INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

General Object Reconstruction based on Simplex Meshes

Hervé Delingette

N° 3111

Février 1997

THÈME 3

Rapport de recherche

ISSN 0249-6399

General Object Reconstruction based on Simplex Meshes

Hervé Delingette *

Thème 3 — Interaction homme-machine,
images, données, connaissances
Projet Epidaure **

Rapport de recherche n° 3111 — Février 1997 — 60 pages

Abstract:
In this paper, we propose a general tridimensional reconstruction algorithm of range and volumetric images, based on deformable simplex meshes. The algorithm is able to reconstruct surfaces without any restriction on their shape or topology. The different tasks performed during the reconstruction include the segmentation of objects in the scene, the extrapolation of missing data and the control of smoothness, density and geometric quality of the reconstructed model. All surfaces are represented as simplex meshes, that are unstructured meshes whose topology is dual of triangulations. The reconstruction takes place in two stages. First, we initialize the model either manually or using an automatic initialization routine. After the first fit, the topology of the model can be modified by creating holes or increasing its genus. Finally, an iterative adaptation or refinement algorithm decreases the distance of the model from the data while preserving a high geometric and topological quality. We have applied our algorithm to several medical images or range images.

Key-words: image segmentation, deformable models, 3D reconstruction, Medical Imaging, Range Images

(Résumé : tsvp)

* E-mail: Herve.Delingette@sophia.inria.fr
** http://www.inria.fr/Equipes/EPIDAURE-eng.html

Unité de recherche INRIA Sophia Antipolis
2004 route des Lucioles, BP 93, 06902 SOPHIA ANTIPOLIS Cedex (France)
Téléphone : 04 93 65 77 77 – Télécopie : 04 93 65 77 65

Reconstruction de scènes complexes à l'aide de maillages simplexes

Résumé : Dans ce rapport de recherche, nous proposons un algorithme de reconstruction tridimensionnelle utilisant les maillages simplexes déformables. Cet algorithme permet de reconstruire des surfaces à partir d'images de profondeur ou d'images volumiques, sans restriction sur leur géométrie ou leur topologie. Pour effectuer la reconstruction, notre algorithme permet de segmenter des objets dans une scène, d'extrapoler des données manquantes et de contrôler la régularité, la densité et la qualité du maillage. Toutes les surfaces sont représentées à l'aide de maillages simplexes qui sont des maillages non-réguliers dont la topologie est duale de celle des triangulations. La reconstruction s'effectue en deux étapes. Dans un premier temps, le maillage est initialisé manuellement ou grâce à une procedure automatique. Après avoir deformé ce maillage initial, la topologie du maillage est adaptée à celle de l'objet en créant des trous ou en augmentant le genre du maillage. Dans un deuxième temps, un algorithme d'adaptation ou de raffinement permet de diminuer la distance du maillage aux données tout en conservant de bonne qualités géométriques et topologiques. Nous proposons plusieurs exemples de reconstructions à partir d'images médicales ou de données de profondeur.

Mots-clé : segmentation d'images, modèles déformables, reconstruction tridimensionnelle, imagerie médicale

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 three dimensional dataset obtained from a scanning device. The resolution and accuracy of those tridimensional datasets may vary substantially and they may be stored as an ordered set of points, as a two-dimensional range image or as a volumetric image. In all cases, the reconstruction intend to create a geometric model corresponding to the object scanned by the acquisition devices. The required property of this geometric model is dependent on the type of high level task to be performed a posteriori, such as visualization, object recognition or computation of physical properties (such as mechanical or flow analysis). For instance, visualization requires geometric models with few vertices whereas CAD requires a model that closely fits the original dataset. Figure 1 displays the general scheme of object reconstruction.

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

The recovery of objects from tridimensional datasets requires to carry out several tasks including :

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

Filtering : the datasets always include some level of noise. It is therefore often needed to smooth the geometric model in order to decrease the effect of that noise. Smoothing should be done in an ad hoc manner, otherwise salient parts of the object such as sharp edges could be removed.

Extrapolation of missing data : in many cases, range images do not completely describe a given object. This is because the object is seen from a given viewpoint and because the emiter and receptor of the acquisition device are not localized at the same point thus creating "shadow areas". It is then necessary to handle those missing data points by extrapolating the surface in the most "intuitive" manner.

Control of mesh density : when dense dataset are provided, it is often desirable, especially for visualization purposes, to reduce the amount of information stored in the

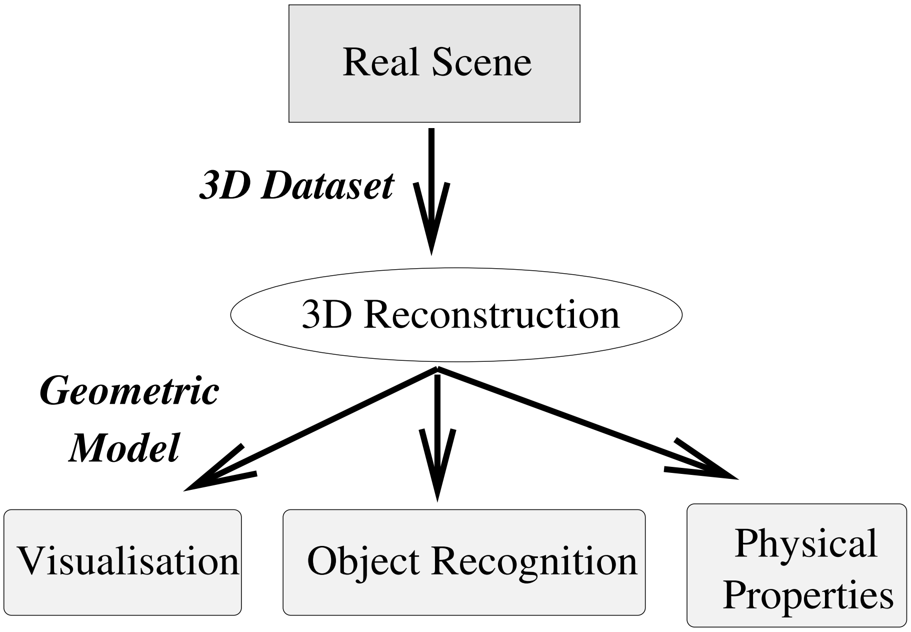


Figure 1: The general reconstruction scheme

geometric model, by a large amount. On the contrary, when sparse data is provided, it is necessary to refine the model to increase the density of vertices. Therefore, it is important to control the amount of information stored in the reconstructed model. An intuitive algorithm consists in linking the level of simplification or refinement to local geometric quantities such as curvature or area.

Control of mesh quality : the stability of computation of physical properties on a geometric model (resistance against deformation, heat,..) requires a mesh of high geometric quality. The quality of a mesh may be defined by several manners [BE91]. On triangulated models, the geometric quality may be measured with the minimum, median or average angle of the set of triangles. The topological quality of a triangulation may be measured by the average number of vertices around each vertex of the triangulation.

Few existing algorithms may be considered as "general reconstruction" techniques. For the reconstruction from volumetric images, two algorithms are widely used. First, isosurface extraction, often based on the Marching Cubes algorithm [LC87] consists in creating a mesh corresponding to a given isovalue. The technique may also be used for the reconstruction from dense range images [CL96]. However, because it does not allow the control of the mesh density and quality, it cannot be considered as a general reconstruction technique.

Delaunay tetrahedrisation and serial slice reconstruction [BG93] are widely used as well but cannot interpolate missing data points since they do not incorporate any smoothing mechanism. Furthermore, those algorithms require that data points must be segmented beforehand.

Deformable models are well-suited for general object reconstruction because they make little assumption about the shape to recover and they can filter and reconstruct objets from various types of datasets. Several formal frameworks of deformable models exist, but a common approach consists in formalizing the deformation as a variational problem involving an internal energy that enforces some continuity constraints, and an external energy controlling the closeness of fit. The trade-off between internal and external energy governs the behavior of deformable models and therefore their ability to recover the shape of an object even in presence of noise and outliers. Many researchers have proposed reconstruction systems based on deformable models [CCA92, DHI91, VM93, McI93, CM94] (for a survey of deformable models in medical image analysis see [IT96]). However, few systems address the problem of general surface reconstruction, in particular the five tasks listed in this paragraph.

1.2 Deformable Models

A necessary condition for proposing a general reconstruction system based on deformable models, is that no restriction exists on the geometry or the topology of the reconstructed objects. Therefore, it is of high importance that the surface representation associated with a deformable model, is as general as possible.

In this section, we review most of the different representations of deformable models and then explain our approach based on a new surface representation.

1.3 The Parameterization Problem

Most reconstruction systems are based on parametric representations, such as splines or finite elements. A parametric representation consists in a continuous transformation between the parameter space Ω embedded in the Euclidean plane $\mathbb{R}^2$ and the surface of that object. The parametric representation have several advantages over unstructured meshes. First, the shape of the object is defined everywhere. This feature is often required in high level tasks such as visualization or CAD design. Second, it allows the stable computation of geometric entities such as normal orientation or curvatures which is more problematic with unstructured meshes.

However, parametric-based reconstruction mainly suffers from two problems. The first one is that they are not well suited to represent general 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, with zero genus and no holes, at least one degenerate point or "pole" is created

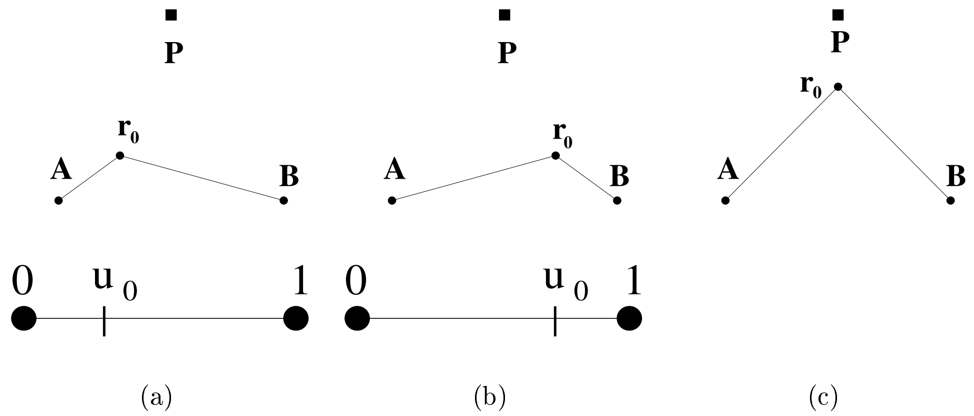


Figure 2: The parameterization problem : (a) The curve minimizing the approximation functional of equation 1 with $u_0 = 0.2$ and $\lambda = 2.0$; (b) same as (a) with $u_0 = 0.8$ (c) The curve minimizing the intrinsic approximation functional of equation 2 with $\lambda = 2.0$.

by the mapping of a plane onto a sphere. This point entails many problems in the deformation of that surface because at that vertex, normal and curvature cannot be computed in a stable manner. A lot of work have been done in the field of computer aided geometric design to overcome those topological problems. A popular approach consists in sewing several parametric patches and by ensuring that proper geometric continuity is realized between patches. For instance Hoppe and Eck [EH96] use the construction scheme of Peters [J94] to build G^1 continuous surfaces of arbitrary topology while other reconstruction systems such as Loop and De Rose [LD90] rely on different patch corners such as Sabin nets. An important constraint in order to efficiently represent 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 [Lei93].

The second problem lies in the nature of the parameterization that greatly influences the deformation of the model and therefore its final shape. The influence of the parameterization originates from the nature of fairness functionals that are usually defined in terms of parametric-dependent entities such as partial derivatives. This is the case for instance of the widely used, *thin plate* functional, that is based on first order derivatives. Because, deformation is constrained by parametric-dependent functionals, the nature of the deformation is dependent on the nature of the parameterization.

In figure 2, we show a simple example demonstrating the influence of parameterization on approximation problems. In this case, we are minimizing the following approximation problem :

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

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, ie independent of parameterization, the solution of the approximation problem is no longer dependent of the choice of u_0. In figure 2(a), we use the length of the curve as the fairness functional :

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

Details on the computational aspect of those approximation problems are provided in appendix A. This example reveals the trade-off in choosing a parameter dependent or 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 guaranty a high level of smoothness but lead to complex nonlinear numerical problems.

In most research papers, little attention is paid to the errors introduced in approximation problems by metric distortions between the parameter space and the surface. This is mainly because of the limitation to objects of simple topology (planar or cylindrical). However, Eck and Hoppe [EH96] have addressed this reparameterization problem with the use of harmonic maps. Brechbuhler[BGK92] 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 because of the computation 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 [IT95] defined topologically adaptable snakes through reparameterization on regular simplicial domain. Maladi and Vemuri [MSV95] propose to deform a two dimensional and tridimensional implicit model by propagating front based on the formulation of Osher and Sethian. Muraki [Mur91] and Gascuel *et al.*[TBG95] proposes implicit models based on radial functions to fit range or medical data. Taubin *et al.*[TCSP92] fits algebraic curves and surfaces of high degree on range data. Curless and Levoy [CL96] use distance maps to build a single model from multiple range images. If implicit model recovery has a lot of advantages in terms of flexibility, it does not seem to be well suited for tridimensional models because of the computation cost of reparameterizing at each iteration.

Other surface representations include spherical surfaces based on Fourier descriptors [SD92], modal analysis [PS91], or principal component analysis [CTLC93].

Unstructured meshes where no underlying parameterization are defined have been used as deformable surface representation. The problem with using unstructured meshes lies in the definition of a stable and meaningful internal energy or internal force. Vasilescu and Terzopoulos [VT92] use non-linear springs and masses on triangulated meshes for the reconstruction of range images. If this representation allows 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 to vertices, the system becomes underconstrained and several rest shapes are possibles. On the contrary, if too many springs are attached to vertices, the system is overconstrained 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 toward the center of its neighbors. Taubin [Tau95] reduces the shrinkage of Gaussian smoothing by applying a low-pass filter. Recently, Boyer [Boy96] proposed to minimize locally the area of the triangles adjacent to a vertex. Mallet [Mal89] uses a smoothness energy on a triangulation defined as a quadratic form of the neighboring vertex positions. Those three internal energies provide intuitive deformation, but correspond only to stabilizers of degree 1 and therefore tends to flatten the curved parts of the triangulation. Welch and Witkin [WW94] 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 refinement or large deformation.

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 triangulations. However, unlike triangulations, we can easily control the shape of simplex meshes through the control of discrete geometric entities such as curvature or normal vectors. In fact, those meshes are handled in a similar fashion than a system of springs and masses but with a higher level of shape control that include several degrees of continuity constraint and the existence of deformable contours on deformable surfaces. Simplex meshes are subject to internal forces computed locally that do not require any intensive computation.

Furthermore, topological operators are defined to locally refine a simplex mesh or change its genus, in an efficient manner. Because simplex meshes are not parametric models, we do not need to update a rigidity matrix or the parameterization mapping when performing those local transformations.

In this paper, we present a shape recovery system based on this surface representation. The high flexibility of simplex meshes, allows to model complex shapes even in presence of noisy data or outliers. In particular, we discuss the problem of the initialization of a simplex mesh, that is often considered as the main drawback of deformable models. Finally, we present a general algorithm for controlling both the topology and the geometry of a model. The number of holes or the genus of a simplex mesh may be changed to match the topology of datasets. Furthermore, the level of refinement and adaptation is governed by the minimization of a geometric criterion based on distance to the data or local curvature.

The outline of the paper is as follows. In section 2, we briefly introduce the notion of simplex mesh both in terms of topology as well as geometry. In section 3, we introduce the internal and external forces applied on those 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 5, we present several examples of shape recovery and conclude in section 6.

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 a complete exposé can be found in [Del94].

2.1 Definition of Simplex Meshes

The definitions of simplex meshes and triangulations are closely related. More precisely, their underlying graph are dual of each other. Another important property of simplex meshes, is their constant connectivity between vertices. In this section, we will only address the topological properties of simplex meshes, that exists independently of the space of embedding. We first give the general definition of a k-simplex mesh embedded in a Euclidean space $\mathbb{R}^d$ of dimension d. Then, we only consider 1 and 2-simplex meshes of $\mathbb{R}^3$ as representations of surfaces and contours.

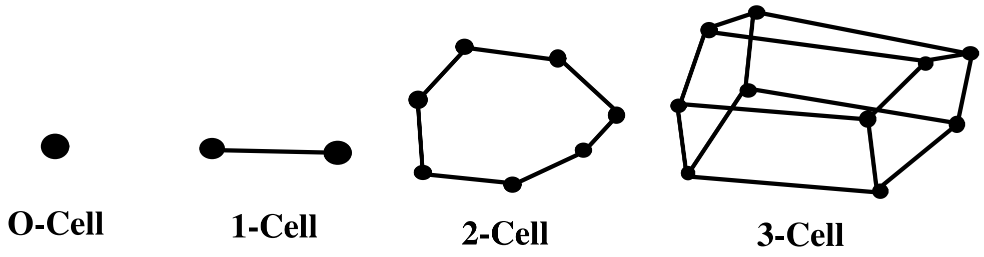


Figure 3: Examples of p-cells $0 \leq p \leq 3$.

2.1.1 Definition of Cells and Simplex Mesh

We define a k-simplex as a union of p-cells. Since those 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* ≥ 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.*
3. *Given two vertices of* $\mathcal{C}$*, there exists a path that links the two vertices.*

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 $\mathbb{R}^d$. The third condition, ensures that a p-cell is simply connected. Examples of p-cells are shown in figure 3. 0-cell are called *vertices*, 1-cell *edges* and 2-cells *faces*.

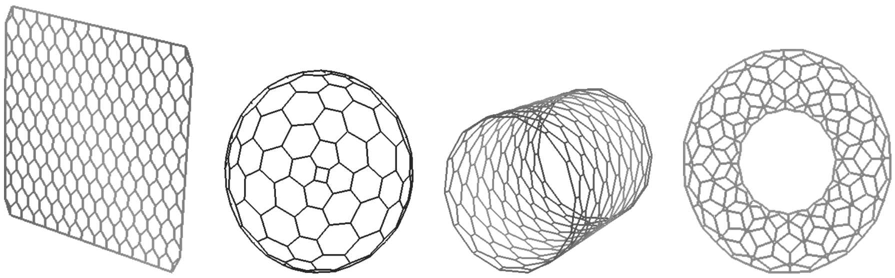
A simplex mesh is simply defined as:

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.

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

The constant connectivity between vertices implies the simple relation between the number of vertices and the number of edges. Table 1 summarizes the existing connectivity between vertices, edges, faces and cells of a k-simplex mesh.



(a) Plane : Genus = 0 Hole = 1

(b) Sphere : Genus = 0 Hole = 0

(c) Cylinder : Genus = 0 Hole = 2

(d) Tore : Genus = 1 Hole = 0

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

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

Table 1: The connectivity relations 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, in figure 5, we show a 3-connected mesh that cannot be a 2-simplex mesh since it has two faces intersecting along 2 edges.

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

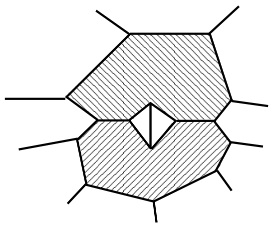


Figure 5: This mesh is 3-connected but cannot be decomposed in terms of a union of cells because two faces intersect along 2 edges.

2.1.2 Duality with Triangulations

It is important to stress the dual nature between k-simplex meshes and k-triangulations. *k-triangulations* also called *k-manifolds* [Rus91], are sets of p-simplices[1] $(0 < p \leq k)$ that follow strict topological rules such as the Euler-Poincaré relation. k-triangulations are actually a subset of more general sets of p-simplices that are called *k-simplicial complexes*.

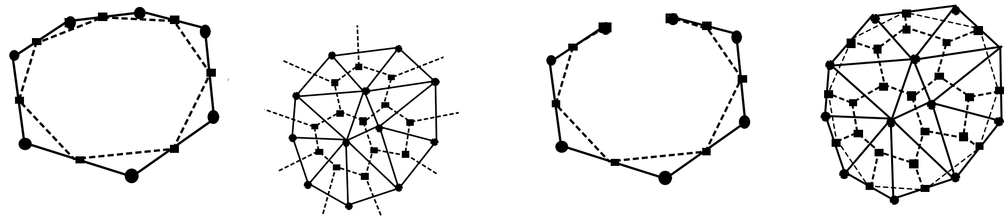
A k-triangulation is made of p-simplices $(0 \leq p \leq k)$ which are called the *p-faces* of the triangulation. The 0-faces are *vertices*, 1-faces the *edges* and 2-faces *triangles*.

We can define a topological transformation that associates a k-simplex mesh to a k-triangulation. This transformation is pictured in figure 6 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 simplex mesh.

Table 2 summarizes the transformation between a p-face of a k-triangulation and a p-cell of a k-simplex mesh. For cells or faces that belong to the boundary of a triangulation or a simplex mesh, the dual transformation applies differently. Vertices, edges and triangles that belong to the boundary of a k-triangulation 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).

This difference of symmetry in the duality between triangulations and simplex meshes originates from the fact that unlike simplex meshes, triangulations include the notion of

[1] a p-simplex is a set of $p+1$ independent points



(a) (b) (c) (d)

Figure 6: Duality between 1, 2-triangulations (drawn in solid lines and with circles) and the 1, 2-simplex meshes (drawn in dashed lines and with squares). The first two figures correspond to triangulations without boundaries while the last two ones consider the existing boundaries.

holes : a triangle is not necessarily 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.

	1−Tr $\Longleftrightarrow$ 1−SM	2−Tr $\Longleftrightarrow$ 2−SM
$p=0$	Vertex $\Longleftrightarrow$ Edge	Vertex $\Longleftrightarrow$ Face
$p=1$	Edge $\Longleftrightarrow$ Vertex	Edge $\Longleftrightarrow$ Edge
$p=2$		Triangle $\Longleftrightarrow$ Vertex

Table 2: Duality between a k-triangulation and a k-simplex mesh without consideration of the boundaries.

	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 $\Longrightarrow$ Vertex	Vertex $\Longrightarrow$ Face Vertex $\Longrightarrow$ Edge	Vertex $\Longrightarrow$ (nil)	Vertex $\Longrightarrow$ (nil)
$p=1$		Edge $\Longrightarrow$ Edge Edge $\Longrightarrow$ Vertex		Edge $\Longrightarrow$ (nil)

Table 3: Duality between a k-triangulation $\mathcal{T}$ and a k-simplex mesh $\mathcal{M}$ for faces that belong to the boundary of $\mathcal{T}$ and $\mathcal{M}$.

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*[Boi84].

Voronoï diagrams are cellular complexes and the duality relation with Delaunay triangulation is geometric because there exist a geometric transformation that associate one with the other. On the contrary, the duality between triangulations and simplex meshes is only in terms of topology. In another words, there are no geometric bijections existing between simplex meshes and triangulations.

The non-existence of geometric duality can be proved 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 holes, the Euler relation gives :

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

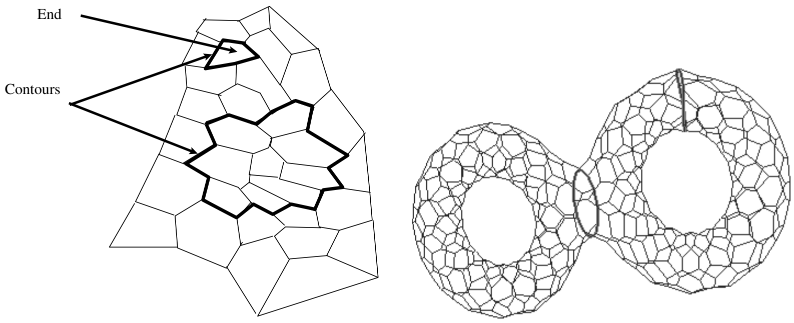
Therefore, we cannot build an geometric homeomorphism between triangulations and simplex meshes since the vectorial space representing their space of possible configurations have different dimensions. Only for 1-simplex meshes, is it possible to build a geometric dual 1-triangulation since they have 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 non existence of a geometric transformation between simplex meshes and triangulation 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 distinct surface representation from triangulations.

2.1.3 Contours

In order to define holes in a tridimensional surface, we introduce the notion of contours. Contours defined on 2-simplex meshes, are 1-simplex meshes, ie closed polygonal curves. They are simply defined as a set of neighboring vertices such that a vertex on a contour has two and only two neighbors that belong to that contour (see figure 7(a)). Contours can be defined around any face of a mesh. In particular, we will always create contours around holes, i.e. "empty faces". Contours do not always divide a mesh into two parts if the mesh has a genus greater than zero (see figure 7(b)).

Contours are considered as deformable models moving independently from the mesh. The surface mesh is thus attached to contour vertices that set the boundary conditions for the mesh deformation. This framework allows the deformation of meshes with an arbitrary number of holes.



(a) (b)

Figure 7: (a) Definition of contours in a 2-simplex mesh; (b) Two contours defined on a mesh on genus two.

2.1.4 Mesh Transformations

Simplex meshes similarly to 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.

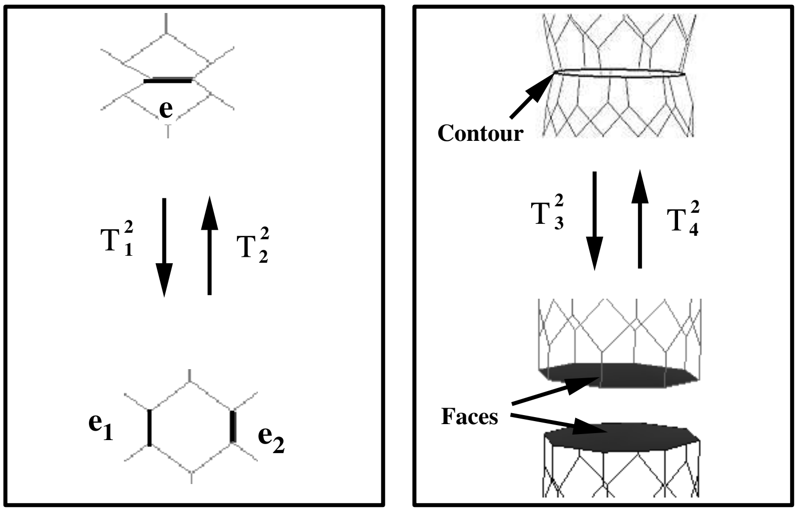
We define four basic topological operators acting on a simplex mesh, $T_1^2, T_2^2, T_3^2, T_4^2$, described in figure 8. The first two operators, are *Eulerian* because they do not modify the mesh topology. On the other hand, T_3^2 and T_4^2 are *meta-operators* because they can break a mesh into two pieces, or change its genus (number of surface handles).

All operations on a mesh can be decomposed into a set of those four operators. For instance, the refinement algorithm described in section 4.4.2, makes use of operator T_2^2 as well as the edge-swap operator that can be decomposed in terms of T_1^2 and T_2^2 (see figure 9).

Unlike most deformable models developed so far, it is possible with simple operations to merge two meshes or to break a mesh into two pieces as well as to change the genus of a mesh. This property provides a high topological flexibility of those deformable models.

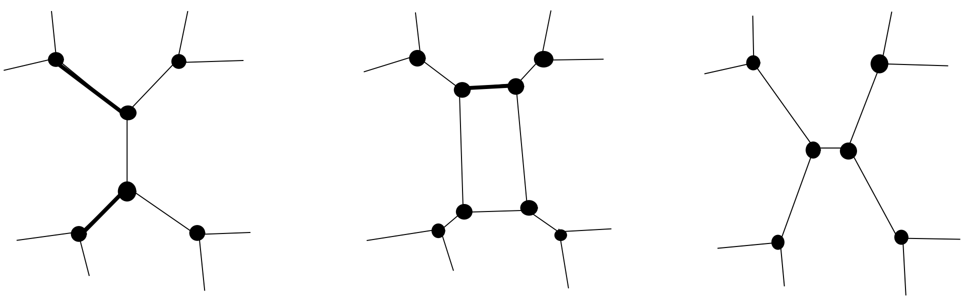
2.2 Geometry of simplex meshes

In this section, we present the main geometric relations existing in a 1-simplex mesh of $\mathbb{R}^2$ and a 2-simplex mesh of $\mathbb{R}^3$. The main result consists in a simple equation (equation 3 and equation 12) giving the position of a vertex as function relatively to its neighbors and two geometric entities : the *simplex angle* and the *metric parameters*. Those two entities are used to control the shape of simplex meshes as described in section 3.1.



(a) (b)

Figure 8: (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.



(a) (b) (c)

Figure 9: T_7^2 consists in swapping an edge.It can be decomposed into one operator T_2^2 followed by one operator T_1^2.

2.2.1 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 10). We define as well r_i as the half distance between P_{i-1} and P_{i+1} : $r_i = \|P_{i-1}P_{i+1}\|/2$.

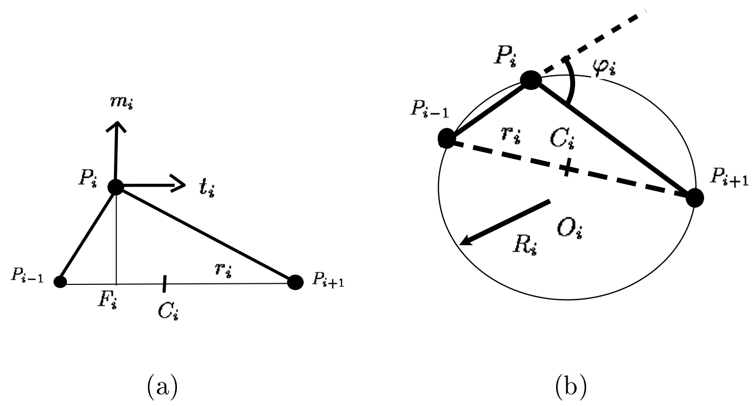


Figure 10: (a) Definition of tangent and normal vector around a vertex P_i of a 1-simplex mesh. (b) Definition of curvature κ_i and circumscribed circle.

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, $(\epsilon_i^1, \epsilon_i^2)$ are defined as the barycentric coordinates of F_i relative to P_{i-1} and P_{i+1} :*

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

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 $[P_iP_{i+1}]$.*

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 $(\epsilon_i^1, \epsilon_i^1, \varphi_i)$:

$$P_i = \epsilon_i^1 P_{i-1} + \epsilon_i^2 P_{i+1} + L(r_i, |2\epsilon_i^1 - 1| r_i, \varphi_i)\mathbf{n}_i \tag{3}$$

where :

$$L(r_i, d_i, \varphi_i) = \frac{(r_i^2 - d_i^2)\tan(\varphi_i)}{\epsilon\sqrt{r_i^2 + (r_i^2 - d_i^2)\tan^2(\varphi_i)} + r_i} \tag{4}$$

and

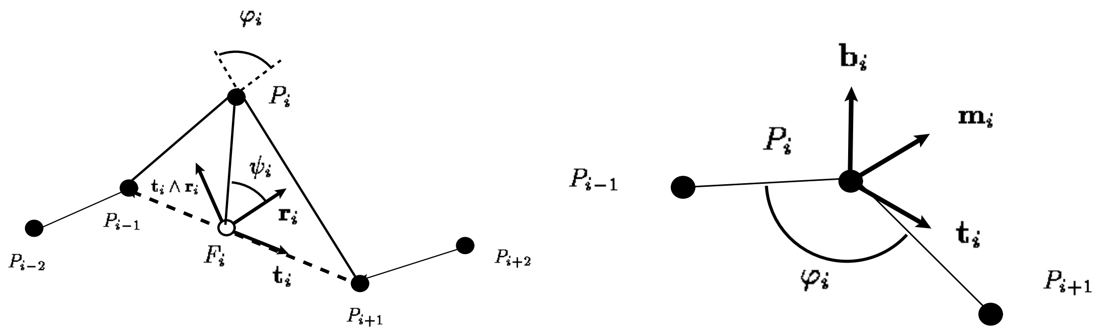
$$\begin{array}{l} \epsilon = 1 \quad if \quad |\varphi_i| < \pi/2 \\ \epsilon = -1 \quad if \quad |\varphi_i| > \pi/2 \end{array}$$

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

2.2.2 Geometry of tridimensional contours

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 $\mathbf{b}_i$:

$$\mathbf{t}_i = \frac{P_{i-1}P_{i+1}}{\|P_{i-1}P_{i+1}\|} \quad \mathbf{n}_i = \mathbf{b}_i \wedge \mathbf{t}_i \quad \mathbf{b}_i = \frac{P_{i-1}P_i \wedge P_iP_{i+1}}{\|P_{i-1}P_i \wedge P_iP_{i+1}\|} \tag{5}$$



(a) (b)

Figure 11: (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 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 $\mathbf{r}_i$:

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

($\mathbf{r}_i$ 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 $\mathbf{r}_i$:

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

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

$$P_i = \epsilon_i^1 P_{i-1} + \epsilon_i^2 P_{i+1} + L(r_i, d_i, \varphi_i)(\cos(\psi_i)\mathbf{r}_i + \sin(\psi_i)\mathbf{t}_i \wedge \mathbf{r}_i) \tag{6}$$

2.2.3 Geometry of tridimensional 2-simplex meshes

On a tridimensional 2-simplex mesh $\mathcal{M} \in \mathbb{R}^3$, we define a normal vector $\mathbf{n}_i$ as :

$$\mathbf{n}_i = \frac{P_{N_1(i)} \wedge P_{N_2(i)} + P_{N_2(i)} \wedge P_{N_3(i)} + P_{N_3(i)} \wedge P_{N_1(i)}}{\|P_{N_1(i)} \wedge P_{N_2(i)} + P_{N_2(i)} \wedge P_{N_3(i)} + P_{N_3(i)} \wedge P_{N_1(i)}\|} \tag{7}$$

The definition of the simplex angle generalizes the definition of section 2.2.1.

We introduce 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 with the two equations :

$$\begin{array}{c} \varphi_i \in [-\pi, \pi] : \\ \sin(\varphi_i) = \frac{r_i}{R_i} \text{ signe}(P_i P_{N_1(i)} \cdot \mathbf{n}_i) \\ \cos(\varphi_i) = \frac{\|O_i C_i\|}{R_i} \text{ signe}(O_i C_i \cdot \mathbf{n}_i) \end{array} \tag{8}$$

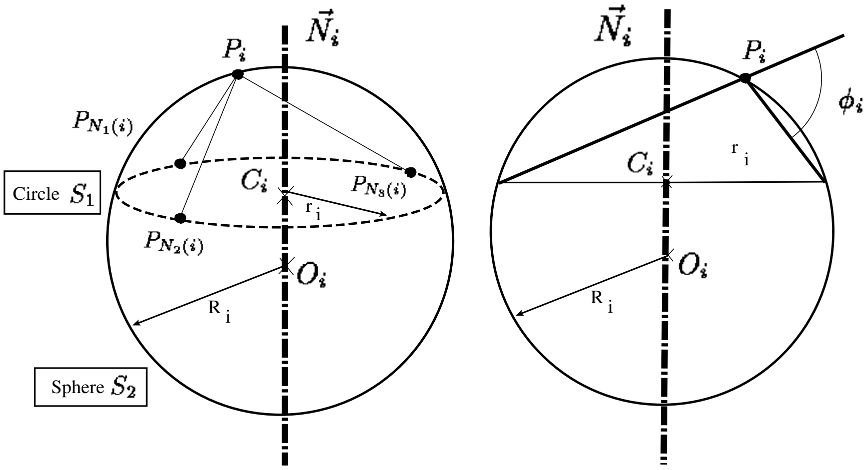
The simplex angle has many original properties that are developed in [Del94]. It is important to note that there is a simple relation between the simplex angle φ_i and the curvature $H_i = \frac{1}{R_i}$, also called the *mean curvature* at vertex P_i :

$$H_i = \frac{\varphi_i}{r_i} \tag{9}$$

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 $(P_{N_1(i)}, P_{N_2(i)}, P_{N_3(i)})$:

$$F_i = \epsilon_i^1 P_{N_1(i)} + \epsilon_i^2 P_{N_2(i)} + \epsilon_i^3 P_{N_3(i)} \tag{10}$$

$$\epsilon_i^1 + \epsilon_i^2 + \epsilon_i^3 = 1 \tag{11}$$



(a) (b)

Figure 12: (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 with two metric parameters (since the three metric parameters are linked with equation 11) represent the position of P_i with respect to its three neighbors :

$$P_i = \epsilon_i^1 P_{N_1(i)} + \epsilon_i^2 P_{N_2(i)} + \epsilon_i^3 P_{N_3(i)} + L(r_i, d_i, \varphi_i)\mathbf{n}_i \tag{12}$$

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 a function described in equation 4

We see that the geometry of tridimensional 2-simplex meshes is a straightforward generalization of the geometry of planar 1-simplex meshes.

3 Deformable Simplex Meshes

In this section, we describe a law of deformation for simplex meshes. Unlike parametric representations, the geometry of a mesh is only defined by the position of its vertices. The deformation of those meshes is therefore 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 simplex angle and 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 consisting of an internal and external force :

$$m\frac{d^2 P_i}{dt^2} = -\gamma \frac{dP_i}{dt} + \mathbf{F}_{int} + \mathbf{F}_{ext} \tag{13}$$

where m is the mass unit of a vertex and γ is the damping factor. $\mathbf{F}_{int}$ is the internal force that enforces the continuity of the shape of the mesh. $\mathbf{F}_{ext}$ is the external force that constrains the mesh to be close to some tridimensional dataset.

We discretized the time t in order to compute the evolution of the simplex mesh under the law of motion 13. Using central finite differences with an explicit scheme, the law of motion is discretized as :

$$P_i^{t+1} = P_i^t + (1-\gamma)(P_i^t - P_i^{t-1}) + \mathbf{F}_{int} + \mathbf{F}_{ext} \tag{14}$$

Both forces $\mathbf{F}_{int}$ and $\mathbf{F}_{ext}$ are computed at time t. With equation 14, the internal and external forces have the dimension of a distance.

3.1 Internal Force Computation

The internal force acting on simplex meshes is computed based on the minimization of a local criterion. In that respect, it can be compared to the deformation of particle systems such as those introduced by Szelisky [ST92] or springs systems. The internal force is derived from the minimization of a local energy that is proportional to the deflection from some ideal position. Global smoothness functionals have been used mostly with parametric models such as in [McI93, CCA92]. Those global functionals provide an objective evaluation of the goodness of a solution as well as a well-posed mathematical framework. However, in practise, the minimization is transformed into the iterative application of an internal force, similarly to our approach.

Unlike springs and masses systems, the smoothness constraints applied on simplex meshes range from position continuity constraints to curvature continuity constraints with a control over the scale at which the smoothness applies. Examples in figure 14 show that perfect continuity across patches can be achieved with simplex meshes, independently of topology. Another interesting property is the ability to have different continuity constraint on different parts of a mesh.

The local internal energy $\mathcal{S}_i$ at vertex P_i is proportional to the square distance between P_i and the target position $P_i^\star$:

$$\mathcal{S}_i = \frac{\alpha_i}{2} P_i P_i^{\star 2} \tag{15}$$

In another words, $\mathcal{S}_i$ is the deflexion of P_i to some ideal position. The internal force is then :

$$\mathbf{F}_{int} = \frac{\partial \mathcal{S}_i}{\partial P_i} = \alpha_i P_i P_i^\star$$

where

- α_i is a scalar $(0 < \alpha < 0.5)$ that weights the internal constraint with respect to external constraint. Values of α_i over 0.5 result in unstable deformations.
- $P_i^\star$ is defined with respect to the neighbors of P_i with the simplex angle $\varphi_i^\star$ and the metric parameters $(\epsilon_i^{1\star}, \epsilon_i^{2\star}, \epsilon_i^{3\star})$

Using equation 12, the internal force, may be written as :

$$\mathbf{F}_{int} = \alpha_i(\epsilon_i^{1\star} P_i P_{N_1(i)} + \epsilon_i^{2\star} P_i P_{N_2(i)} + \epsilon_i^{3\star} P_i P_{N_3(i)} + L(r_i, d_i, \varphi_i^\star)\mathbf{n}_i) \tag{16}$$

The metric parameters $(\epsilon_i^{1\star}, \epsilon_i^{2\star}, \epsilon_i^{3\star})$ are constants that are imposed during the deformation. The local adaptation algorithm optimizes those metric parameters in order to concentrates vertices around parts of high curvature (see section 4.4.1). In all cases, in order to ensure a stable deformation, those metric parameters must be strictly positive.

The simplex angle $\varphi_i^\star$ may not be constant over times depending on the type of regularity constraint that should be enforced. We list four regularity constraints that may be enforced. More explanations are provided in [Del94].

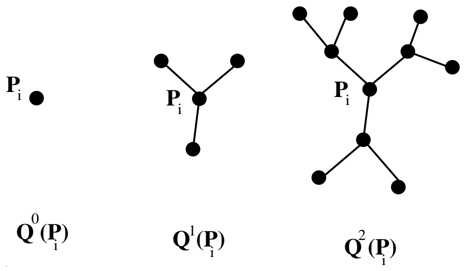


Figure 13: Description of a neighborhood $Q^{s_i}(P_i)$ around a vertex P_i.

Position Continuity Constraint We set $\varphi_i^\star = \varphi_i$, where φ_i is the current value of the simplex angle. The surface can freely bend around vertex P_i without any constraint on the value of the simplex angle. This is useful to represent surface orientation discontinuities.

Surface Orientation Continuity constraint We have simply $\varphi_i^\star = 0$. Hence, the internal force writes as: $\mathbf{F}_{int} = \alpha_i(\epsilon_{1i}P_{N_1(i)} + \epsilon_{2i}P_{N_2(i)} + \epsilon_{3i}P_{N_3(i)} - P_i)$. When metric parameters are equal to 1/3, we have $\mathbf{F}_{int} = \frac{\alpha_i}{3}(P_{N_1(i)}+P_{N_2(i)}+P_{N_3(i)}-3P_i)$. The vertex P_i is attracted towards the center of its neighbors. Continuity of normal orientation is guaranteed but not the continuity of simplex angle or mean curvature.

Shape Constraint Given the constant φ_i^0 by setting $\varphi_i^\star = \varphi_i^0$ we constrain the simplex angle at P_i to φ_i^0. The mesh is deformed towards some reference shape described by the simplex angle φ_i^0 and metric parameters $(\epsilon_i^{1\star}, \epsilon_i^{2\star}, \epsilon_i^{3\star})$.

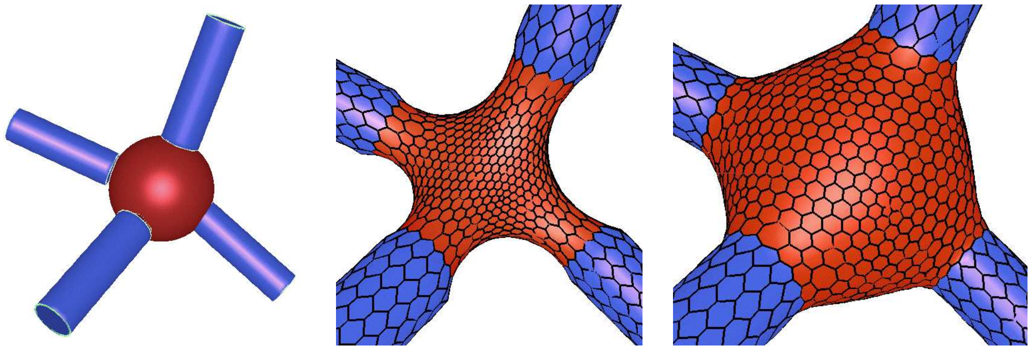
Simplex Angle Continuity Constraint $\varphi_i^\star$ is defined as :

$$\varphi_i^\star = \sum_{j \in Q^{s_i}(P_i)} \varphi_j \tag{17}$$

$\varphi_i^\star$ is the weighted average of the simplex angles on the neighborhood $Q^{s_i}(P_i)$ of size s_i around P_i. This neighborhood is defined in a recursive manner. We define $Q^0(P_i) = \{P_i\}$ and $Q^1(P_i) = \{P_i, P_{N_1(i)}, P_{N_2(i)}, P_{N_3(i)}\}$. The neighborhood $Q^{s_i}(P_i)$, $s_i > 1$ is the union of $Q^{s_i-1}(P_i)$ with the vertices of $\mathcal{M}$ that have a neighboring vertex in $Q^{s_i-1}(P_i)$ (see figure 13).

s_i corresponds intuitively to the notion of rigidity, or scale of deformation. On 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 on the surface, since the curvature is averaged over a large number of vertices. The rigidity has an effect of the dynamics of deformable simplex meshes, as well. 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.

Because the internal constraint of simplex meshes relies on *geometric* quantities instead of *parametric* quantities, a high level of visual smoothness can be achieved. For instance, in figure 14, we smooth a surface of complex topology, with four holes, with various types of continuity constraints.



(a) (b) (c)

Figure 14: Example of smoothness constraints applied on simplex meshes. We constraint a surface with four holes (figure (a)) to smoothly connect 4 cylinders with surface orientation continuity (figure (b)) and simplex angle continuity (figure (c)).

In most cases, we apply the *simplex angle continuity constraint* on deformable simplex meshes, because it ensures the highest order of regularity. We keep the constant α_i equal to 0.5 but we change the value of the rigidity parameter s_i during deformation.

3.2 Internal Force applied on contours.

Contours defined on simplex meshes are deformable as well, and vertices belonging to those contours follow the same law of motion than equation 13, but with a different expression of the internal force. The internal constraints are closely related to those of surface mesh, ranging from position continuity to curvature continuity. In most cases, we apply simplex angle continuity constraint on contours.

Let $\mathcal{C}$ be a contour defined on a simplex mesh $\mathcal{M}$. The shape at a vertex $P_{J(i)}$ is defined by the position of its four neighbors on the contour $P_{J(l)}$, $l = i-2, i-1, i+1, i+2$ and the angle values $\varphi_{J(i)}$ and $\psi_{J(i)}$. The regularizing force $\mathbf{F}_{int}$ corresponds to the minimization of the local criterion $\mathcal{S}_{J(i)}$:

$$\mathcal{S}_i = \frac{\alpha_{J(i)}}{2} \| P_{J(i)} P^{\star}_{J(i)} \|^2$$

The internal force is then proportional to the displacement vector joining $P_{J(i)}$ to the point $P^{\star}_{J(i)}$ determined by the two angles $\varphi^{\star}_{J(i)}$ and $\psi^{\star}_{J(i)}$:

$$\begin{aligned}
\mathbf{F}_{int} &= \alpha_{J(i)} P_{J(i)} P^{\star}_{J(i)} \\
\mathbf{F}_{int} &= \alpha_{J(i)} (\epsilon_{J(i)} P_{J(i)} P_{J(i-1)} + (1 - \epsilon_{J(i)}) P_{J(i)} P_{J(i+1)} + \\
&\quad \lambda^{\star}_{J(i)} (\cos(\psi^{\star}_{J(i)}) \mathbf{r}_{J(i)} + \sin(\psi^{\star}_{J(i)}) \mathbf{t}_{J(i)} \times \mathbf{r}_{J(i)})) \\
\text{with} \quad & \lambda^{\star}_{J(i)} = L(\frac{\| P_{J(i-1)} P_{J(i+1)} \|}{2}, (\epsilon_{J(i)} - \frac{1}{2}) \| P_{J(i-1)} P_{J(i+1)} \|, \varphi^{\star}_{J(i)})
\end{aligned}$$

$\epsilon_{J(i)}$ is the metric parameter and this parameter is fixed during the deformation. We define the following constraints on a contour vertex :

Position Continuity Constraint . We set $\varphi^{\star}_{J(i)} = \varphi_{J(i)}$ and $\psi^{\star}_{J(i)} = \psi_{J(i)}$ where $\varphi_{J(i)}$ and $\psi_{J(i)}$ are the current values of curvature and torsion.

Normal Continuity Constraint . We set $\varphi^{\star}_{J(i)} = 0$ and $\psi^{\star}_{J(i)} = 0$. The force then simply writes as $\vec{F}_{int} = \alpha_{J(i)} \Big(\epsilon_{J(i)} P_{J(i)} P_{J(i-1)} + (1 - \epsilon_{J(i)}) P_{J(i)} P_{J(i+1)} \Big)$.

Angle Continuity Constraint . We set $\psi^{\star}_{J(i)} = 0$ and $\varphi^{\star}_{J(i)} = \frac{\varphi_{J(i-1)} + \varphi_{J(i)} + \varphi_{J(i+1)}}{3}$. We can generalize this expression by averaging over a larger extent s_i:

$$\varphi^{\star}_{J(i)} = \frac{\sum_{i - s_i \leq j \leq i + s_i} \varphi_{J(j)}}{2 s_i + 1}$$

s_i is the rigidity coefficient that is similar to the simplex mesh case.

Shape constraint We set $\varphi^{\star}_{J(i)} = \varphi^{0}_{J(i)}$ and $\psi^{\star}_{J(i)} = \psi^{0}_{J(i)}$ where $\varphi^{0}_{J(i)}$ and $\psi^{0}_{J(i)}$ are two constants corresponding to the reference shape.

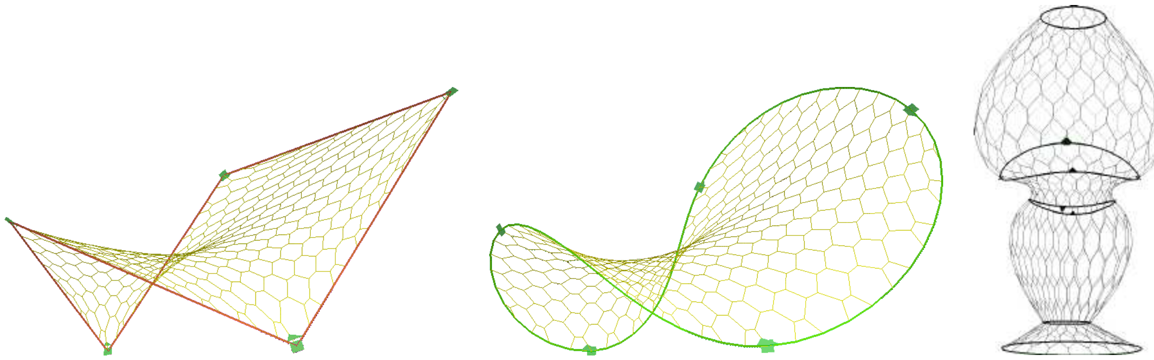
3.3 Boundary Conditions

Because contour vertices are deformed independently of mesh vertices, contours provide boundary conditions for the surface mesh. The boundary conditions are posed in terms of vertex position but additional conditions can be added. There are two types of surface-contour constraint:

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

Tangent Condition The angle between the normal at the mesh and the contour normal measured around the contour tangent vector, intuitively corresponds to the angle between the surface and the contour. The angle between the two normal vectors is controlled by the simplex angle, as computed on a 2-simplex mesh, at contour vertices.

In figure 15, we show examples of mesh and contour deformations. In the first two examples, a contour is constrained to approximate five fixed points with normal orientation and simplex angle constraint. In the last figure, we built a vase with five contours with appropriate simplex angle and tangent boundary conditions.



(a) (b) (c)

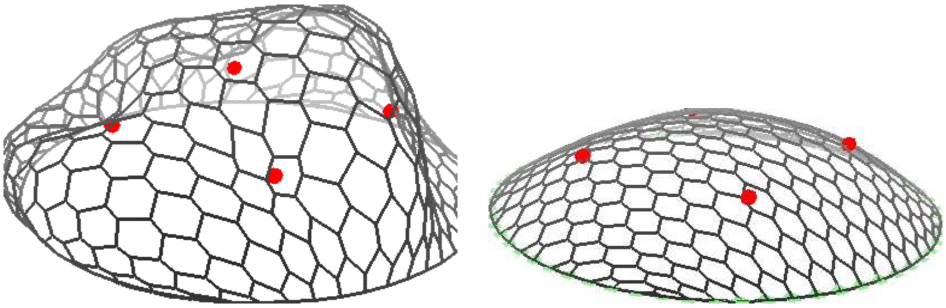
Figure 15: (a) A contour defined on a simplex mesh is submitted to surface orientation continuity constraints. The contour interpolates the position of five contour vertices displayed with black squares. The surface mesh has everywhere null mean curvature; (b) Same as (a) except that the contour is submitted to curvature continuity constraints. The continuity of surface orientation is then verified at each five fixed vertices;(c) A vase created from a cylinder and five contours with different end conditions;

3.4 External Force Computation

The external force $\mathbf{F}_{ext}$ is proportional to the distance of P_i to the dataset. The computed 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 guaranty a stable and smooth behavior over time. Indeed, tangential displacements could create vertices with negative metric parameters or even a mesh with self-intersections.

Second, the normal direction of the external force ensures the resulting shape of the mesh will be smooth even in the presence of sparse data. If the attraction force was directed along the direction joining the vertex P_i and the closest data, the mesh at equilibrium would be unstructured with non intuitive shape (see figure 16).



(a) (b)

Figure 16: (a) A simplex mesh is fit on sparse data represented with the 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. (b) The external force is directed along the normal direction resulting in a smooth shape.

The complexity of the computation of $\mathbf{F}_{ext}$ is linear in the number of mesh vertices. This is sharp contrast with previous work of Metaxas [KMB94] and Terzopoulos [McI93] where the complexity is linear with the number of data points since they compute the distance of the data points to the mesh. 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, and some algorithms are needed to decrease the size of the dataset. Finally, there exists no efficient algorithms for computing the true distance of a point to a triangulated mesh whereas computing the distance of a data point to the normal line can be done with efficiency.

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.

3.4.1 External Force Computation On Range Data

The computation of the external force is dependent on the notion of closest point. For every vertex P_i of the mesh, we search for the closest data point $M_{Cl(i)}$ and the force is then computed as :

$$\mathbf{F}_{ext} = \beta_i G\left(\frac{\|P_i M_{Cl(i)}\|}{D_{\mathrm{ref}}}\right)(P_i M_{Cl(i)} \cdot \mathbf{n}_i)\mathbf{n}_i \tag{18}$$

where β_i is a weight parameter, $\mathbf{n}_i$ is the normal vector of the mesh at P_i, $G(x)$ is the stiffness function (see figure 17) and D_{ref} is a reference distance.

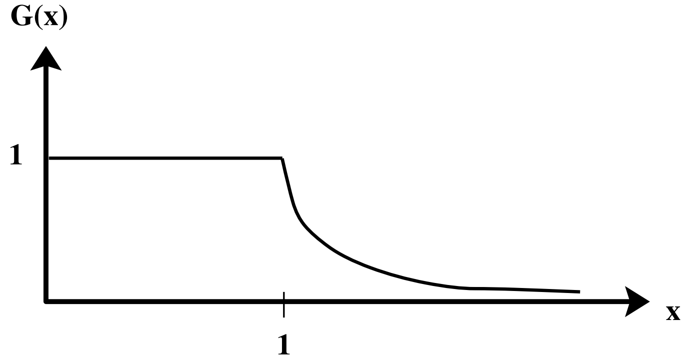


Figure 17: The stiffness function $G(x)$.

The force $\mathbf{F}_{ext}$ is computed as the projection of the vector $P_i M_{Cl(i)}$ along the normal direction. The distance D_{ref} is the maximum distance of attraction of a data point. The stiffness function $G(x)$ ensures that the force decreases sharply when the distance between $M_{Cl(i)}$ and P_i is greater than D_{ref}.

It is indeed very important to control the range of attraction of data points in order to limit the influence of outliers. This distance D_{ref} is computed relative to the overall size of the dataset, i.e. the radius of the sphere surrounding the dataset. During the first stage of the deformation, we pick 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 of the closest point.

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

Projection Method On dense range images extracted from triangulation principles [SW91] or stereovision [DF96], the search of 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 an method in $O(1)$, that approximates the search of 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 18). 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.

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.

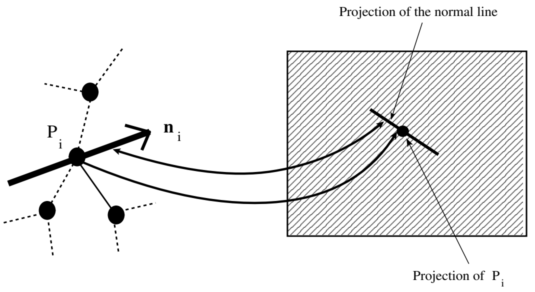


Figure 18: Search of the closest point on a range image

KD-tree When no calibration matrix is available, we compute the closest point with a *kd-tree* [PS90]. This data structure gives the closest point inside a sphere centered around P_i and of radius D_{ref}, at nearly constant time. Because of memory limitation, it is not often possible to store all data points inside the kd-tree. In the first stage of the deformation where D_{ref} is relatively large, we usually subsample the whole dataset in order to guaranty a fast computation of $\mathbf{F}_{ext}$. During the second stage, we discard all data points that are far away from the mesh, and store all 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.

KD-tree and Normal Orientation As for structured range data, when normal vectors are available for each data point, we use a six dimensional kd-tree combining position and orientation information, similarly to Feldmar [FA94].

3.4.2 External Force Computation On Volumetric Images

On volumetric images, the task of reconstruction usually consists in isolating regions of consistent intensity values. Therefore, gradient intensity is the main information on which is based the external force. As in [CCA92] and [McI93], we combine both gradient intensity and edge information for the computation of $\mathbf{F}_{ext}$:

$$\mathbf{F}_{ext} = \mathbf{F}_{grad} + \mathbf{F}_{edge} \tag{19}$$

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 maxima of gradient and entails larger deformations of the mesh. The edge image is a binary image extracted from the gradient intensity image through thresholding.

Previous works [McI93] has derived the computation of $\mathbf{F}_{grad}$ as the gradient of the potential field $||\nabla||^2$, thus generalizing the approach of active contours. However, this formulation has the drawback of creating oscillations of the deformable model across the voxels of high gradient, due to the choice of the time step and the space discretization. Cohen [CC90] has proposed to normalize the gradient vector in order to limit those oscillations within a pixel. However, it 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, of the voxel of maximum gradient intensity. If $\mathcal{V}$ is the voxel containing P_i, then we inspect all voxels in a $m \times m \times m$ window around $\mathcal{V}$ for the voxel center G_i, of highest gradient intensity (see figure 19(a)). The force expression is then :

$$\mathbf{F}_{grad} = \beta_i^{grad}(P_iG_i.\mathbf{n}_i)\mathbf{n}_i \tag{20}$$

β_i^{grad} is weighting parameter with $0 \leq \beta_i^{grad} \leq 1$. Since the force is proportional to the true deflection vector P_iG_i (and not the gradient vector), this force entails no oscillations of the mesh. The computation of this force is actually fast since we can precompute 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 and have a gradient normal that is consistent with the normal direction at P_i : $\nabla I(G_i).\mathbf{n}_i > 0$. This constraint is natural since the gradient is always directed inwards or outwards a region of constant intensity value.

- **intensity value .** The voxel G_i must have an intensity value in a given range value $[val_{min}, val_{max}]$. This constraints helps to discriminate against neighboring regions of high contrast with different intensity values.

The edge image is built by thresholding the gradient intensity values. 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 [CCA92, McI93, NA93]. 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, distance are ambiguous at voxels equidistant from two edge voxels. Our method uses the normal orientation at vertex P_i to raise this ambiguity.

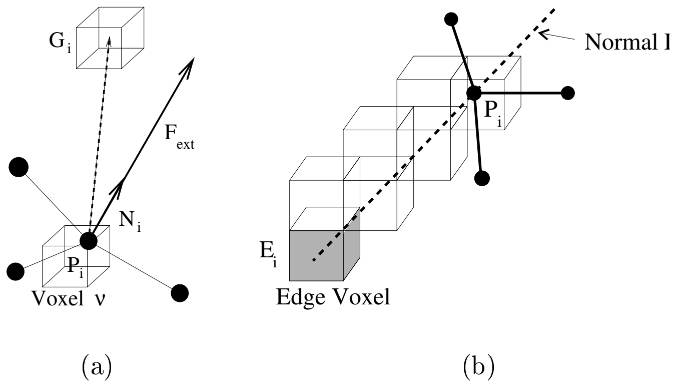


Figure 19: (a) Computation of the gradient force $\mathbf{F}_{gradient}$: G_i is the center of the voxel with the highest gradient intensity in the neighborhood of voxel ν; (b) Search of the closest edge voxel along the normal line for the computation of $\mathbf{F}_{edge}$.

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 19 (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 :

$$\mathbf{F}_{edge} = \beta_i^{edge} P_i E_i \tag{21}$$

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

Similarly to the gradient force, we can add constraints to the determination of the closest edge voxel :

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

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

4 Reconstruction Algorithm

4.1 Modeling Scheme

We propose a reconstruction scheme that is independent on the type of dataset and that requires little a priori knowledge about the scene to be reconstructed. The scheme is presented in figure 20 (a) and consists in two main stages.

The first stage corresponds to the deformation of the mesh from an initial position to a close approximation of the shape of the dataset. To initialize the model, the user can choose the mesh topology among three possible classes : spherical, cylindrical or planar. He can then either initialize manually the mesh around the dataset or called an automatic initialization algorithm described in section 4.2, if the dataset has few outliers.

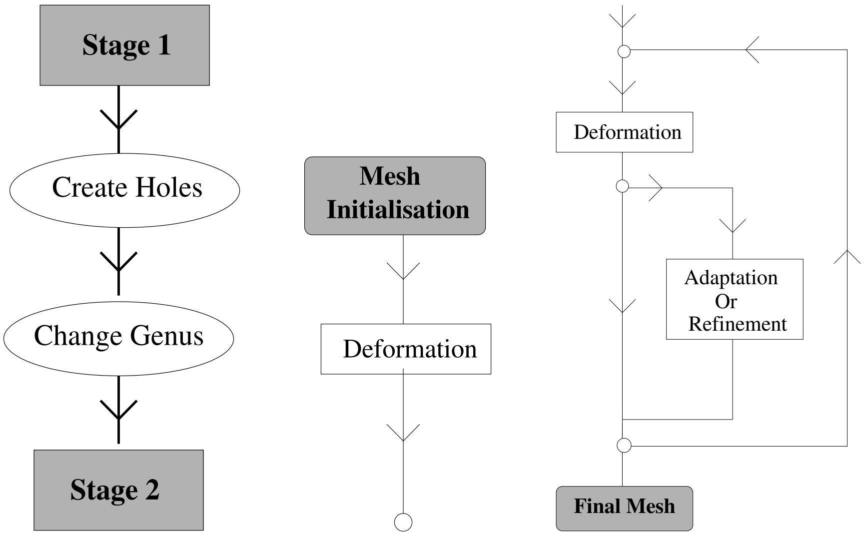
In all cases, the topology of the mesh can be changed after the first stage of deformation. The change of topology occurs first by creating holes in the mesh, and second by increasing the genus of the mesh as seen in the diagram 20.

Once the mesh has the desired topology, the second stage of deformation consists in providing a better geometric representation of the object. In order to control its distance to the data, the mesh may be refined or adapted. Refinement and adaptation are two complementary tasks that consist respectively in adding vertices and moving 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 have smooth and large scale deformations of the mesh. On the contrary, 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 we want to control the distance of the mesh to the data.

4.2 Initialization

One of the main problem of reconstruction systems based on deformable models, is that the recovered shapes depend on the initial position of the mesh. In general, those models need to be initialized closed to the data. When modeling complex shapes, manual initialization



(a) (b) (c)

Figure 20: (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 the deformation : the mesh is refined or adapted in order to create an optimal representation of the object.

may not give proper results, despite the ability to adapt the mesh topology to the topology of the dataset.

We propose an initialization algorithm that can only handle a limited number of topologies (three for 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 that enclose the whole dataset. To do so, we consider a tessellated unit sphere centered on G, and we project each data point or voxel onto that sphere. We store on each surface element on the sphere, the maximum distance of projected points from the center. When no points project on a sphere element, we use the mean value of the radius instead. The variance of the radius on the unit sphere determines the number of vertices of the initial simplex mesh.

For cylindrical meshes, we first compute the axis of minimum inertia. We then divide this axis into n segments. Finally, we project each data point on a segment and then compute for each segment the centroid and the largest distance along radial directions of all projected data points. We then fit a simplex mesh of cylindrical topology onto this generalized cylindrical surface. The mesh handles incomplete data because it interpolates the parts when no data is available. The variance of radius determines the number of vertices of the initial mesh.

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

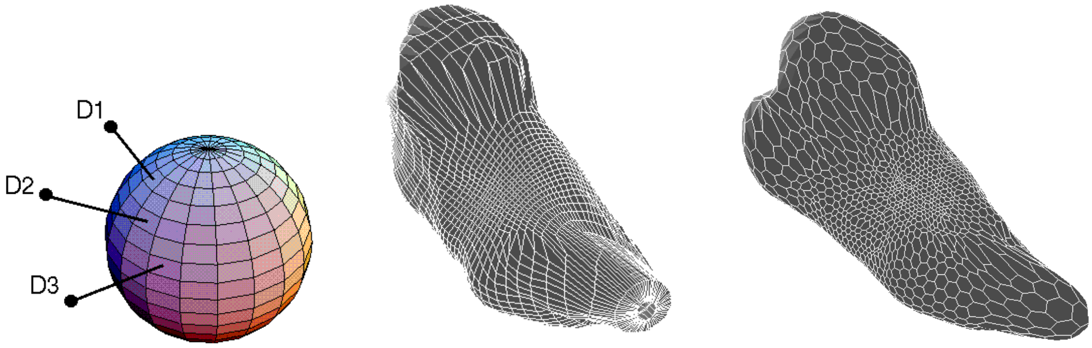
Examples of spherical, cylindrical and planar meshes is given at figures 21, 22 and 23.

4.3 Changing the mesh topology

After the initialization of a mesh, it may occur that the mesh does not have the same topology as the object. Since topology of tridimensional surfaces are characterized by their number of holes and their genus number, we propose two algorithms for changing those topological characteristics. Because initial meshes are, in general, topologically more simple than the object to recover, we only focus on building models with additional holes and increased genuses, though removing holes or decreasing genuses could be performed with our surface representation.

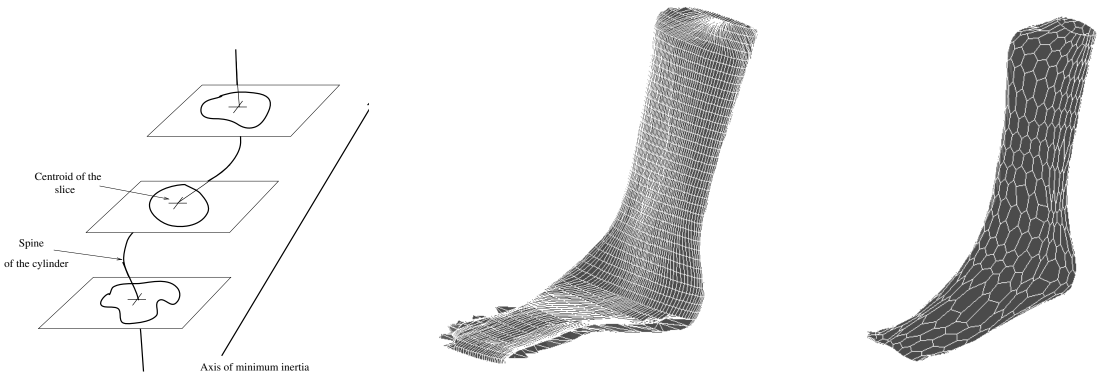
4.3.1 Creating Holes

The principle of this algorithm is to create holes in the mesh where no data points exist. We thus compute for each vertex, its distance from the closest data points and we label the



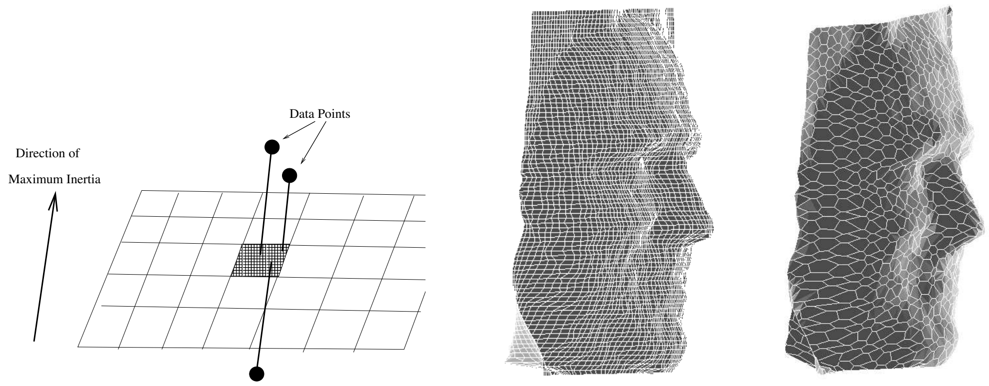
(a) (b) (c)

Figure 21: (a) The principle for initializing a spherical mesh : each data point is projected on a sphere; (b) The surface in spherical coordinates that encloses the range image of a foot; (c) The simplex mesh fit on the previous spherical surface;



(a) (b) (c)

Figure 22: (a) The axis of minimal inertia, the spine and the slices used for the initialization of a cylindrical mesh; (b) The surface in cylindrical coordinates that encloses the range image of a foot; (c) The simplex mesh fit on the previous cylindrical surface;



(a) (b) (c)

Figure 23: (a) The initialization of a planar mesh is based on the projection of data points on the plane of maximum inertia; (b) The surface in planar coordinates that averages the range image of a face; (c) The simplex mesh fit on the previous planar surface;

faces whose vertices have a distance greater than the reference distance D_{ref}. Those faces are gathered in sets of connected components called *zones*. Then with the help of the topological operator T_3^2, each zone is removed from the mesh and a contour is created around the hole. This contour after few iterations, will deform to closely fit the hole existing in the dataset.

Example of this algorithm is demonstrated for the reconstruction of a skull model (figure 24) and face model (figure 25). For the skull, holes has been created at the level of the orbits, the nose and the foramen. Other zones have not been cut due to their small size.

The example of the face shows why this procedure cannot be made fully automatic. Two zones have indeed been created : one around the upper part of the head and at the neck level. The zone at the top of the head corresponds to missing hair data due to the limitation of the acquisition technology. But the zone around the neck does not correspond to existing physical points and should be cut. Therefore, we can run the hole creation module in an interactive mode where the user can specify the zones to be cut based on his knowledge of the tridimensional scene.

4.3.2 Increasing the genus of a mesh

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, as initial shape, meshes with zero genus (sphere, cylinder or plane). It is therefore necessary to provide an algorithm that increases the genus of a mesh.

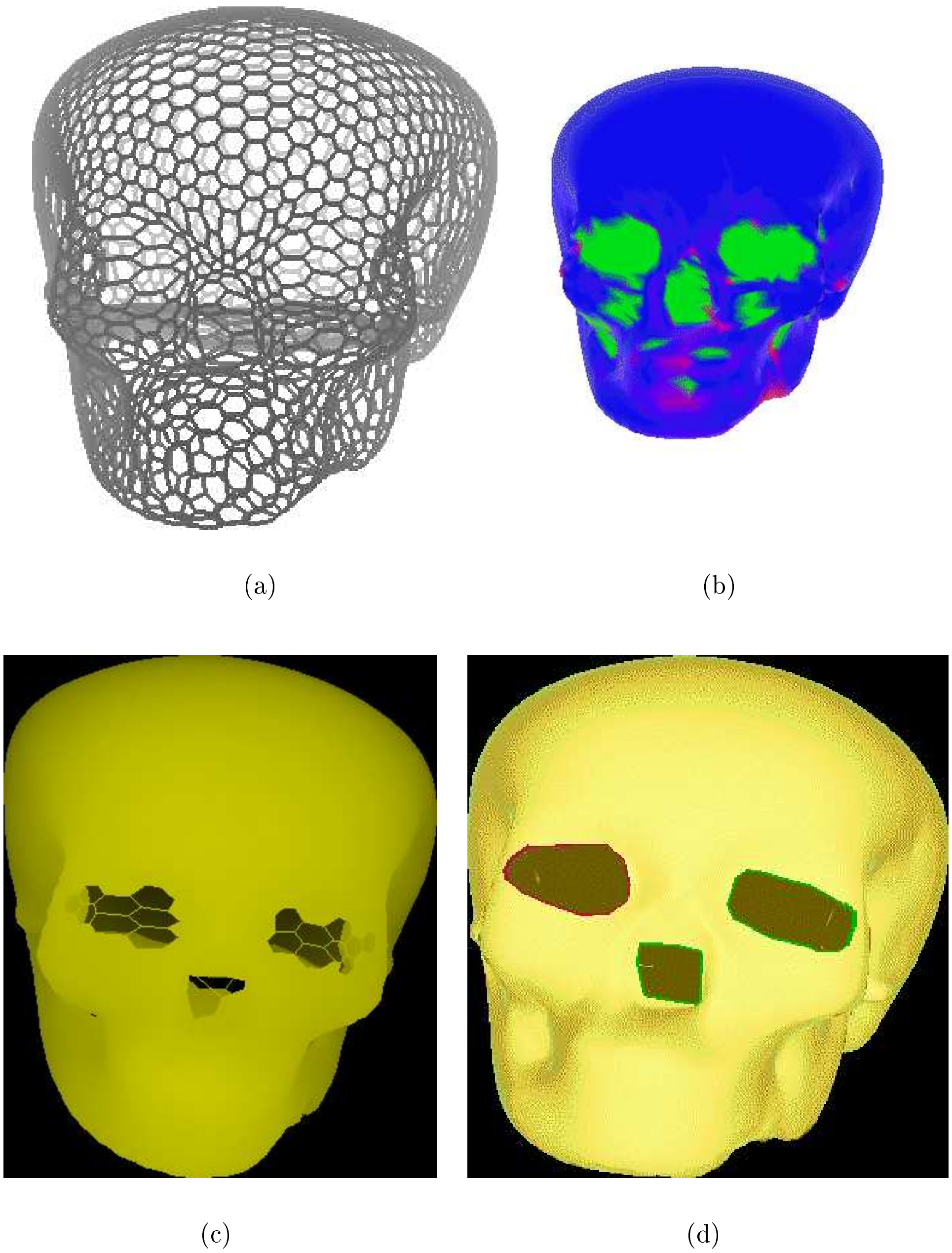
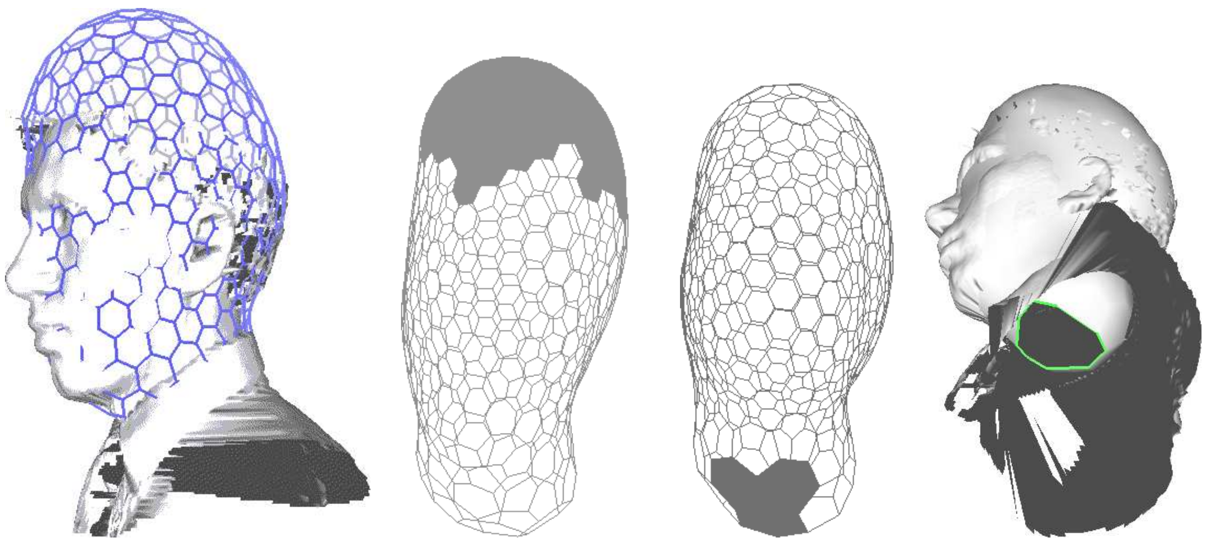


Figure 24: Creation of holes on a skull model : (a) Mesh after first stage; (b) Color coding of the distance of mesh vertices to the dataset; (c) Zones to be cut; (d) Mesh after creation of holes;



(a) (b) (c) (d)

Figure 25: Creation of holes on a face model : (a) Simplex mesh after the first stage; (b) The zone corresponding to the hair where the data was not acquired by the Cyberware scanner; (c) Zone corresponding to the neck; (d) Mesh after cutting the neck zone;

We propose an algorithm consisting in connecting, with the topological operator T_4^2, two contours surrounding two holes. In a simplex mesh, a hole is an “empty” face surrounded by a contour. Among all pair of contours surrounding holes, we choose those verifying the following criteria :

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

In figure 26, we initialize a mesh around a vertebra with the spherical topology. The algorithm described in the previous section, creates automatically two holes on each side of the handle. If two contours verify the three criteria, then they are merged and the mesh is subsequently refined to closely fit the model.

4.4 Controlling the closeness of fit

4.4.1 The adaptation algorithm

Because simplex meshes are discrete representation of surfaces, the goodness of fit of a mesh 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 metric parameters $\epsilon_i = \{\epsilon_{1i}, \epsilon_{2i}, \epsilon_{3i}\}$ control the relative distance of a vertex to its three neighboring vertices. Our algorithm updates the values of the metric parameters depending on the mean curvature values of neighboring vertices.

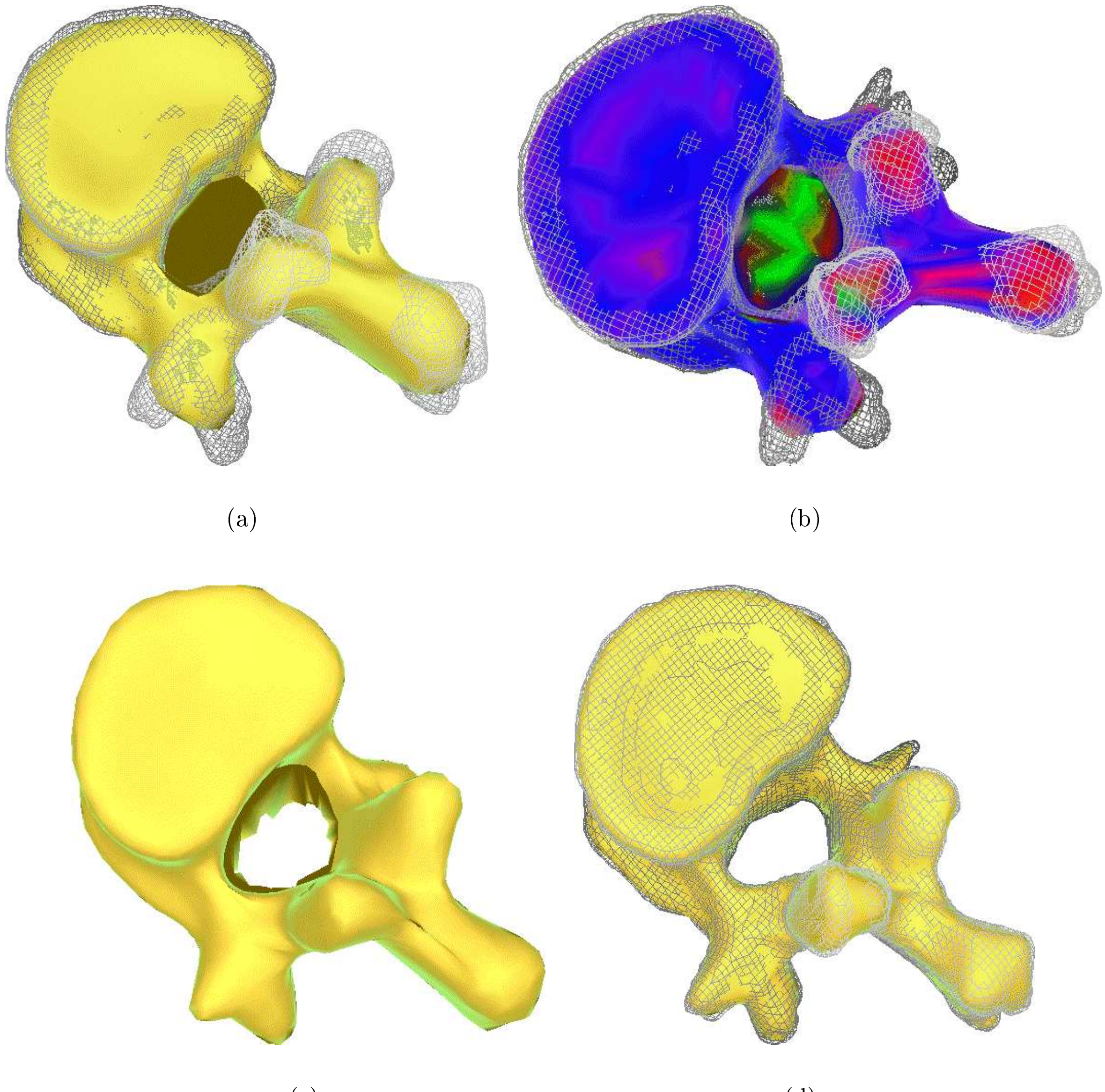
Basically, the algorithm operates as follows : vertices of low mean curvature migrate towards neighboring vertices of relatively larger mean curvature whereas vertices of high mean curvature have metric parameters close to $\frac{1}{3}$ in order to obtain a uniform concentration at highly curved parts.

The concentration of vertices is governed by the local minimization of an energy $\mathcal{E}_i$, function of the variation of mean curvature. We periodically update the metric parameters values : if ϵ_i^t is the value of the metric parameters at iteration t, then we compute the metric parameters at iteration $t+p$ as:

$$\epsilon_i^{t+p} = \epsilon_i^t + \frac{1}{2}\vec{\bigtriangledown}\mathcal{E}_i$$

The energy $\mathcal{E}_i$ is defined similarly to the energy $\mathcal{S}_i$ of equation 15 :

$$\mathcal{E}_i = \frac{1}{2}(\epsilon_i^\star - \epsilon_i)^2 \tag{22}$$



(a) (b) (c) (d)

Figure 26: (a) Deformation of a mesh initialized as a sphere on data representing a vertebra; (b) Color coding of the distance to the data; (c) The automatic hole creation algorithm generates two holes; (d) The two contours are merged and the mesh has a genus equal to one;

where $\epsilon_i^\star$ is computed as a function of the variation of the absolute value of the mean curvature. We derive the expression of $\epsilon_i^\star$ by first considering the mean value of the absolute mean curvature $|\bar{H}_i| = (|H_{N_1(i)}| + |H_{N_2(i)}| + |H_{N_3(i)}|)/3$. We then compute the relative mean curvature deviation vector $\delta|H|_i$:

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

We link the value of the reference metric parameter $\epsilon_i^\star$ with the relative mean curvature deviation vector:

$$\epsilon_i^\star = \frac{1}{3} + \gamma_i \delta|H|_i \tag{23}$$

γ_i is a constant that controls the extent of the adaptation of the simplex mesh and is usually chosen between 0.03 and 0.25. However, since we set bounds for metric parameters ϵ_{ki} (for instance between 0.05 and 0.833), we may compute a value of γ_i substantially lower than 0.03. Finally we have:

$$\epsilon_i^{t+p} = \epsilon_i^t + \frac{1}{2}(\epsilon_i^\star - \epsilon_i^{t+p}) \tag{24}$$

The weight 0.5 enables a smoother variation of the metric parameters over time. 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.

Figure 27 shows the effect of the mesh adaptation on a rounded cube. The metric parameters are initially equal to $\frac{1}{3}$. After the mesh adaptation, vertices nicely concentrates at highly curved parts, the level of concentration being controlled by the value of γ_i.

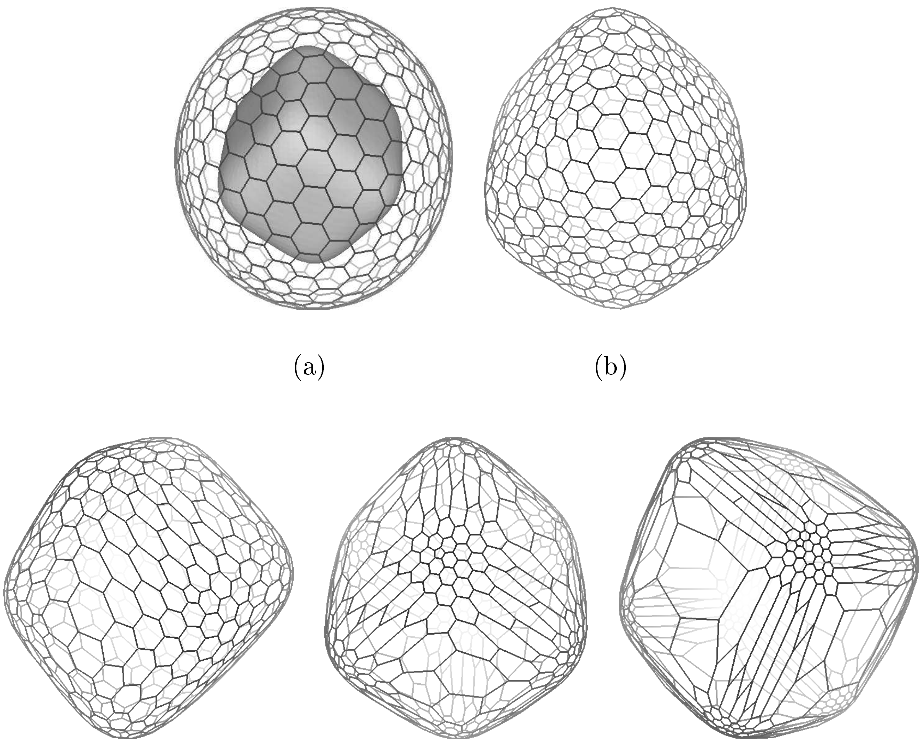
4.4.2 The Refinement Algorithm

Mesh adaptation optimally moves vertices toward parts of high curvature. However, the shape description is limited by the number of vertices of the initial mesh. We therefore introduce a method for refining a simplex mesh in order to control its distance to the original data.

We refine faces of a mesh according to several criteria including curvature, distance from a dataset, area and face elongation. The user may combine several criteria if desired.

For each face, we compute a refinement measure that describes whether the face is a good candidate for refinement. This refinement measure is centered around 1.0.

The refinement algorithm takes place in an iterative manner (see figure 28 (a)). We first look for the face having the maximum refinement measure. If the measure is above a given refinement threshold, we refine the face by creating a new edge with the topological operator T_2^2. The mesh is then locally smoothed around the newly created edge. In order to keep



(a) (b)

(c) (d) (e)

Figure 27: (a) The initial mesh with the isosurface; (b) the mesh fit on the isosurface. The metric parameters are everywhere equal to $\frac{1}{3}$; (c) The adaptive mesh with a value of $\gamma_i = 0.10$; (d) The adaptive mesh with a value of $\gamma_i = 0.15$; (e) The adaptive mesh with a value of $\gamma_i = 0.20$.

the number of vertices per face uniform on the mesh, we iteratively swap edges adjacent to large faces. Each time an edge is swapped, the mesh is locally smoothed around the swapped edge as well. Finally, we recompute the refinement measure for all faces whose vertices have moved. We iterate until no face has a refinement measure above the threshold.

Because the geometry of the mesh is modified only around the refined faces, the algorithm converges quickly and is not time consuming. Furthermore, it is important to swap edges adjacent to large faces in order to guaranty a mesh with high geometric properties (small distance to the dataset) as well as high topological properties (number of vertices per face close to 6).

The four criterion measures are computed as follows :

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

The elongation of a face : It is computed as 1 plus the difference of length between the longest edge and the smallest edge of a face, divided by the median value of the length of the face edges.

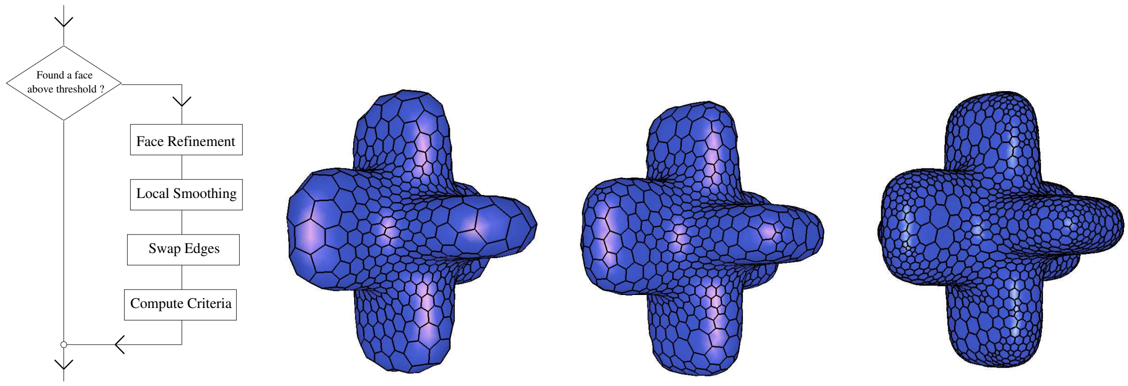
The measure of Gaussian curvature : It is evaluated through the computation of area of the spherical polygon described by the normal vector on the Gaussian sphere.

The relative distance to 3D data : It is based on the ratio of the distance to the closest data point by the radius D_{ref} of the corresponding range data.

The refinement threshold is dimensionless and it is therefore evaluated independently of the size of the mesh. The user can either set a threshold value or select coarse, medium or fine resolution. In this case, the threshold is automatically computed in order to refine 15%, 25% or 40% of all faces.

The refinement may not perform well in the presence of strong noise or outliers where the distance between the mesh vertices and the data is relatively high. At those parts, the refinement would result in actually fitting the noise. Therefore, we may only apply the refinement process at specified parts of a mesh.

As for the adaptation algorithm, the refinement algorithm is only performed at every p iterations. This allow the mesh vertices to nicely spread over the surface object. In general, we choose a value of $p = 5$.



(a) (b) (c) (d)

Figure 28: (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);

5 Results

5.1 Quantitative Results

5.1.1 Influence of the damping factor

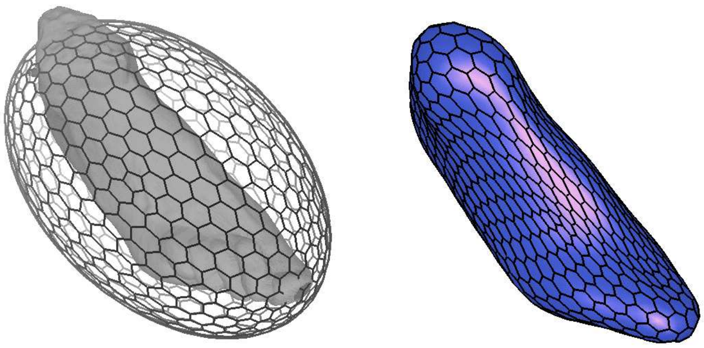
Our deformable model framework is based on a Newtonian law of motion (see equation 13), that includes the acceleration of each vertex and a damping factor in order to prevent oscillations of the system. Most deformable model systems, on the contrary, are based on a Lagrangian law of motion, where the speed of each vertex is equal to the sum of the internal and external forces.

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

$$P_i^{t+1} = P_i^t + (1-\gamma)(P_i^t - P_i^{t-1}) + \mathbf{F}_{int} + \mathbf{F}_{ext} \quad (25)$$

We note that if $\gamma = 1$, then equation 25 corresponds to a Lagrangian law of motion. As the damping factor γ decreases towards zero, the effect of the acceleration factor 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 simplex meshes on a range image of a foot initialized as an ellipsoid. The deformation is stopped after 30 iterations. The initial mesh position and the deformed mesh with a value of $\gamma = 0.80$ are shown in figure 29.



(a) (b)

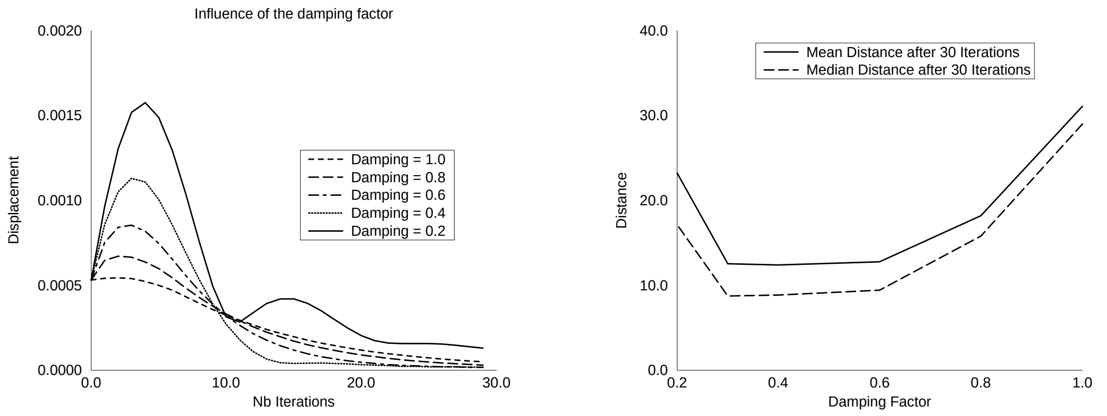
Figure 29: (a) Initialization of the mesh around the range data of a foot; (b) Mesh after 30 iterations with $\gamma = 0.80$

In figure 30, we have studied the displacement of vertices during 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 speed of convergence increases. However, when γ is smaller than 0.20, large oscillations prevent the mesh to converge towards the shape of the range data. For $\gamma = 0.20$, we observe that the resulting mesh self-intersects.

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 a 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 practise, we choose a value of $\gamma = 0.65$.

5.1.2 Evaluation of Performance of the Closest-Point Algorithms

We have introduced two methods for finding the closest data point from a given vertex. The *projection method* uses the calibration matrices of camera-based range-finders to search the closest point along the normal line. This method does not guaranty to 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.



(a) (b)

Figure 30: (a) Chart of the mean displacement of vertices as a function of the number of iterations for several values of the damping factor; (b) The mean and median distance to the range data for several values of the damping factor

	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

Figure 31: Accuracy and computational cost of the projection and kd-tree method.

In this section, we compare the accuracy and the computational efficiency of those two methods on three cases. The three meshes are shown respectively in figure 29 (a) and figure 32 (a) (b). They correspond to three stages of deformation with decreasing distance to the original data : after initialization, after the first deformation and after refinement.

Since the kd-tree method gives the true closest Euclidean distance to the data, we evaluate the accuracy of the projection method, by comparing for each vertex, the closest distance given by the projection method with the distance given by the kd-tree method. All distances are measured along the normal direction at a vertex, in order to remove the effect of sparse data. In table 31, we can see that the accuracy of the projection method, increases drastically as the mesh is closer to the range data. The vertices where the error remains large even on the refined mesh, correspond to 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 table 31. It appears that the efficiency of the kd-tree method is linked with the radius D_{ref} of search around the vertex and therefore with the distance between the mesh and the data. When the mesh is far away from the data, the kd-tree method is 35 times slower than the projection method whose computational cost is independent of D_{ref}. On the other hand, the kd-tree method is only 2.5 times more expensive. In all cases, the cost for building the kd-tree of the 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 not a problem because the stiffness parameter β_i is then very low ($\beta_i = 0.1$). It is therefore not necessary, in such cases, to compute with high accuracy the closest point.

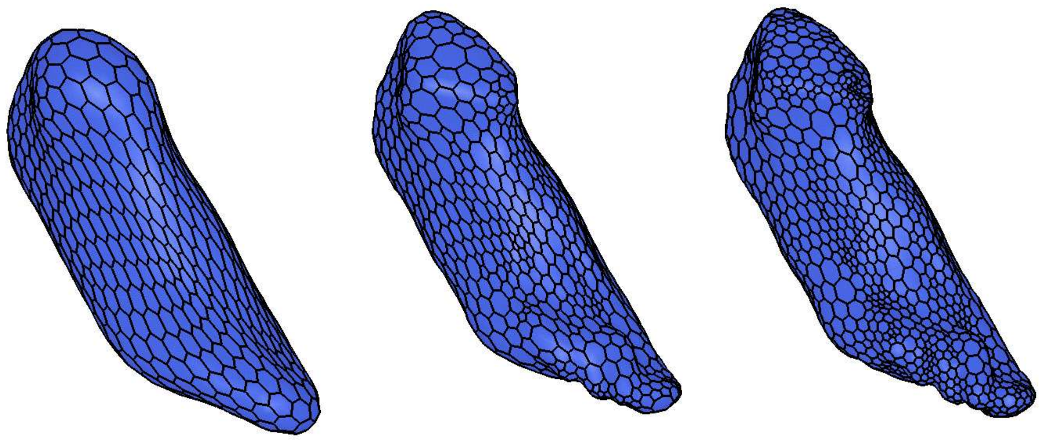
5.1.3 Effect of the Refinement on the Distance Error

In this section, we study the effect of refinement on the overall distance error of the mesh. Starting from a rough estimate shown in figure 29 (a), the mesh position after applying the first stage of deformation is shown in figure 32 (a). This model has 1280 vertices and oversmoothes the shape of the original data.

We refine this mesh based on the mesh distance to the data. In this example, we choose two different refinement levels set by two refinement measures. The first refinement measure has been automatically computed in order to refine 25% of the original number of faces while the second corresponds to a refinement level of 40%. Since the refinement measure is linked with the maximum distance to the data, to choose a refinement measure is equivalent to choose a maximum bound for that distance. The first refinement measure corresponds to a maximum distance of 22.10 mm while the second of 14.25 mm. Note that this distance to the data is computed not only for all mesh vertices, but also for the centers of all faces.

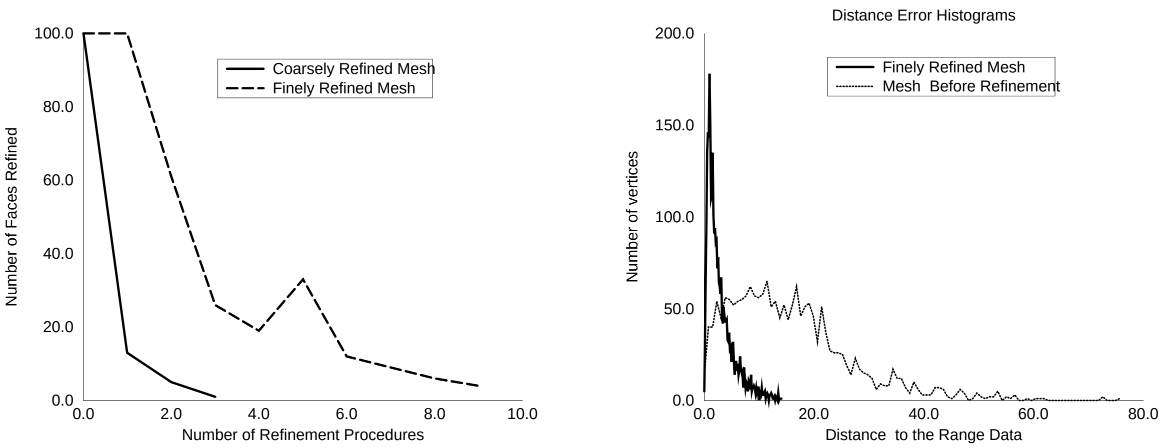
The meshes before and after refinement are shown in figure 32. Quantitative results about distance to the original data, are given in table 34. The refinement procedure is called every 5 iterations and stops when there are less than 5 faces to refine. We limit the number of faces refined per procedure to 100, in order to obtain a more homogeneous spacing of vertices. The number of faces refined per call to the refinement procedure is described in figure 33 (a). The total number of iterations increases from 45 to 130 as the required refinement level is modified.

In figure 33 (b), we can see the change of the distance error histogram before and after refinement. As shown in table 34, the maximum error is within the bounds set by the



(a) (b) (c)

Figure 32: (a) The mesh of figure 29 (a) after the first stage of deformation; (b) Mesh after coarse refinement; (c) Mesh after fine refinement.



(a) (b)

Figure 33: (a) The number of faces refined for each refinement procedure called; (b) The distance error histogram before and after refinement.

refinement criterion. Note that the median error does not vary substantially between the two levels of refinement. This is because the median error corresponds approximately to the resolution of the range data.

	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

Figure 34: Distance error and computation time for three meshes at different refinement levels.

The computation times were evaluated on a DEC Alphastation 200/233. We indicate in table 34 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.

5.2 Qualitative Results

5.2.1 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 creation of holes, as described in section 4.3.1, can separate 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 fit those contours. Direct reconstruction techniques, such as Delaunay triangulation [BG93] or the Marching Cubes algorithm, extrapolate the contours without taking into account any smoothness constraint.

In our framework, the deformable mesh is attracted by the contours and submitted to regularizing forces. To compute the attraction force, we simply consider the contours as edge voxels. We start the reconstruction of the lungs with the automatic initialization of the mesh around the data. We chose the spherical topology and obtain a mesh surrounding both lungs (see figure 35(a)). The first stage of deformation creates a smooth model where we can correctly detect the part that has been interpolated. Because this part has two borders, the removal of the interpolated vertices entails the separation of the mesh into two meshes (figure 35(c)). Those meshes are then refined and the final reconstruction is seen in figure 35(f).

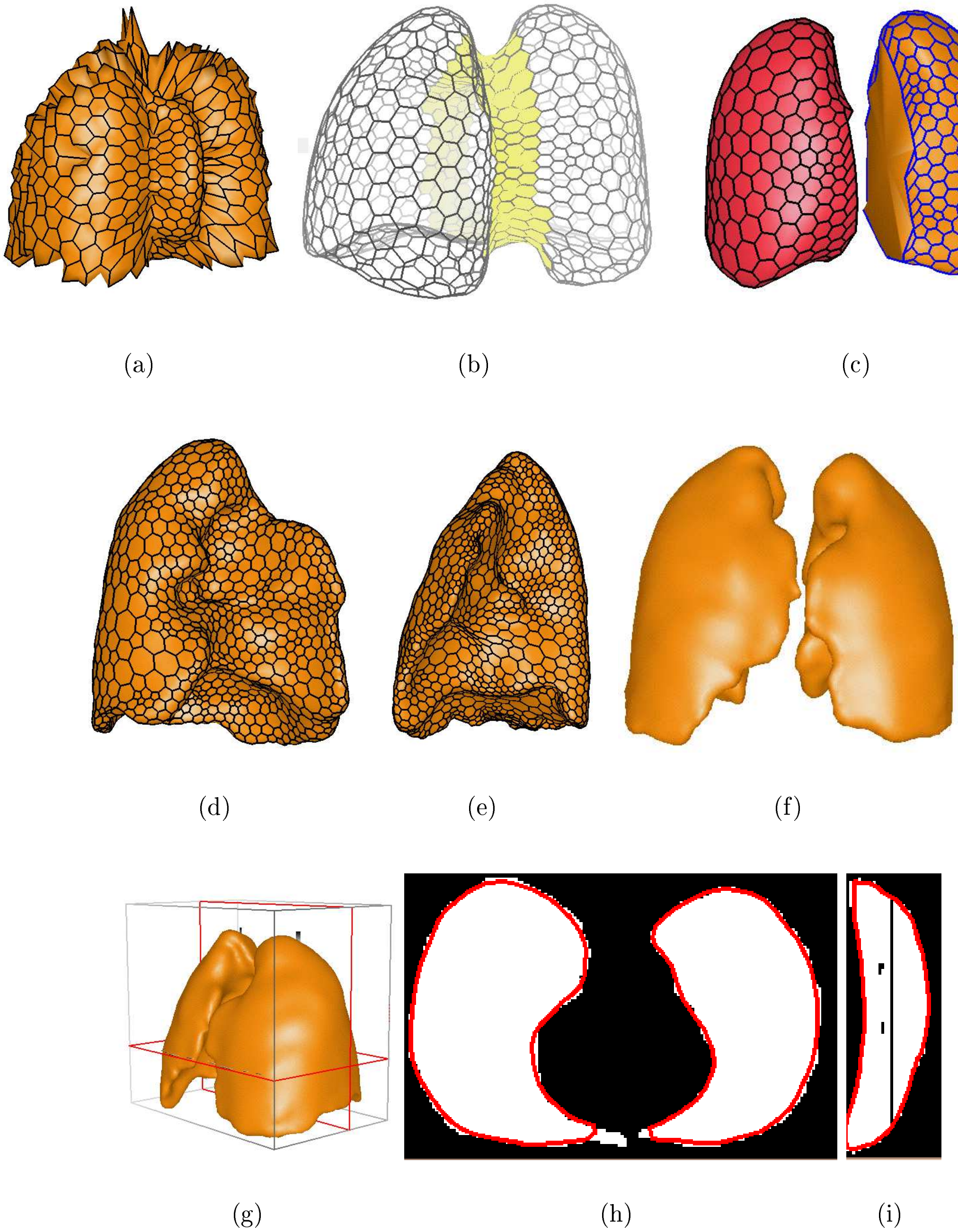


Figure 35: (a) Initial mesh; (b) The zone detected as interpolated; (c) Change of connectivity; (d) and (e) Reconstructed model of the first and second lung; (f) and (g) Final reconstruction; (h) and (i) Slices of the lung models.

5.2.2 Reconstruction from volumetric images

In this example, we reconstruct the ventricles, atriums 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 initialize automatically our models because it is impossible to isolate the ventricles and atriums in the image. Instead, we initialize the ventricles and atrium models as ellipsoids (figure 36(a)) and we manually position them within the image. The pericardium is initialized as a cylinder.

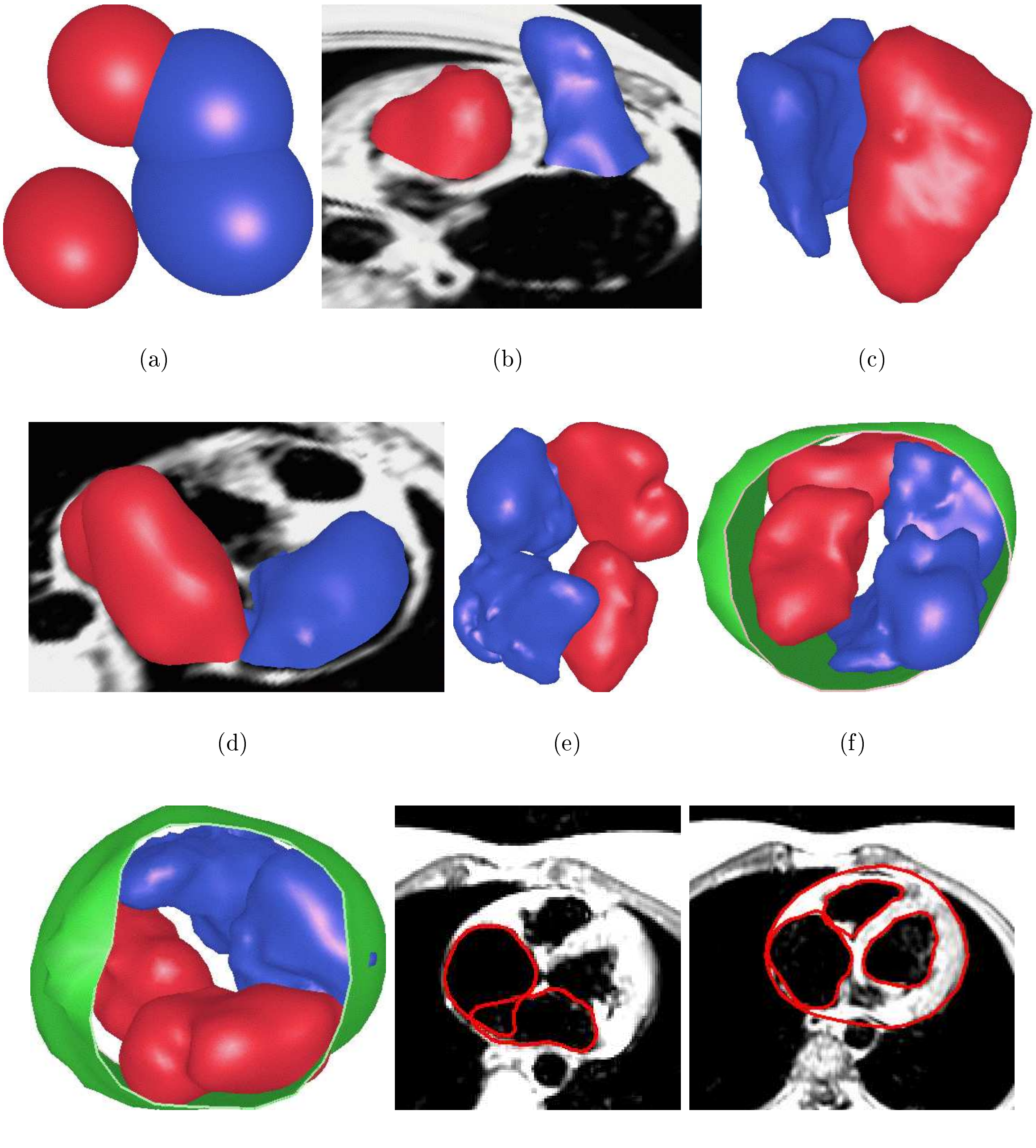
The models are attracted towards edges with the additional constraint that the surface normal must be coherently oriented with the gradient direction in the image. The gradient information was computed as a 2D gradient because the interslice distance is six times larger than the pixel size.

We first deform those models with a high rigidity parameter and, in a second stage, we decrease the rigidity parameter to a minimum while slightly refining the meshes based on area. Because of the rough initialization of the models, we allow the user to grab the mesh locally 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 36(b) and (c)) while the recovery of the atriums 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 36(h)). In figure 36(i) the slicing of the different models shows the accuracy reached in the segmentation. A better fit could be reached 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.

5.2.3 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 37(a). After recovering each pieces, they have been manually connected with several T_3^2 operators. The connected parts 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 to perform the reconstruction by parts in a straightforward manner.



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 36: (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) Slicing of the different models with the image.

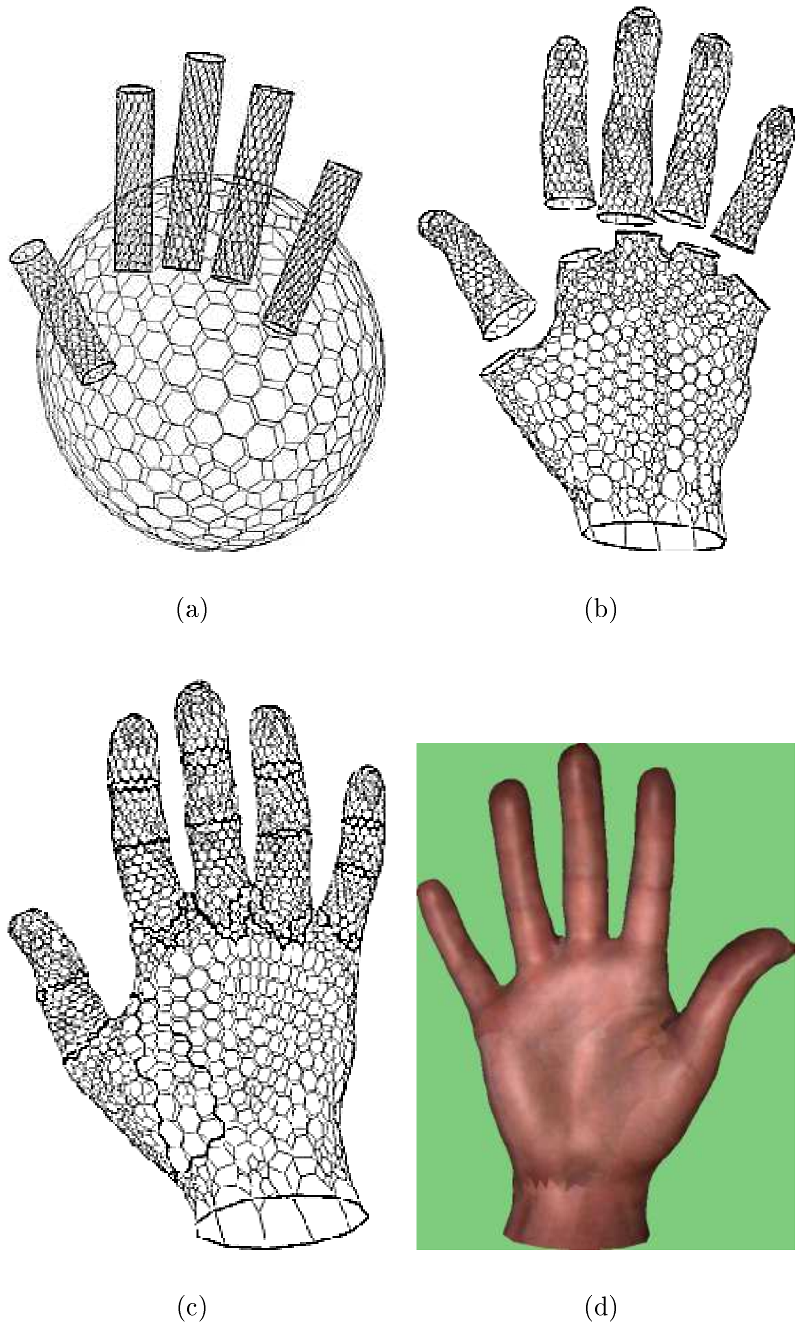
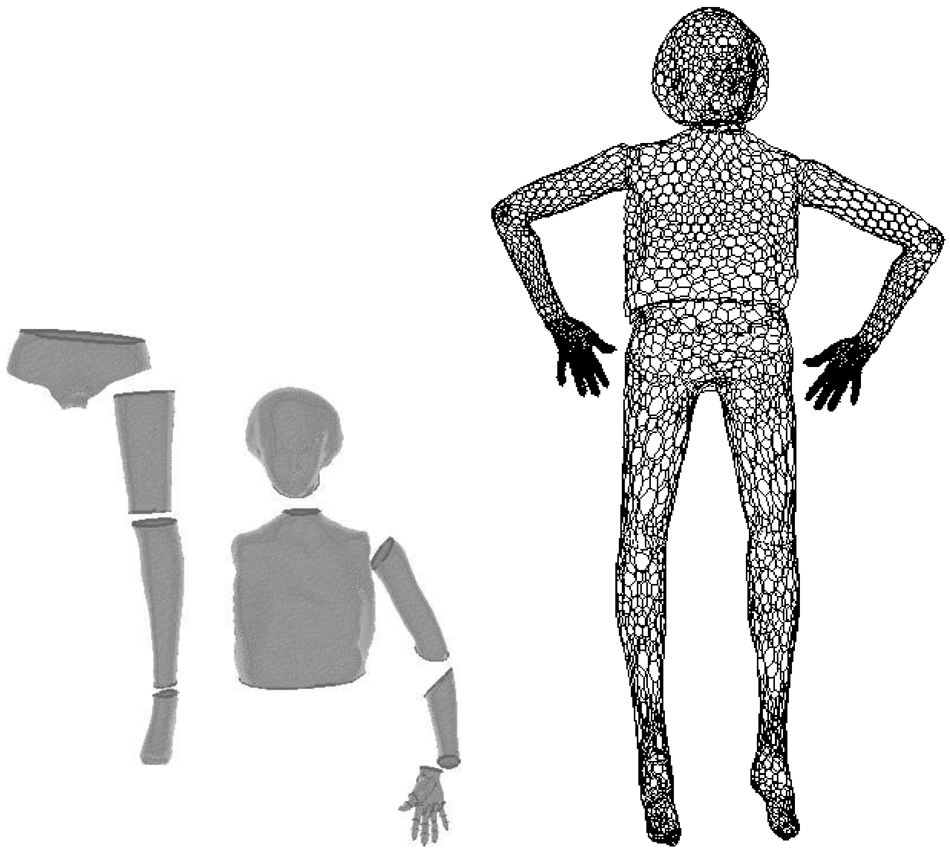


Figure 37: (a) Initial estimate of the palm and the five fingers; (b) The six recovered meshes; (c) Simplex mesh model after connecting the pieces together; (d) Textured hand model.

This technique is useful as well for reconstructing an object from many non-recovering range images. In this case, it is difficult to merge all data points in the same frame. We can recover the different parts of the object from each image, and then merge the meshes based on some rough rigid transformation between the various images. We have build a complete body model with eleven distinct body parts stored as Cyberware range images (see figure 38).



(a) (b)

Figure 38: (a) The different parts extracted from range images; (b) The body model built by connecting the 11 body parts.

6 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 with deformable surfaces. The deformable models can be refined, adapted, and their level of continuity and smoothness can be easily controlled. Because simplex meshes can represent all topologies, we have devised an algorithm for adapting the mesh topology to the topology of the object. Furthermore, we have proposed a solution to the initialization problem for the spherical, planar or cylindrical topologies. Finally, we have applied this framework to the reconstruction of complex objects from range images or volumetric images.

We would like to improve the robustness of this framework by creating better initial models. The initialization technique we have proposed, requires that the object has been isolated from the scene. We intend to develop a more general initialization technique that does not require the a priori knowledge of the object topology.

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 be specifically beneficial for segmenting medical images where databases of volumetric images and anatomic structures could be used.

Appendix A Approximation problem

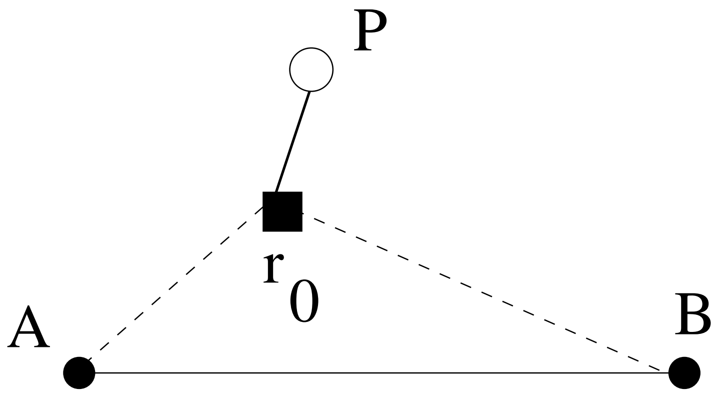


Figure 39: The solution of this approximation problem is piecewise linear.

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 **A** and **B** the extremities of the curve, **P** the data point and $\mathbf{r}_0$ the vertex of parameter u_0, then the problem consists 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 :

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

References

[BE91] M. Bern and D. Eppstein. Mesh generation and optimal triangulation. In D. Du and F. Hwang, editors, *Computing in Euclidean Geometry*, pages 23–90, World Scientific Publishing Co, 1991.

[BG93] 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*, San Jose, CA, February 1993.

[BGK92] C. Brechbuhler, G. Gerig, and O. Kubler. Surface parameterization and shape description. In Richard A. Robb, editor, *Visualisation in Biomedical Computing*, pages 80–89, 1992.

[Boi84] JD. Boissonnat. Geometric structures for three-dimensional shape representation. *ACM Transactions in Graphics*, 1984.

[Boy96] E. Boyer. Reconstruction et régularisation de la surface d'objets courbes. In *Conference sur la Reconnaissance des formes et intelligence artificielle (RFIA'96)*, pages 23–32, Rennes, France, January 1996.

[CC90] 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.

[CCA92] 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*, 242–263, 1992.

[CL96] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Computer Graphics (SIGGRAPH'96)*, 1996.

[CM94] 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.

[CTLC93] T.F. Cootes, C.J. Taylor, A. Lanitis, D.H. Cooper, and J. Graham. Building and using flexible models incorporating grey-level information. In *Proc. of the Fourth Int. Conf. on Computer Vision (ICCV'93)*, pages 242–245, 1993. Berlin.

[Del94] H. Delingette. *Simplex Meshes: a General Representation for 3D Shape Reconstruction.* Technical Report 2214, INRIA, March 1994.

[DF96] F. Devernay and O. Faugeras. From projective to euclidian reconstruction. In *International Conference on Computer Vision and Pattern Recognition (CVPR'96)*, San Francisco, 1996.

[DHI91] 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.

[EH96] M. Eck and Hugues Hoppe. Automatic reconstruction of b-spline surfaces of arbitrary topological type. In *Computer Graphics (SIGGRAPH'96)*, 1996.

[FA94] J. Feldmar and N. Ayache. Rigid and affine registration of smooth surfaces using differential properties. In *3rd European Conference on Computer Vision (ECCV'94)*, Stockholm, May 1994.

[IT95] T. Mc Inerney 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.

[IT96] T. Mc Inerney and D. Terzopoulos. Deformable models in medical image analysis. *Journal of Medical Image Analysis*, Vol 1(No 2):91–108, 1996.

[J94] Peters J. *Constructing C1 Surfaces of arbitrary topology using biquadratic and bicubic splines*, pages 277–293. SIAM, 1994.

[KMB94] 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)*, pages 441–446, Stockholm, 1994.

[LC87] W. Lorensen and H.E. Cline. Marching cubes: a high resolution 3d surface construction algorithm. *ACM Computer Graphics (SIGGRAPH'87)*, 21:163–169, 1987.

[LD90] C. Loop and T. DeRose. Generalized b-spline surfaces of arbitrary topology. In *Computer Graphics (SIGGRAPH'90)*, pages 347–356, 1990.

[Lei93] F. Leitner. *Segmentation Dynamique d'images tridimensionnelles*. Ph.D. dissertation, Institut National Polytechnique de Grenoble, Grenoble, France, September 1993.

[Mal89] J.-L. Mallet. Discrete smooth interpolation. *ACM Transaction on Graphics*, 8(2):121–144, April 1989.

[McI93] 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.

[MSV95] R. Malladi, J. Sethia, and B. Vemuri. Shape modeling with front propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2):158–175, 1995.

[Mur91] S. Muraki. Volumetric shape description of range data using "blobby model". In *Computer Graphics (SIGGRAPH'91)*, pages 227–235, 1991.

[NA93] 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.

[PS90] F. P. Preparata and M. I. Shamos. *Computational Geometry: an introduction Texts and monographs in computer science*. Springer-Verlag, 1990.

[PS91] Alex Pentland and Stan Sclaroff. Closed-form solutions for physically based shape modelling and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(7):715–729, July 1991.

[Rus91] P. Rushenas. *Etude et Implantation d'un systeme de modelisation geometrique multidimensionnelle pour la conception assistee par ordinateur*. Ph.D. dissertation, Universite de Montpellier II, July 1991.

[SD92] L. Staib and J. Duncan. Deformable fourier models for surface finding in 3D images. In R. Robb, editor, *Visualization in Biomedical Computing (VBC'92)*, pages 90–104, SPIE, 1992. Chapell Hill.

[ST92] R. Szeliski and D. Tonnesen. Surface modeling with oriented particle systems. In *Computer Graphics (SIGGRAPH'92)*, pages 185–194, July 1992.

[SW91] 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.

[Tau95] 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.

[TBG95] 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.

[TCSP92] 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.

[VM93] B. Vemuri and R. Malladi. Constructing intrinsic parameters with active models for invariant surface reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 668–681, 1993.

[VT92] 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.

[WW94] W. Welch and A. Witkin. Free-form shape design using triangulated surfaces. In *Computer Graphics (SIGGRAPH'94)*, pages 247–256, ACM, Orlando, USA, July 1994.

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399