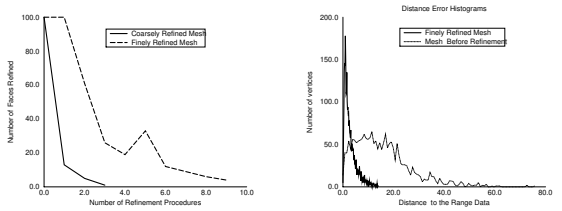
this distance is computed not only for all mesh vertices, but also for the all face centers.



(a) (b)

Fig. 27. (a) The mesh of figure 25 (a) after the first stage of deformation; (b) Mesh after fine refinement.

Meshes before and after refinement are shown in figure 27. Quantitative results on the distance to the original data, are given in figure 5. The refinement procedure is called every 5 iterations and stops when there are less than 5 faces to refine. We limit the number of faces refined per procedure to 100, in order to obtain a more homogeneous spacing of vertices. The number of faces refined per call is described in figure 28 (a). The total number of iterations increases from 45 to 130 as the required refinement level is modified.



(a) (b)

Fig. 28. (a) The number of faces refined for each call of the refinement procedure; (b) The distance error histogram before and after refinement.

In figure 28 (b), we can see the change of the distance error histogram before and after refinement. As shown in figure 5, the maximum error is within the bounds set by the refinement criterion. Note that the median error does not vary substantially between the two levels of refinement since it approximately corresponds to the range data resolution.

The computation times were evaluated on a DEC Alphastation 200/233. We indicate in figure 5 the time needed for the first stage of deformation (where the mesh is deformed from an ellipsoid) as well as for the refinement stage. Within the refinement stage, most of the time is spent on evaluating the refinement criterion for all faces.

6.2. *Qualitative Results*

6.2.1. Changing the Connectivity In this section, we demonstrate that our reconstruction scheme can take into account the change of connectivity. More precisely, starting from a simplex mesh, the

Table 5. Distance error and computation time for three meshes at different refinement levels.

	Mean Distance	Median Distance	Maximum Distance	Number of Vertices	Number of Iterations	Computation Time
Deformed Mesh	16.01 mm	14.53 mm	75.70 mm	1280	30	23.77 s
Coarsely Refined Mesh	3.56 mm	2.44 mm	21.98 mm	1518	20	35.73 s
Finely Refined Mesh	2.94 mm	2.27 mm	14.07 mm	2020	50	124.43 s

creation of holes, as described in section 4.3.1, can cut a simplex mesh into two independent meshes. The change of connectivity occurs when the part of the zone interpolated has more than one border.

In this example, we use a pre-segmented volumetric image consisting of manually segmented contours. The difficulty lies in building a smooth geometric model that closely fits those contours. Direct reconstruction techniques, such as Delaunay triangulation [3] or the Marching Cubes algorithm, extrapolate the contours without taking into account any smoothness constraints.

In our framework, the deformable mesh is attracted by the volumetric image and submitted to regularizing forces. To compute the attraction force, we simply consider the contours as edge voxels. We start the reconstruction with an automatic mesh initialization of spherical topology around the data. We obtain a mesh surrounding both lungs (see figure 29 (a)). The first deformation stage creates a smooth model where we

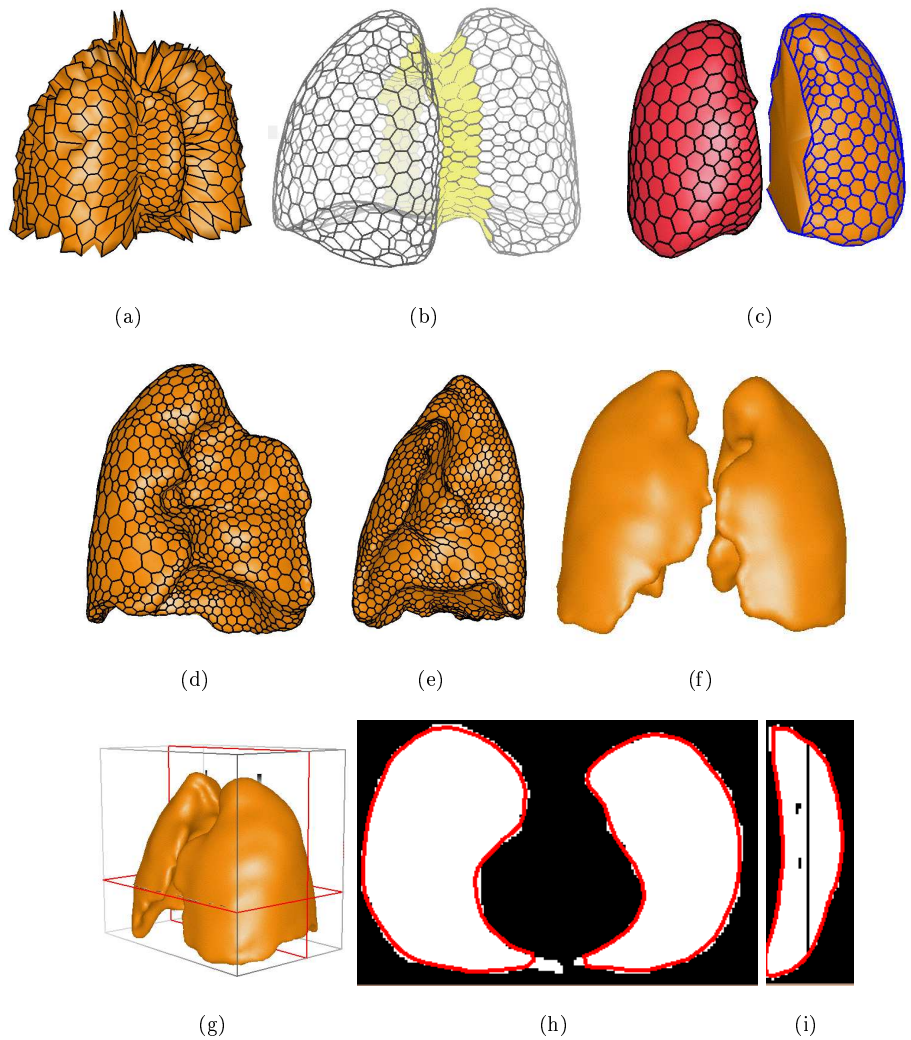


Fig. 29. (a) Initial mesh; (b) A zone detected as interpolated; (c) Change of connectivity; (d) and (e) Reconstructed model of the first and second lung; (f) and (g) Final reconstruction; (h) and (i) Slices of the lung models.

can correctly detect the part which has been interpolated. Since this interpolated part has two borders, its removal entails the creation of a second simplex mesh model (figure 29 (c)). Those meshes are then refined and the final reconstruction is pictured in figure 29(f).

6.2.2. Reconstruction from Volumetric Images In this example, we reconstruct the ventricles, atria and the pericardium from a T1-weighted MRI volumetric image. The image corresponds to a routine acquisition of the abdomen and consists of only 10 slices without any contrast agent. Because of the low-contrast, deformable models are the only alternative to manual segmentation. Furthermore, we cannot automatically initialize our models because it is impossible to isolate the ventricles and atria in the image. Instead, the ventricles and atria models are initialized as ellipsoids (figure 30 (a)) and positioned manually within the volumetric image. The pericardium is initialized as a cylinder.

All models are attracted towards edges with the additional constraint that the surface normal must be coherently oriented with the image gradient direction. The gradient information was computed as a 2D gradient since the inter-slice distance is six times larger than the pixel size.

First, the different models are deformed with a high rigidity parameter and, in a second stage, the rigidity parameter is decreased to a minimum while slightly refining the meshes based on area. Because of the rough mesh initialization, we allow the user to locally guide the mesh in order to remove it from a local minimum. However, we found that using the coherence between the gradient direction and the orientation of the deformable model greatly helps the segmentation. The right and left ventricles have been easily segmented (figure 30 (b) and (c)) whereas the recovery of the atria is more problematic. In particular, the right and left atrium intersect each other because the septum separating the two cavities is missing in the image (figure 30(h)). In figure 30(i), we show the slices of the different reconstructed models. A better fit could be achieved by increasing the resolution of the models at the expense of computation time. The number of vertices for the right/left ventricles, the right/left atrium and the pericardium is respectively 1920, 1280, 1320, 1592 and 366.

6.2.3. Reconstruction by Parts When the object to recover is strongly non-convex or non star-shaped, it may be necessary to recover sub-parts of that object and subsequently join the different parts.

For instance, we have reconstructed a hand model by reconstructing independently the palm and the 5 fingers. Those models have been initialized with a sphere and 5 cylinders, as seen in figure 31 (a). After recovering each piece, we manually connect them together with several T_3^2 operators. The junctions between meshes are then automatically smoothed to ensure a continuity of normal orientation. The final model has about 8000 vertices and uses the texture of both range data. The flexibility of the simplex mesh representation allows us to perform the reconstruction by parts in a straightforward manner.

7. Conclusion

In this paper, we have presented a general reconstruction framework based on deformable simplex meshes. It differs from previous approaches by using a non-parametric representation of surfaces and by the addition of deformable contours lying on the surface model. The deformable models can be refined, adapted, and the level of smoothness can be controlled in a computationally efficient manner. Because simplex meshes can represent surfaces of all topologies, we have proposed an algorithm for adapting the mesh topology to the topology of the object. Finally, we have proposed new formulations of the external forces from range data or volumetric images.

In the current implementation, we rely on an initialization algorithm that is limited to only three different topologies. Furthermore, it requires that objects of interest have been previously isolated from other objects. We intend to develop a more general initialization technique that does not require any a priori knowledge of the object topology. Such an approach must be coupled with a low-level segmentation algorithm that can extract a region of interest.

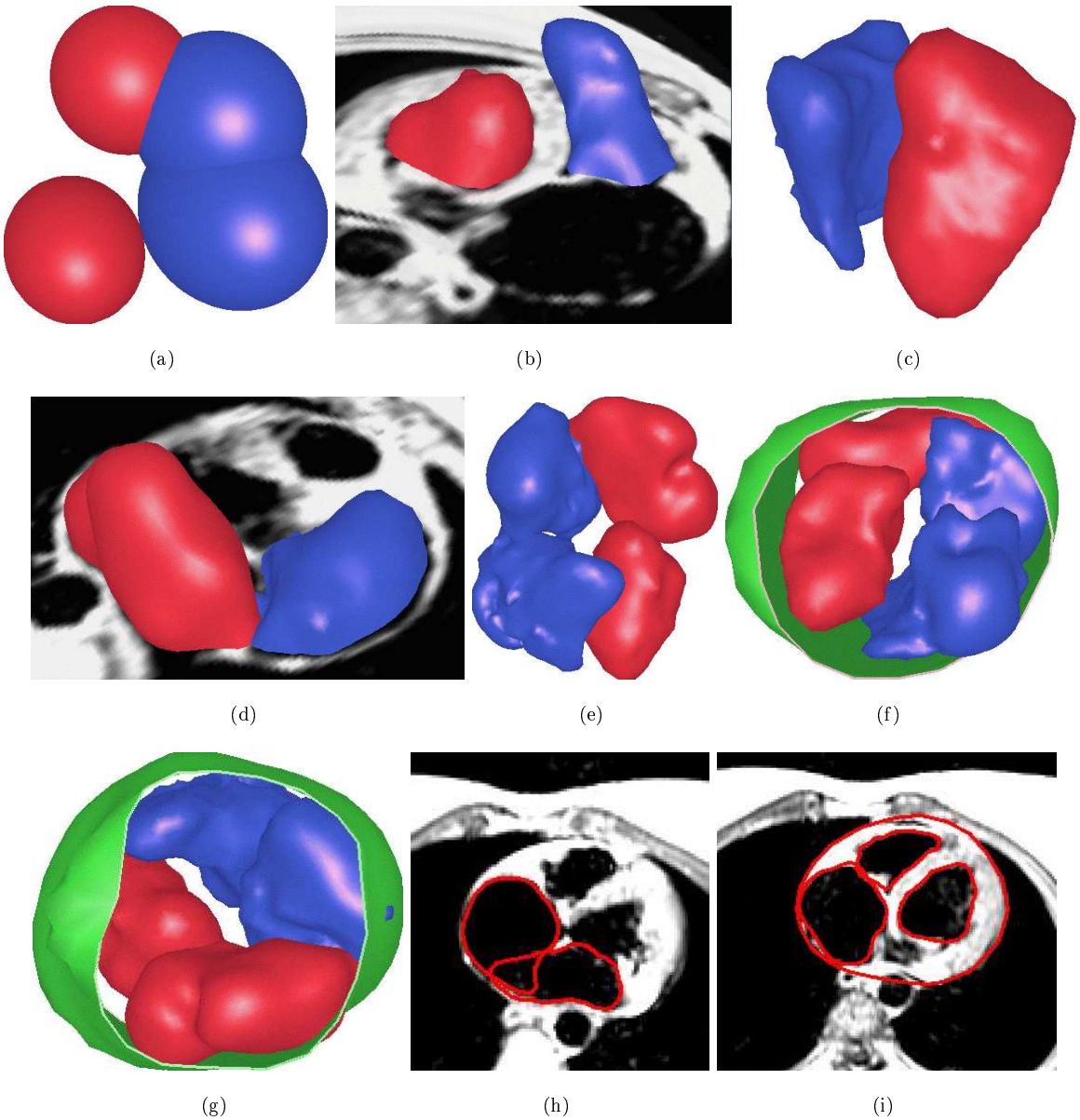


Fig. 30. (a) Initial meshes; (b) and (c) The right and left reconstructed ventricles; (d) The right and left atrium; (e) (f) and (g) The 5 recovered models from the MRI image; (h) and (i) Slices of the different models.

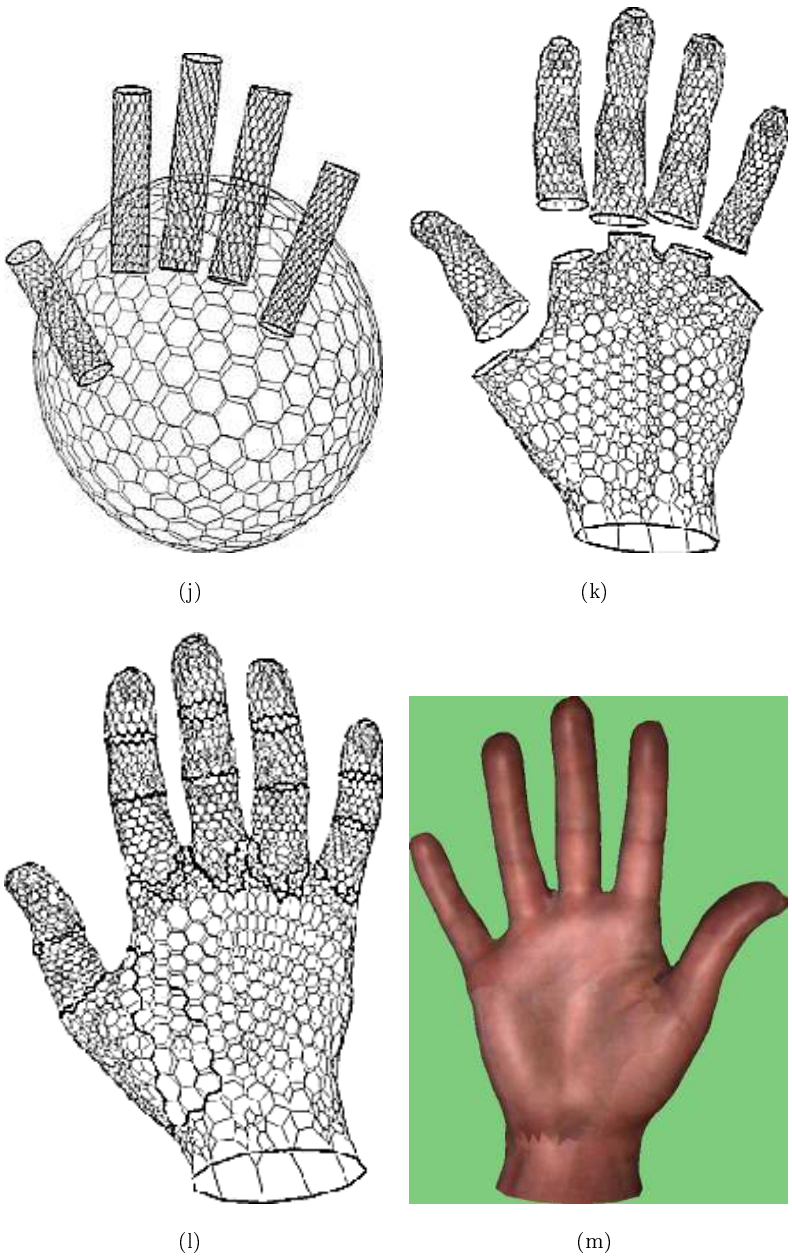


Fig. 31. (a) Initial estimate of the palm and the five fingers; (b) The six recovered meshes; (c) Simplex mesh model after connecting the meshes together; (d) Textured hand model.

Finally, when the object to recover is known a priori, the robustness of segmentation and reconstruction could be improved by including in the deformable model a notion of shape statistics. This would specifically be beneficial for segmenting medical images where databases of volumetric images and anatomic structures could be used.

Acknowledgements

The author would like to thank Y. Watanabe from NTT Yokosuka Research Laboratory for providing the Cyberware range images as well as Pr Darcourt from the Centre Antoine Lacassagne in Nice (France) for providing the MRI heart images. Also, the author would like to thank all the members of the Epidaure project for interesting discussions regarding the topic of this paper.

Appendix

1. Approximation Problem

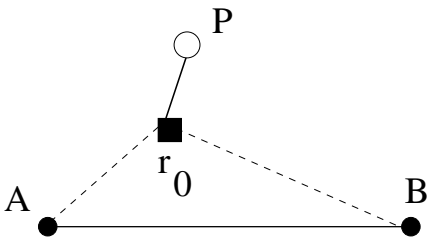


Fig. A.1. The solution of this approximation problem is piecewise linear.

From the Euler-Lagrange equation associated with equation 1, it can be proved that the curve minimizing equation (1) is piecewise linear. If we note $\mathbf{A}$ and $\mathbf{B}$ the extremities of the curve, $\mathbf{P}$ the data point and $\mathbf{r}_0$ the vertex of parameter u_0, then the problem lies in finding $\mathbf{r}_0$ that minimizes:

$$E(\mathbf{r}_0) = \frac{\|\mathbf{A}-\mathbf{r}_0\|^2}{u_0} + \frac{\|\mathbf{B}-\mathbf{r}_0\|^2}{1-u_0} + \lambda\|\mathbf{r}_0-\mathbf{P}\|^2$$

which leads to the following equation:

$$\mathbf{r}_0 = \frac{\lambda u_0(1-u_0)\mathbf{P} + u_0\mathbf{B} + (1-u_0)\mathbf{A}}{1+\lambda u_0(1-u_0)}$$

Replacing the sum of square derivatives with the total curve length, removes u_0 in the expression of the optimal curve:

$$E_{\text{intrinsic}}(\mathbf{r}_0) = \|\mathbf{A}-\mathbf{r}_0\| + \|\mathbf{B}-\mathbf{r}_0\| + \lambda\|\mathbf{r}_0-\mathbf{P}\|^2$$

which leads to the following non-linear equation that does not have any closed form solution:

$$\mathbf{r}_0 = \frac{2\lambda\|\mathbf{B}-\mathbf{r}_0\|\|\mathbf{A}-\mathbf{r}_0\|\mathbf{P} + \|\mathbf{A}-\mathbf{r}_0\|\mathbf{B} + \|\mathbf{B}-\mathbf{r}_0\|\mathbf{A}}{\|\mathbf{B}-\mathbf{r}_0\| + \|\mathbf{A}-\mathbf{r}_0\| + 2\lambda\|\mathbf{B}-\mathbf{r}_0\|\|\mathbf{A}-\mathbf{r}_0\|}$$

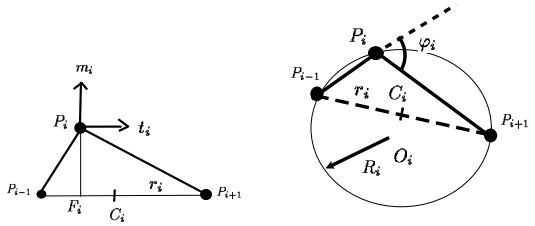
2. Geometry of Planar 1-Simplex Meshes

The geometry of a planar 1-simplex mesh $\mathcal{M} \in \mathbb{R}^2$ is described by its metric parameters and its simplex angle.

We first define the local tangent $\mathbf{t}_i$ and normal vector $\mathbf{n}_i$ around a vertex P_i as:

$$\mathbf{t}_i = \frac{P_{i+1}-P_{i-1}}{\|P_{i+1}-P_{i-1}\|} \quad \mathbf{n}_i = \mathbf{t}^{\perp}$$

The local curvature κ_i is defined as the inverse of the radius of the circumscribed circle at (P_{i-1}, P_i, P_{i+1}) (see figure A.2). We define as well r_i as the half distance between P_{i-1} and P_{i+1} : $r_i = \|P_{i+1}-P_{i-1}\|/2$.



(a) (b)

Fig. A.2. (a) Definition of tangent and normal vector around a vertex P_i of a 1-simplex mesh. (b) Definition of the simplex angle φ_i and the circumscribed circle.

We then define the metric parameters and the simplex angle at P_i.

Definition 3. Let P_i be a vertex of a planar 1-simplex mesh. Let F_i be the orthogonal projection of P_i onto the line $[P_{i+1}-P_{i-1}]$. Then, the metric parameters at P_i, $(\epsilon_{1i}, \epsilon_{2i})$ are defined as

the barycentric coordinates of F_i relative to P_{i-1} and P_{i+1}:

$$\epsilon_{1i} P_{i-1} + \epsilon_{2i} P_{i+1} = F_i$$
$$\epsilon_{1i} + \epsilon_{2i} = 1$$

Definition 4. The simplex angle φ_i at P_i is the oriented angle existing between the two adjacent segments $[P_{i-1}P_i]$ and $[P_i P_{i+1}]$.

The fundamental relation gives the position of vertex P_i as a function of its two neighbors P_{i-1} and P_{i+1} and the three shape parameters $(\epsilon_{1i}, \epsilon_{2i}, \varphi_i)$:

$$P_i = \epsilon_{1i} P_{i-1} + \epsilon_{2i} P_{i+1} + L(r_i, |2\epsilon_{1i} - 1| r_i, \varphi_i)\mathbf{n}_i \quad \text{(A1)}$$

The values of the metric parameters and the simplex angle describe the shape of the mesh up to a translation, rotation, and scale transformation.

3. Geometry of Tridimensional 1-Simplex Mesh

Contours are tridimensional 1-simplex mesh $\mathcal{M} \in \mathbb{R}^3$, where we define a local frame made of the tangent $\mathbf{t}_i$, normal $\mathbf{n}_i$ and binormal $\mathbf{b}_i$:

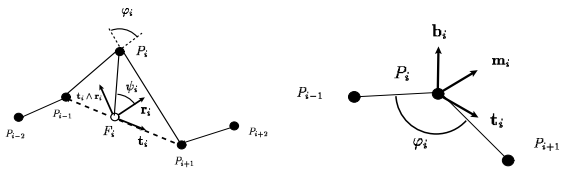
$$\mathbf{t}_i = \frac{P_{i+1} - P_{i-1}}{\|P_{i+1} - P_{i-1}\|} \quad \text{(A2)}$$
$$\mathbf{n}_i = \mathbf{b}_i \wedge \mathbf{t}_i \quad \text{(A3)}$$
$$\mathbf{b}_i = \frac{(P_i - P_{i-1}) \wedge (P_{i+1} - P_i)}{\|(P_i - P_{i-1}) \wedge (P_{i+1} - P_i)\|} \quad \text{(A4)}$$

The metric parameters and the simplex angle are defined as for planar meshes. However, we need to introduce another angle ψ_i as additional shape parameter. This parameter is defined through vector $\mathbf{r}_i$:

$$\mathbf{r}_i = \frac{\mathbf{t}_i \wedge ((P_{i-1} - P_{i-2}) \wedge (P_{i+2} - P_{i+1}))}{\|\mathbf{t}_i \wedge ((P_{i-1} - P_{i-2}) \wedge (P_{i+2} - P_{i+1}))\|}$$

($\mathbf{r}_i$ and $\mathbf{t}_i \wedge \mathbf{r}_i$) defines the normal plane, orthogonal to $\mathbf{t}_i$. Therefore, we define ψ_i as the angle between $\mathbf{n}_i$ and $\mathbf{r}_i$:

$$\mathbf{n}_i = \cos(\psi_i)\mathbf{r}_i + \sin(\psi_i)\mathbf{t}_i \wedge \mathbf{r}_i$$



(a) (b)

Fig. A.3. (a) Definition of curvature κ_i with the radius of the circumscribed circle $R_i = 1/\kappa_i$. C_i is the middle of segment $[P_{i-1}P_{i+1}]$. (b) Definition of tangent and normal vectors. F_i is the foot of P_i on $P_{i-1}P_{i+1}$.

The shape of a tridimensional simplex mesh is then fully described by its metric parameters, simplex angle and angle ψ_i :

$$P_i = \epsilon_{1i} P_{i-1} + \epsilon_{2i} P_{i+1} + L(r_i, d_i, \varphi_i)\cos(\psi_i)\mathbf{r}_i + L(r_i, d_i, \varphi_i)\sin(\psi_i)\mathbf{t}_i \wedge \mathbf{r}_i$$

with $r_i = \frac{1}{2}\|P_{i+1} - P_{i-1}\|$ and $d_i = |2\epsilon_{1i} - 1| r_i$.

4. Internal Force Applied on Contours.

The internal force applied on a tridimensional contour $\mathcal{C}$ can also be decomposed into a tangential and normal force.

$$\mathbf{F}_{int} = \mathbf{F}_{\mathrm{T}angent} + \mathbf{F}_{\mathrm{N}ormal}$$

For the tangential force, we define a set of reference metric parameters $(\tilde{\epsilon}_{1i}, \tilde{\epsilon}_{2i})$ which corresponds to the position in the tangent plane of a vertex with respect to its 2 neighbors. If P_i is a vertex of $\mathcal{C}$ and if P_{i-1}, P_{i+1} are its two neighbors, then the tangential force is expressed as:

$$\mathbf{F}_{\mathrm{T}angent} = (\tilde{\epsilon}_{1i} - \epsilon_{1i})P_{i-1} + (\tilde{\epsilon}_{2i} - \epsilon_{2i})P_{i+1}$$

With the notation of appendix A.3, the normal force is written as:

$$\mathbf{F}_{\mathrm{N}ormal} = \left(L(r_i, d_i, \tilde{\varphi}_i)\cos(\tilde{\psi}_i)` - L(r_i, d_i, \varphi_i)\cos(\psi_i)\right)\mathbf{r}_i + \left(L(r_i, d_i, \tilde{\varphi}_i)\sin(\tilde{\psi}_i)` - L(r_i, d_i, \varphi_i)\sin(\psi_i)\right)\mathbf{t}_i \wedge \mathbf{r}_i$$

The choice of $\tilde{\psi}_i$ and $\tilde{\varphi}$ determines the level of continuity constraint.

Position Continuity or C^0 Constraint: . We set $\tilde{\varphi}_i = \varphi_i$ and $\tilde{\psi}_i = \psi_i$. The normal force is then null and the contour can freely bend around each vertex.

Normal Continuity or C^1 Constraint: We set $\tilde{\varphi}_i = 0$ and $\tilde{\psi}_i = 0$. The force then simply writes as $\mathbf{F}_{int} = \tilde{\epsilon}_{1i} P_{i-1} + \tilde{\epsilon}_{2i} P_{i+1} - P_i$ which corresponds to "Laplacian smoothing".

Curvature Continuity or C^2 Constraint: We set $\tilde{\psi}_i = 0$ and $\tilde{\varphi}_i = \frac{\varphi_{i-1}+\varphi_i+\varphi_{i+1}}{3}$.

Shape constraint: We set $\tilde{\varphi}_i = \varphi_i^0$ and $\tilde{\psi}_i = \psi_i^0$ where φ_i^0 and ψ_i^0 are two constants corresponding to the reference shape.

References

1. Nina Amenta, Marshall Bern, and Manolis Kamvysselis. A new voronoi-based surface reconstruction algorithm. In Michael Cohen, editor, SIGGRAPH 98 Conference Proceedings, Annual Conference Series, pages 415–422. ACM SIGGRAPH, Addison Wesley, July 1998. ISBN 0-89791-999-8.
2. M. Bern and D. Eppstein. Mesh generation and optimal triangulation. In D. Du and F. Hwang, editors, Computing in Euclidean Geometry, volume 1, pages 23–90. World Scientific Publishing Co, 1991.
3. J.D. Boissonnat and B. Geiger. Three dimensional reconstruction of complex shapes based on the delaunay triangulation. In R.S. Acharya and D.B. Goldgof, editors, SPIE Conference on Biomedical Image Processing and Biomedical Visualization, volume 1905, San Jose, CA, February 1993.
4. E. Boyer. Reconstruction et régularisation de la surface d'objets courbes. In Conference sur la Reconnaissance des formes et intelligence artificielle (RFIA'96), volume 1, pages 23–32, Rennes, France, January 1996.
5. C. Brechbuhler, G. Gerig, and O. Kubler. Surface parameterization and shape description. In Richard A. Robb, editor, Visualisation in Biomedical Computing, volume Proc. SPIE 1808, pages 80–89, 1992.
6. Y. Chen and G. Medioni. Surface description of complex objects from multiple range images. In Proc. of Int. Conf. on Computer Vision and Pattern Recognition (CVPR'94), pages 153–158, Seattle, USA, June 1994.
7. I. Cohen, L.D. Cohen, and N. Ayache. Using deformable surfaces to segment 3-d images and infer differential structures. Computer Vision, Graphics, and Image Processing: Image Understanding, pages 242–263, 1992.
8. L. Cohen. On active contour models and balloons. Computer Vision, Graphics, and Image Processing: Image Understanding, 53(2):21–218, March 1991.
9. L. Cohen and I. Cohen. A finite element method applied to new active contour models and 3d reconstruction from cross sections. Technical Report 1245, INRIA, June 1990.
10. B. Curless and M. Levoy. A volumetric method for building complex models from range images. In Computer Graphics (SIGGRAPH'96), 1996.
11. H. Delingette. Simplex meshes: a general representation for 3d shape reconstruction. Technical Report 2214, INRIA, March 1994.
12. H. Delingette. Simplex meshes: a general representation for 3d shape reconstruction. In Proc. of Int. Conf. on Computer Vision and Pattern Recognition (CVPR'94), pages 856–857, Seattle, USA, June 1994.
13. H. Delingette, M. Hebert, and K. Ikeuchi. Shape representation and image segmentation using deformable surfaces. In IEEE Computer Vision and Pattern Recognition (CVPR91), pages 467–472, June 1991.
14. F. Devernay and O. Faugeras. From projective to euclidian reconstruction. In International Conference on Computer Vision and Pattern Recognition (CVPR'96), San Francisco, 1996.
15. M. Eck and Hugues Hoppe. Automatic reconstruction of b-spline surfaces of arbitrary topological type. In Computer Graphics (SIGGRAPH'96), 1996.
16. H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, and W. Stuetzle. Piecewise smooth surface reconstruction. In Computer Graphics (SIGGRAPH'94), pages 295–302, Orlando, USA, 1994.
17. E. Koh, D. Metaxas, and N. Badler. Hierarchical shape representation using locally adaptive finite elements. In Jan-Olof Eklundh, editor, 3rd European Conference on Computer Vision (ECCV'94), volume vol 1, pages 441–446, Stockholm, 1994.
18. F. Leitner. Segmentation Dynamique d'images tridimensionnelles. PhD thesis, Institut National Polytechnique de Grenoble, Grenoble, France, September 1993.
19. C. Loop and T. DeRose. Generalized b-spline surfaces of arbitrary topology. In Computer Graphics (SIGGRAPH'90), volume 24, pages 347–356, 1990.
20. W. Lorensen and H.E. Cline. Marching cubes: a high resolution 3d surface construction algorithm. ACM Computer Graphics (SIGGRAPH'87), 21:163–169, 1987.
21. R. Malladi, J. Sethian, and B. Vemuri. Shape modeling with front propagation: a level set approach. IEEE Transactions on Pattern Analysis and Machine Intelligence, 17(2):158–175, 1995.
22. J.-L. Mallet. Discrete smooth interpolation. ACM Transaction on Graphics, 8(2):121–144, April 1989.
23. D. McInerney, T. Terzopoulos. A finite element model for 3d shape reconstruction and nonrigid motion tracking. In Proc. of the Fourth Int. Conf. on Computer Vision (ICCV'93), pages 518–523, 1993.
24. T. McInerney and D. Terzopoulos. Medical image segmentation using topologically adaptable snakes. In N. Ayache, editor, Computer Vision, Virtual Reality and Robotics in Medicine, pages 92–101, 1995.
25. T. McInerney and D. Terzopoulos. Deformable models in medical image analysis. Journal of Medical Image Analysis, Vol 1(No 2):91–108, 1996.
26. H. P. Moreton and C. H. Sequin. Functional optimization for fair surface design. In Computer Graphics (SIGGRAPH'92), pages 167–176, July 1992.
27. S. Muraki. Volumetric shape description of range data using "blobby model". In Computer Graphics (SIGGRAPH'91), pages 227–235, 1991.
28. C. Nastar and N. Ayache. Fast segmentation, tracking and analysis of deformable objects. In Proc. of the Fourth Int. Conf. on Computer Vision (ICCV'93), pages 275–279, May 1993.

29. J. Peters. Constructing C1 Surfaces of arbitrary topology using biquadratic and bicubic splines, pages 277–293. SIAM, 1994.
30. U. Pinkall and K. Polthier. Computing discrete minimal surfaces and their conjugates. Experimental Mathematics, 2(1):15–36, 1993.
31. F. P. Preparata and M. I. Shamos. Computational Geometry: an introduction Texts and monographs in computer science. Springer-Verlag, 1990.
32. P. Rushenas. Etude et Implantation d'un systeme de modelisation geometrique multidimensionnelle pour la conception assistee par ordinateur. PhD thesis, Universite de Montpellier II, July 1991.
33. Jos Stam. Exact evaluation of catmull-clark subdivision surfaces at arbitrary parameter values. In Michael Cohen, editor, SIGGRAPH 98 Conference Proceedings, Annual Conference Series, pages 395–404. ACM SIGGRAPH, Addison Wesley, July 1998. ISBN 0-89791-999-8.
34. Y. Suenaga and Y. Watanabe. A method for the synchronized acquisition of cylindrical range and color data. IEICE Transactions, E 74(10):3407–3416, October 1991.
35. R. Szeliski and D. Tonnesen. Surface modeling with oriented particle systems. In Computer Graphics (SIGGRAPH'92), pages 185–194, July 1992.
36. G. Taubin. Curve and surface smoothing without shrinkage. In Proceedings of the Fifth International Conference on Computer Vision (ICCV'95), pages 852–857, Boston, USA, June 1995.
37. G. Taubin, F. Cukierman, S. Sullivan, J. Ponce, and D.J. Kriegman. Parameterizing and fitting bounded algebraic curves and surfaces. In Proc. of Int. Conf. on Computer Vision and Pattern Recognition (CVPR'92), 1992.
38. N. Tsingos, O. Bittar, and MP. Gascuel. Semi-automatic reconstruction of implicit surfaces for medical applications. In Computer Graphics International (CGI'95), Leeds, June 1995.
39. M. Vasilescu and D. Terzopoulos. Adaptative meshes and shells. In Proc. of Int. Conf. on Computer Vision and Pattern Recognition (CVPR'92), pages 829–832, 1992.
40. B. Vemuri and R. Malladi. Constructing intrinsic parameters with active models for invariant surface reconstruction. IEEE Transactions on Pattern Analysis and Machine Intelligence, pages 668–681, 1993.
41. W. Welch and A. Witkin. Free-form shape design using triangulated surfaces. In Computer Graphics (SIGGRAPH'94), pages 247–256, Orlando, USA, July 1994. ACM.