The Ellipsoidal Skeleton in Medical Applications

F. Banégas
CIRAD[*]
fbanegas@wanadoo.fr

M. Jaeger
CIRAD
jaeger@cirad.fr

D. Michelucci
EMSE[†]
micheluc@emse.fr

M. Roelens
EMSE
roelens@emse.fr

Abstract

Rough 3D data images obtained by computed tomography or magnetic resonance imagery are inadequate: this paper proposes a high-level data structure called ellipsoidal skeleton. It is based on a tree of best partitions of the points set and features data compression, multi-level representation capabilities, surface reconstruction, interactive visualization, relevant parameters extraction, automatic matching and recognition.

Keywords: Medical Applications, Dynamic Clustering, Surface Reconstruction, Shape Matching and Recognition, Signature, Tabu Search

1 Introduction

Medical Imaging has to cope with increasingly huge 3D data images: CT-scan or MRI images. The data size does not permit interactive visualization, and lack of structure prevents extraction of relevant informations. Some paradigm is dramatically required, to visualize interactively relevant data with various levels of details, to reconstruct the surface of organs, to extract relevant features for diagnosis or surgical interventions, and to automatically match objects extracted from different 3D data images.

The heart of the solution proposed here is a method to find a tree of best partitions of the set of 3D points from segmented image data. For a given number of classes, the best partition is the one which maximizes the homogeneity of classes and the differences between classes. The partitions in $1, 2 \ldots k$ classes are organized in a tree: the E-skeleton, the construction of which is presented in section 2.

For a given level of details, *i.e.* for a given number k of classes, the k-partition permits to reconstruct the surface of the 3D object. Each class is geometrically approximated by an ellipsoid, and all k ellipsoids are merged with a Blinn surface. Possibly, a Barr deformation can be applied to each ellipsoid to best fit the surface. See section 3.

Due to the hierarchical nature of the E-skeleton, it is then possible to reconstruct and visualize the surface with several levels of detail, as explained in section 4.

The E-skeleton is invariant through isometries, and very robust against discretisation noise, over- and under- sampling, local deformations. Thus two points sets of similar objects give similar E-skeletons. This steadyness property is exploited for automatic matching between shapes, computing distances between shapes, shape recognition: see section 5.

Beyond ellipsoids, it is possible to extract from E-skeleton other parameters relevant for a given application. Examples are given in section 6.

2 Construction of the E-Skeleton

2.1 Expectations and Variances

Let $\mathcal{C} = \text{cloud}\,(\Omega, P)$ be a cloud of n points, with weights $\Omega \in \mathbb{R}^n$ and coordinates $P = (X \in \mathbb{R}^n, Y \in \mathbb{R}^n, Z \in \mathbb{R}^n)$. They are equally distributed in the interior of the studied organ. In our applications, all initial points have equal weights, but unequal weights occur during computations; moreover some segmentation methods may weight points. The X expectation (or mean value, or center of gravity): $\mathrm{E}\,(\Omega, X)$ or $\mathrm{E}\,(X)$ for short, is:

$$\mathrm{E}\,(X) \overset{\text{def}}{=} \frac{\sum_1^n \Omega[i] \times X[i]}{\sum_1^n \Omega[i]}$$

and Y- and Z-expectation are defined similarly. The covariance of X and Y: $\mathrm{C}\,(\Omega, X, Y)$ or $\mathrm{C}\,(X, Y)$ for short, is:

$$\begin{aligned} \mathrm{C}\,(X,Y) &\overset{\text{def}}{=} \mathrm{E}\,(XY) - \mathrm{E}\,(X) \times \mathrm{E}\,(Y) \\ &= \mathrm{C}\,(Y,X) \end{aligned}$$

where XY stands for $[\ X[1] \times Y[1], \ldots X[n] \times Y[n]\ ]$. The variance of X: $\mathrm{V}\,(\Omega, X)$ or $\mathrm{V}\,(X)$ for short, is:

$$\begin{aligned} \mathrm{V}\,(X) &\overset{\text{def}}{=} \mathrm{C}\,(X,X) \\ &= \mathrm{E}\left(X^2\right) - \left(\mathrm{E}\,(X)\right)^2 \end{aligned}$$

It is also the expectation of the squares of the differences between X and $\mathrm{E}\,(X)$:

$$\mathrm{V}\,(X) = \mathrm{E}\left(X - \mathrm{E}\,(X)\right)^2\Big)$$

Some simple and useful properties ($k \in \mathbb{R}$, and $k + X$ is a shortcut for $[k\ k\ \ldots k] + X$):

$$\begin{aligned} &\mathrm{E}\,(k + X) = k + \mathrm{E}\,(X) \\ &\mathrm{E}\,(kX) = k \times \mathrm{E}\,(X) \\ &\mathrm{E}\,(X + Y) = \mathrm{E}\,(X) + \mathrm{E}\,(Y) \\ &\mathrm{V}\,(k + X) = \mathrm{V}\,(X) \\ &\mathrm{V}\,(kX) = k^2 \times \mathrm{V}\,(X) \end{aligned}$$

[*] CIRAD: Centre International de Recherche en Agronomie et Développement, avenue d'Agropolis, BP5035, 34032 Montpellier cedex 1, France.

[†] EMSE: Ecole des Mines de St-Etienne, 158 cours Fauriel, 42023 St Etienne cedex 2, France.

$$\mathrm{V}(X+Y) = \mathrm{V}(X) + \mathrm{V}(Y) + 2\mathrm{C}(X,Y)$$
$$\mathrm{C}(X, Y+k) = \mathrm{C}(X,Y)$$
$$\mathrm{C}(X, kY) = k \times \mathrm{C}(X,Y)$$
$$\mathrm{C}(X, Y+Z) = \mathrm{C}(X,Y) + \mathrm{C}(X,Z)$$

The euclidean variance of $\mathcal{C}$ is:

$$\mathrm{V}(\mathcal{C}) \stackrel{\mathrm{def}}{=} \mathrm{V}(X) + \mathrm{V}(Y) + \mathrm{V}(Z)$$

Actually it is also the expectation of the squares of the distances between points P and their gravity center $(\overline{X} = \mathrm{E}(X), \overline{Y} = \mathrm{E}(Y), \overline{Z} = \mathrm{E}(Z))$:

$$\begin{aligned}&\mathrm{E}\left((X-\overline{X})^2 + (Y-\overline{Y})^2 + (Z-\overline{Z})^2\right)\\ &= \mathrm{E}\left((X-\overline{X})^2\right) + \mathrm{E}\left((Y-\overline{Y})^2\right) + \mathrm{E}\left((Z-\overline{Z})^2\right)\\ &= \mathrm{V}(X) + \mathrm{V}(Y) + \mathrm{V}(Z) = \mathrm{V}(\mathcal{C})\end{aligned}$$

thus, since distances are invariant through isometries (translations, rotations, symmetries and their composition), $\mathrm{V}(\mathcal{C})$ as well.

2.2 The Covariance Matrix

The covariance matrix M of the cloud $\mathcal{C}$ is:

$$M \stackrel{\mathrm{def}}{=} \begin{pmatrix} \mathrm{V}(X) & \mathrm{C}(X,Y) & \mathrm{C}(X,Z) \\ \mathrm{C}(Y,X) & \mathrm{V}(Y) & \mathrm{C}(Y,Z) \\ \mathrm{C}(Z,X) & \mathrm{C}(Z,Y) & \mathrm{V}(Z) \end{pmatrix}$$

This matrix is symmetric and positive, so eigenvalues are real and non-negative. In the generic case, the matrix is definite: degeneracies occur for instance when all points in the cloud lie on a plane or a line. Such degeneracies are easy to detect because one or two eigenvalues are much smaller than the greatest one. From now on, only the generic case is considered. Thus the 3 eigenvalues are real and positive : $\lambda_1 \geq \lambda_2 \geq \lambda_3 > 0$.

The 3 corresponding eigenvectors, once normalized, define an orthonormal coordinates system, and are called the main axis of the cloud $\mathcal{C}$. The major main axis corresponds to λ_1.

When expressed in this coordinates system (called the natural coordinates system of the cloud), the covariance matrix of the cloud is the diagonal matrix $(\lambda_1, \lambda_2, \lambda_3)$, *i.e.* $\mathrm{C}(X,Y) = \mathrm{C}(Y,Z) = \mathrm{C}(X,Z) = 0$. Up to orientations, this natural coordinates system is unique. Accounting for orientations, there are 2 such coordinates systems in one dimension, and 2^{d-1} in d dimensions when $d > 1$.

Figure 1: There are 2 natural coordinates systems in 2D.

The covariance matrix has same eigenvectors as the matrix of inertia, more familiar to mechanical engineers, and which has a strong physical meaning.

2.3 Best Partitions

Let $\{\mathcal{C}_1, \ldots \mathcal{C}_k\}$ be a k-partition (*i.e.* a partition in k non empty classes) of the cloud $\mathcal{C}$. The euclidean variance $\mathrm{V}(\mathcal{C})$ of the cloud $\mathcal{C}$ is decomposed into two parts: the sum of all euclidean variances of clouds $\mathcal{C}_1, \ldots \mathcal{C}_k$ (the latter are called: intraclass), and the interclass variance $\mathrm{V}(\{\mathcal{C}_1, \mathcal{C}_2 \ldots \mathcal{C}_k\})$:

$$\mathrm{V}(\{\mathcal{C}_1, \mathcal{C}_2 \ldots \mathcal{C}_k\}) \stackrel{\mathrm{def}}{=} \mathrm{V}(\mathcal{C}) - \sum_{i=1}^{k} \mathrm{V}(\mathcal{C}_i)$$

Let $\Omega_i, P_i = (X_i, Y_i, Z_i)$ be the vectors of weights and coordinates of the cloud $\mathcal{C}_i$. Let n_i be the number of points in $\mathcal{C}_i$. $G_i = (\mathrm{E}(\Omega_i, X_i), \mathrm{E}(\Omega_i, Y_i), \mathrm{E}(\Omega_i, Z_i))$ is the gravity center of $\mathcal{C}_i$, and $w_i = \sum_{j=1}^{n_i} \Omega_i[j]$ is the weight of G_i, *i.e.* the weights sum of points in $\mathcal{C}_i$. Pose $G = (G_1, G_2 \ldots G_k)$ and $W = (w_1, w_2 \ldots w_k)$: they define a cloud $\mathcal{G} = \mathrm{cloud}(W, G)$, whose euclidean variance is the euclidean interclass variance of the partition $\{\mathcal{C}_1, \ldots \mathcal{C}_k\}$.

The best k-partition is the one with maximal interclass variance, and thus with minimal sum of intraclass variances, since the sum of both is constant. It means classes are as different and homogeneous as possible.

Extreme cases: if $k = 1$, $\mathrm{V}(\{\mathcal{C}_1\}) = 0$: the interclass variance vanishes. If $k = n$ (there is one class per point), $\mathrm{V}(\mathcal{C}) = \mathrm{V}(\{\mathcal{C}_1, \ldots \mathcal{C}_n\})$ and $\mathrm{V}(\mathcal{C}_i) = 0, i = 1 \ldots n$.

The progressive dynamic clustering algorithm finds the best partitions.

2.4 Computing Best Partitions

2.4.1 The Dynamic Clustering Algorithm

To compute the best k-partition of a cloud, *i.e.* the partition that maximizes the interclass variance, the DC algorithm [15] proceeds as follow:

```
Choose an initial $k$-partition
Repeat
    For each class $\mathcal{C}_i$, $i = 1 \ldots k$
        Compute $G_i$ the center of gravity of $\mathcal{C}_i$
        Empty the class $\mathcal{C}_i$
    EndForEach
    For each point $p_j$
        Add $p_j$ to class $\mathcal{C}_i$ where $G_i$ is closest to $p_j$
    EndForEach
Until neither of $G_i$ change
```

This algorithm is guaranteed to find the optimal solution; oscillations due to degeneracies or inaccuracies may happen in some case but are easily avoided. The DC algorithm has two main drawbacks: the number of steps can be very important, and the initial choice of the partition has a great influence on this number. It is therefore crucial to start with a good initial partition.

2.4.2 The Progressive DCA

To improve the DC algorithm, we construct a progressive DCA: initially, it is trivial to find the optimal 1-partition. Assume we know the optimal k-partition $(\mathcal{C}_1, \mathcal{C}_2 \ldots \mathcal{C}_k)$. For each and every class $\mathcal{C}_i, i = 1 \ldots k$, we split $\mathcal{C}_i$ by the plane containing G_i and orthogonal to the major main axis of $\mathcal{C}_i$; a DCA is used to split optimally $\mathcal{C}_i$ in two sub-classes $\mathcal{C}'_i$ and $\mathcal{C}''_i$; we then compute the ratio $r_i \stackrel{\mathrm{def}}{=} \frac{\mathrm{V}(\mathcal{C}'_i) + \mathrm{V}(\mathcal{C}''_i)}{\mathrm{V}(\mathcal{C}_i)}$.

The initial $k+1$-partition for the DCA: $\{\mathcal{C}_1, \ldots, \mathcal{C}'_i, \mathcal{C}''_i, \ldots \mathcal{C}_k\}$ is obtained by splitting the class $\mathcal{C}_i$ with smallest r_i ratio.

2.4.3 Optimal Maximal Number of Classes

The simplest criterion to stop the subdivision process is the ratio of the sum of intraclass variance to the cloud variance: when this ratio becomes less than a given threshold, typically 5%, the subdivision is stopped. This ratio is a monotonic decreasing function of k.

Many methods have been proposed to find optimal values for k, but they are rather complex to implement and are often application specific [33, 21]. For our application, the simplest criterion is sufficient.

2.4.4 An Improvement

An easy way to speed up the progressive DCA is to undersample the cloud $\mathcal{C}$ into a cloud $\mathcal{C}'$, keeping for instance only the central voxel (or the gravity center) of each cube of $3 \times 3 \times 3$ contiguous voxels in the 3D image of $\mathcal{C}$, to compute the best k-partition P' of $\mathcal{C}'$, to replace in P' the voxels of $\mathcal{C}'$ by the corresponding voxels in $\mathcal{C}$, and to use the latter as an initial k-partition for the DCA—maybe this principle can be used recursively, while the cloud has enough points, say k^2, but this last recursive optimization has not been tried.

This optimization uses the grid structure of 3D data images, which is always present in medical imaging. We have not investigated the use of octree data structures to speed up the DCA.

2.4.5 Examples

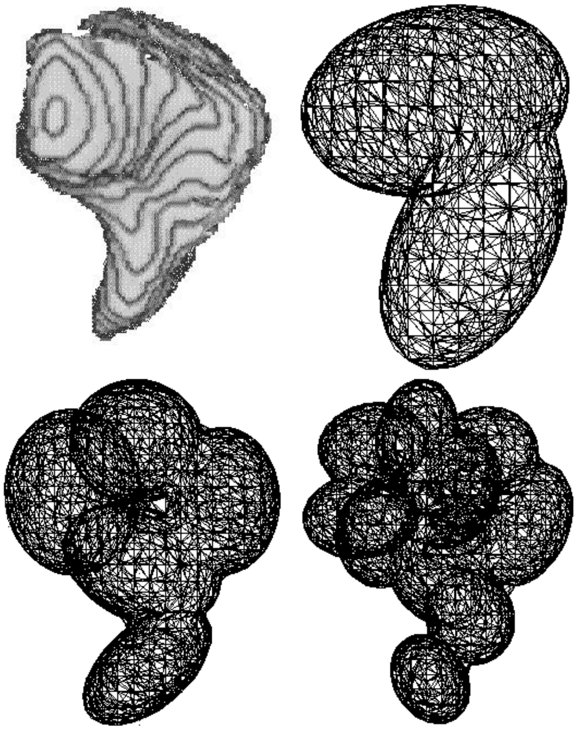

Figure 2: A carpal bone, the hamatum: the cloud, 2-, 5-, and 10-partition.

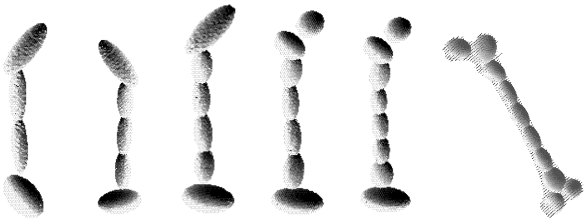

Figure 3: A femur. 4-, 5-, 6-, 7-, 8-, 10- partitions.

2.4.6 Mahalanobis Variant

With increasing k, classes of k-partitions become more and more spherical, so that a tubular part (for instance the middle part of a femur) is partitionned by a string of balls. One may prefer to keep such a part as a single class. A solution is to apply to the initial cloud $\mathcal{C}$, with eigenvalues λ_i, a scaling transform, with factors $(\lambda_1^{-1/2}, \lambda_2^{-1/2}, \lambda_3^{-1/2})$, in the natural coordinates system of $\mathcal{C}$. Let $\mathcal{C}'$ be the resulting cloud: all eigenvalues of its covariance matrix are equal to 1. Then the k-partition of $\mathcal{C}'$ is computed with the progressive DCA (with or without optimizations). Due to the 1-1 correspondance between $\mathcal{C}$ and $\mathcal{C}'$, the k-partition of $\mathcal{C}'$ is also a k-partition of $\mathcal{C}$, which can be called the optimal *normalized* k-partition of $\mathcal{C}$. This variant is equivalent to using Mahalanobis distance. An advantage of this approach is that two clouds equal up to some affine transform have equal optimal normalized k-partitions, by definition. This variant has been tested, but unfortunately it gives counter intuitive results with increasing k (see [3] for details), and we do not retain it.

2.5 Construction of the Tree

A tree: the E-skeleton, keeps track of the subdivision process. Each node or leaf represents a class. At level k (1 for the root) this tree has k nodes, which represent the optimal k-partition. At each level k, except the leaves level, $k - 1$ nodes have exactly one children, and one node has two child: the latter represents the class which was split to obtain the optimal $k + 1$-partition.

Apart its childhood, each node stores its gravity center, its weight, its number of points, its covariance matrix, its eigenvalues and eigenvectors (the main axis). Geometric values α and modal deformation u, which are used for geometric reconstruction and defined below, are also stored.

3 Reconstruction of the Surface

3.1 Previous Works

Various approaches have been proposed for surface reconstruction, from tetrahedral reconstruction [9] to generalized cylinders [35]. Such techniques remain locally based or need strong topological information about contour points, which can be ambiguous in some case.

Methods that are more generic exist, like active contours [11] or snakes, offering compact analytical expression of 3D objects without the need of any knowledge of the topology. The concept of reconstruction with multiple primitives [24] can be seen as a generalization of these snakes, and has become very popular [13, 38, 6]. Formalisms have been proposed to provide strong bases for primitive combining [2, 31, 39, 28] and fitting [4, 37].

Parametric primitives [14], while providing great flexibility, are often complex to combine. Implicit surfaces [7] simplify this issue but must be wisely used to avoid the evaluation of numerous expensive equation. It is why implicit primitives must not only be positioned carefully according to the point cloud, but also be as representative as possible [1]. They should offer a level of realism of their own.

Combined with this feature, level of detail generation must be present: multi-resolution representation [29] can be achieved via triangle decimation [30], progressive meshes [20] or discretization step control [22].

3.2 Our Approach

An optimal k-partition of the cloud can be used to construct a geometric approximation of the surface of the object. To get a rough approximation, sufficient when the object is far from the eye, we approximate each class with its best ellipsoid (section 3.3), and union them. To get a smoother and more accurate approximation, we use the Blinn formalism [7], inspired from molecular chemistry: a surface is defined as the set of points where a potential is equal to 1. This potential is defined as the sum of elementary potentials: in our model, each class $\mathcal{C}_i$ of the skeleton is attached a geometric primitive, which emits an elementary potential $P_i(x, y, z)$ in all space:

$$P_i(x, y, z) \overset{\text{def}}{=} e^{-\alpha_i d_i(x,y,z)} \tag{1}$$

where $d_i(x, y, z)$ is some distance to the geometric primitive boundary, and $\alpha > 0$ some adjusting parameter (the slope of the potential curve as a function of the distance d_i). The distance $d_i(x, y, z)$ must be zero on the boundary of the geometric primitive of the class, negative inside the geometric primitive, and positive outside. We now present possible geometric primitives for a class, and the corresponding distance.

3.3 The Best Ellipsoid of a Class

The geometric primitive for a class can be an ellipsoid: the center is the gravity center $G = (x_g, y_g, z_g)$ of the class, its main axis are the main axis of the class (*i.e.* eigenvectors of its covariance matrix), and its radius a, b, c are $\gamma\sqrt{\lambda_i}, i = 1, 2, 3$, where λ_i are eigenvalues of the related covariance matrix, and γ is a scaling constant.

There are 2 possible choices for γ. Either we want that the best ellipsoid is the cloud itself when the latter is a regularly sampled ellipsoid: then $\gamma = \sqrt{d+2}$, where d is the space dimension; in our case, $d = 3 \Rightarrow \gamma = \sqrt{5} \approx 2.236068$. Or we use the big numbers law: if the random variable X follows a normal law, the interval $[-2\sqrt{\mathrm{V}(X)}, 2\sqrt{\mathrm{V}(X)}]$ contains 90% of the samples, and so $\gamma = 2$, in all dimensions. We choose the latter.

In the natural coordinates system of the class, the ellipsoid has equation: $E(x, y, z) = 0$ where:

$$E(x, y, z) = \frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} - 1$$

Expressing $E(x, y, z)$ in world coordinates gives the simplest possible distance for equation 1. More precisely, this equation becomes:

$$E(p) = (p - G)Q(p - G)^t$$

where G is the gravity center, $Q = RDR^t$, $D = 1/\gamma^2 \times$ diag$(\lambda_1^{-1/2}, \lambda_2^{-1/2}, \lambda_3^{-1/2})$, and R is the matrix of eigenvectors of the covariance matrix, so that $R^t = R^{-1}$ and $M = R^t$diag$(\lambda_1, \lambda_2, \lambda_3)R$. The diagonalization of M can be computed with Jacobi rotations [27].

3.4 The Best Superquadric of a Class

Superquadrics are another geometric primitive. In the natural coordinates system of the class, they have equation $S(x, y, z) = 0$ with:

$$S(x, y, z) = \left(\left(\frac{x}{a}\right)^{\frac{2}{\epsilon_1}} + \left(\frac{y}{b}\right)^{\frac{2}{\epsilon_1}}\right)^{\frac{\epsilon_1}{\epsilon_2}} + \left(\frac{z}{c}\right)^{\frac{2}{\epsilon_2}} - 1$$

with $\epsilon_1 > 0, \epsilon_2 > 0$. For $\epsilon_1 = \epsilon_2 = 1$, superquadrics reduce to ellipsoids. Moreover, these parameters allow a continuous characterization of the shape of each class, adding useful information for pattern matching 5.2.

Expressing $S(x, y, z)$ in world coordinates gives the corresponding distance function in equation 1. To find best parameters ϵ_1, ϵ_2, some optimization is required (section 3.6).

3.5 Deformed Superquadric of a Class

Deformed superquadrics are the last possible geometric primitive of our model. We use the modal deformation D_u, as defined by Barr [4]. u is a 30 dimensional vector: $(t_x\ t_y\ t_z\ \phi\ \theta\ \psi\ a\ b\ c\ \mu_1\ \ldots \mu_p)$ whose coordinates describe elementary deformations (translation, rotation ... bending, twisting). u defines an affine map $D_u(p)$ in every point p of space, *i.e.* a 4×4 matrix in homogeneous coordinates, which is a function of $p \in \mathbb{R}^3$. If $S(p) = 0$ is the equation of the non deformed superquadric, then the deformed superquadric has equation: $S_u(p) = S(p \times D_u(p)^{-1}) = 0$. Expressing S_u in world coordinates gives the distance function for the deformed superquadric in equation 1. Some optimization is required (section 3.6) to find the best vector u.

3.6 Optimisation

In the simplest case the k best ellipsoids are just merged with $\alpha_i = 1$, or even just unioned: no optimization is needed. It gives a schematic representation which is perfect for pedagogical illustrations.

If a more accurate reconstruction is needed, we have first to find the best geometric primitive for each sub-cloud, second to find best α_i for merging the k best geometric primitives. These two stages simplify the resolution process. We use the same resolution method for both problems, so we discuss only the first one.

The optimization method has to find the best values for parameters $\epsilon_1, \epsilon_2, u$, in order for the deformed superquadric to best fit the boundary points of the class. The optimization minimizes the error measure: $e = \sum_{p \in B} d(p)^2/\text{card}(B)$ where B is the set of boundary points, card(B) its cardinality, and $d(p)$ is the distance function the parameters of which are optimized. B may be undersampled, to speed up. A point of the cloud is a boundary point iff one of its 6 neighbors, in the initial 3D data image, does not belong to the same cloud (It is the second place where the E-skeleton software uses the grid structure of 3D data images). Several optimization methods have been tried:

Descent methods (Levenberg-Marquardt, conjugate gradients) often stall in local minima.

Homotopy projects boundary points on the undeformed geometric primitive, moves them incrementally to their right location, and at each step performs a descent method. It works much better, but unfortunateley not always, and it is time consuming.

Tabu, a stochastic method, resembles simulated annealing [18, 19]. Tabu walks in a discretized search space: at a current vertex of this search space, it randomly chooses t_n vertices in some neighborhood, eliminates those which were recently visited (tabu manages a tabu list of the last t_v visited points, to avoid cycling), and jumps to the best vertex, even if it is worse than the current one or the best vertex met so far. The neighborhood radius decreases during the walk. Tabu stops when it has found a solution with error less than a prescribed threshold, or after a prescribed number of steps. For our application, values $t_n = 5$ and $t_v = 10$ were sufficient to find the optimal solution. Tabu is time consuming, but easy to implement and the most robust method so far. Its solution is then polished with a classic descent method.

3.7 Reconstruction Examples

After optimization, the quality of the solution is measured with the more accurate, but more time consuming, error measure $e_{me} = \sum_{p \in B} \delta(p)/\text{card}(B)$ or $e_{max} = \max_{p \in B}\delta(p)$ where [36]:

$$\delta(p) = \frac{|d(p)|}{||\nabla d\ (p)||} \approx \text{dist}(p, B)$$

The mean error is typically divided by 10, and the maximal one by 3 or 4, relatively to the simplest approximation which blends all best ellipsoids with $\alpha_i = 1$.

The table 1 gives some results for two objects: a femur and a hamatum. The following parameters are given: number of primitives n_p, number of boundary points B_p, maximal length l of the cloud (in millimeters), mean error e_{me} and maximal error e_{max} (in millimeters), and computation time t (in minutes), on a SGI O^2 workstation, R5000 at 180 Mhz.

A low number of geometric primitives gives visually satisfying results. A better accuracy may be achieved, but at the cost of a bigger number of classes, and is irrelevant for medical application, where the 3D data generation process (acquisition and segmentation) may introduce errors about 7 millimeters for a femur. Another argument against a greater accuracy is the plasticity of some organs (lever, heart, etc).

Table 1: Performance results

Object	B_p	n_p	l	e_{me}	e_{max}	t
Hamatum	7500	4	20	1.1	3.2	4’
Hamatum	7500	5	20	0.8	2.8	5’
Femur	25000	6	400	2.4	5	20’
Femur	25000	8	400	2	4.7	15’

It might be noticed that the computation time for the femur is shorter for 8 primitives than for 6: this is due to the fact that ellipsoids for 8 primitives are more spherical, that speeds up optimization.

Figures 4 to 7 show some reconstruction examples. Surfaces are tesselated with some variant of the Marching Cubes algorithm [22] (see section 4). The computation of the E-skeleton and the surface reconstruction (without marching cubes) needs typically from 2 minutes for the hamatum with 38,767 points with $k = 10$, to 6 minutes for the femur with 152,727 points also with $k = 10$ (times for Intel Pentium II at 233 MHz). The E-skeleton, including the analytical description of the surface (u vectors, α_i), is about one thousand more compact than the initial 3D data image. It is also about one hundred more compact than a medium-quality polygonization of the initial 3D image.

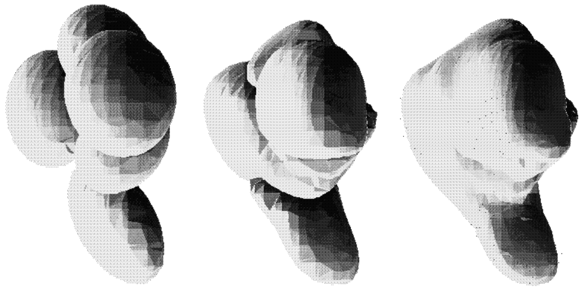

Figure 4: Reconstruction of hamatum: ellipsoids, deformed ellipsoids, blended deformed ellipsoids.

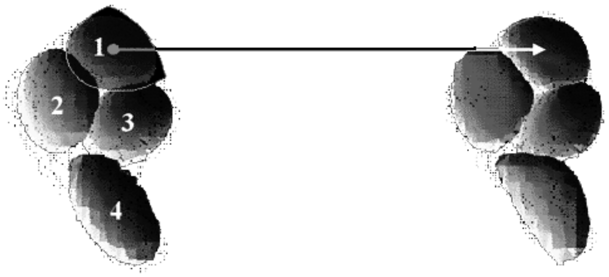

Figure 5: Deformation of ellipoids for the 4-partition of the hamatum.

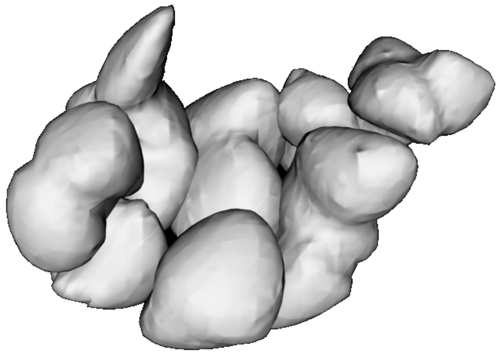

Figure 6: Reconstruction of the eight carpal bones: each bone is separately blended.

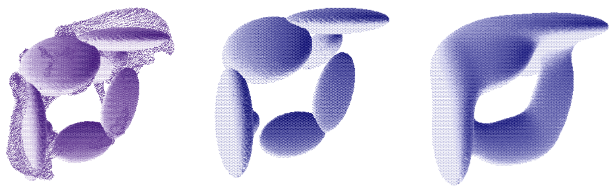

Figure 7: Reconstruction of an iliac bone: cloud and ellipsoids, ellipsoids, merged undeformed ellipsoids.

3.8 Implicit Surface with Even Degree

This paragraph describes another solution, not yet implemented: the geometric primitive for approximating a class can be any implicit algebraic surface with even degree $2d$ (to ensure boundedness), having equation $F(x, y, z) = 0$ where:

$$F(x,y,z) = \sum_{0 \leq i+j+k \leq 2d} c_{i,j,k} x^i y^j z^k \qquad (2)$$

Moreover, it is convenient to impose $c_{0,0,0} = 1$ (which implies the origin does not lie on the surface: some translation can be applied to the cloud to ensure that). For $p = (p_x, p_y, p_z)$ a given point, $F(p)$ is then a computable linear expression in $C = [c_{i,j,k}]$: $F(p) = 1 + C.l$, where l is the vector $[p_x^i \times p_y^j \times p_z^k]$ (in the same order than the $c_{i,j,k}$ in the C vector); if p is a boundary point of the class, we want $F(p) = 1 + C.l$ as small as possible. Expressing this constraint for all boundary points p, we have to minimize the euclidean norm of the vector $[1, \ldots 1] + CL$. It is a least square problem, whose analytical solution is $C = [-1, \ldots -1]L^+$, where L^+ is the pseudo inverse of L: $L^+ = L^t(LL^t)^{-1}$. An advantage of this formulation is that fast least square methods can be used [27].

Finally, the power basis: $(1, x, x^2 \ldots) \times (1, y, y^2 \ldots) \times (1, z, z^2 \ldots)$ was used only for simplicity in formula 2: as well known, it has not the property of affine invariance. The Bernstein basis does have, and should be preferred in order to avoid the shape of the resulting surface to depend on the used coordinates system.

4 Multi Level Reconstruction and Visualization

At each level, the E-skeleton stores for an object an analytical form: the surface is defined by an equation $F(x, y, z) = 1$. For interactive applications, a polygonal definition of the object is needed (this form is required by advanced graphic workstations): a meshing algorithm is then used. We choose a tetrahedric variant of the Marching Cubes algorithm [22]: this algorithm may use an extra parameter, the discretization step s.

Each mesh generation needs less than one second: we don't have to evaluate the potential in each vertex of the Marching Cube grid. Instead, for each ellipsoid, we start from its center, walk upwards until we reach the surface boundary. Either the corresponding voxel already contains some reference to a polygon: then this connected component of the surface has already been tesselated and we are done; or the corresponding voxel is empty, and we start a new meshing stage from this voxel, marking the visited voxels. Using continuity avoids computing potentials in all voxels.

We have two parameters to choose a polygonal mesh: the semantic level k and the discretization step s. This provides a great flexibility for surface rendering. To control the number of meshes, k and s are decreasing staircase functions of the distance between the eye and the object. Instead of precomputing all meshes at all levels and all discretization steps, meshes are lazily computed (*i.e.* when needed) and saved to disk after creation.

Two extra parameters may also be used to monitor the surface reconstruction process: either we use surface blending or boolean union, either we deform geometric primitives or not. For union of undeformed ellipsoids, it should be interesting to polygonize directly: the marching cube algorithm is no more needed.

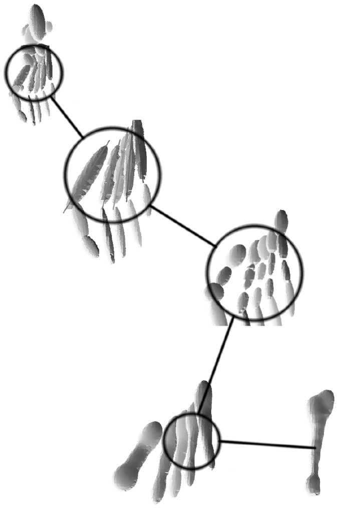

Figure 8: Dynamic rendering for visualization and exploration on tarsal bones.

5 Shape Recognition

5.1 Previous Works

Many techniques have been investigated, that may be divided into two classes: either they rely on external [8, 1, 35] or internal expression of objects. Former ones are based on the surface definition of the object (B-rep, polygonal ou B-spline representation), and latter ones use area descriptor methods (medial axis, shape decomposition). Such descriptors are either structural, like medial axis [16, 25] or scalar [32].

Shape decomposition is a powerful technique as it achieves data reduction by imposing a particular organization on it [24, 26]. Pixel- or voxel-scaled information is organized into larger models, which provides multiscale capabilities [23]. Some theories even suggest that such decomposition is present in human perception [5].

The model we propose in this paper combines all aspects of internal description-based algorithms, as it uses both structural (*i.e.* geometric) and scalar (*i.e.* by extracting relevant parameters or signatures) descriptions in a hierarchical fashion thanks to semantically zoomable shape decomposition approach.

5.2 Matching k-Partitions

This section presents the automatic comparison of two objects, at a given level k of detail, *i.e.* both clouds are described by their best k-partition. When the size is not relevant (*i.e.* we want to find equal two homothetic objects), some homothecy is applied to give them volume 1. We assume both clouds are expressed in their natural coordinates systems (expectations, variances, etc are updated accordingly).

5.2.1 Class Signature and Distance

To compute a *distance between two classes*, we attach a characteristic vector, or *signature* [34], to each class, then compute their Manhattan distance (the euclidean distance is another possibility). Our software associates to a class $\mathcal{C}_j$ the signature vector: $(\sqrt{\lambda_{j1}}, \sqrt{\lambda_{j2}}, \sqrt{\lambda_{j3}}, \rho_j)$ where λ_{ji} are eigenvalues of the covariance matrix of $\mathcal{C}_j$, and ρ_j is the distance between the gravity center of $\mathcal{C}_j$ and the one of the whole cloud $\mathcal{C}$ (*i.e.* the origin). Other choices are possible, which for example take into account angles, like angles cosines between main axis of $\mathcal{C}_j$ and main axis of the whole cloud $\mathcal{C}$.

Note it makes no sense to directly compare coordinates in $\mathcal{C}$ and coordinates in $\mathcal{C}'$, due to the existence of several natural coordinates systems.

In our applications, the simplest definition was always sufficient to match objects. More on chirality and signatures in sections 5.5, 5.6.

5.2.2 The Matching Problem

Let $\mathcal{C}$ and $\mathcal{C}'$ be two clouds described by two k-partitions: $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2 \ldots \cup \mathcal{C}_k$ and $\mathcal{C}' = \mathcal{C}'_1 \cup \mathcal{C}'_2 \ldots \cup \mathcal{C}'_k$. The distance between $\mathcal{C}$ and $\mathcal{C}'$ is:

$$\text{dist}(\mathcal{C}, \mathcal{C}') = \min \text{dist}(\mathcal{C}_1, \sigma(\mathcal{C}_1)) + \ldots + \text{dist}(\mathcal{C}_k, \sigma(\mathcal{C}_k))$$

where σ describes the $k!$ bijections from $\{\mathcal{C}_1, \ldots \mathcal{C}_k\}$ to $\{\mathcal{C}'_1, \ldots \mathcal{C}'_k\}$. This is a minimal matching problem: given the square $k \times k$ matrix of all distances between $\mathcal{C}_i$ and $\mathcal{C}'_j$, find k entries, one per line, one per column, with minimal sum. It is also a max flow min cost problem: each class of $\mathcal{C}$ and $\mathcal{C}'$ is represented by a vertex; all vertices of $\mathcal{C}$ are linked to all vertices of $\mathcal{C}'$, each edge $\mathcal{C}_i\mathcal{C}'_j$ has cost $\text{dist}(\mathcal{C}_i, \mathcal{C}'_j)$. A source vertex A is connected to all vertices of $\mathcal{C}$, with cost zero, and all vertices of $\mathcal{C}'$ are connected to a sink vertex Z also with cost zero. All edges have minimal capacity 0, maximal capacity 1. The problem is to find the maximal flow from A to Z, at minimal cost. This is a well known problem, and many algorithms exist. The optimal ones need $O(k^2 + \log k)$ time [17, 12].

5.3 Recognition of an Object

Recognition of a given shape $\mathcal{C}$ is finding the closest shape $\mathcal{S}$ in a given library of shapes $\mathcal{S}_1, \ldots \mathcal{S}_N$, and finding the differences between $\mathcal{C}$ and $\mathcal{S}$.

Our approach describes all shapes by their E-skeletons. To speed up the recognition of the shape $\mathcal{C}$, $\mathcal{C}$ is compared to each library shape at the first level of detail: only 1-partitions are compared. Some prescribed ratio, typically 90%, of the library shapes, the farthest from $\mathcal{C}$, are discarded. The remaining library shapes are then compared to $\mathcal{C}$ at the second level of detail, using their 2-partitions. Again, the worse 90% library shapes are eliminated, and so on, until it remains 10 or less library shapes: since each of these steps divides by ten the number of possible library shapes, $O(\log_{10} N)$ steps are needed to find the best library shapes. $\mathcal{C}$ is then compared to all remaining library shapes, with the maximal possible level of detail. The shape with smaller distance is the searched library shape $\mathcal{S}$.

5.4 Isomorphism

One may think that the E-skeletons of similar objects must be isomorphic. It suggests an alternate definition for recognition, and the following method, which we had no time enough to investigate.

Assume the bijection σ from the classes $C_i, i = 1 \ldots k$ of the k-partition of $\mathcal{C}$ to the classes $C'_i, i = 1 \ldots k$ of the k-partition of $\mathcal{C}'$ is known: initially it is true for $k = 1$. Let C_j be the split class in $\mathcal{C}$, C_j has child C'_j and C''_j. Then the split class in $\mathcal{C}'$ must be $\sigma(C_j)$; if $\sigma(C_j)$ is really the split class in $\mathcal{C}'$, we can extend the isomorphism between the two trees: if $\sigma(C_j)$ has child c', c'', then there is only two possibilities to extend σ for the $k+1$-partition: either $\sigma(C'_j) = c'$ and $\sigma(C''_j) = c''$, or $\sigma(C'_j) = c''$ and $\sigma(C''_j) = c'$. It is a very simple minimum matching problem, for a 2×2 matrix. The deeper two E-skeletons are isomorphic this way, the greater is their similarity.

5.5 About Chirality

A funny feature of our recognition procedure is that it finds equal two enantiomorph objects (*i.e.* differing by a symmetry, like the left and right hand), which makes sense in many medical applications.

It is not due to the existence of several natural coordinates systems. It is due to the fact that only various signatures for classes in the clouds $\mathcal{C}$ and $\mathcal{C}'$ are compared —the class signature $(\sqrt{\lambda_1}, \sqrt{\lambda_2}, \sqrt{\lambda_3}, r)$ in section 5.2.1— and they are invariant by symmetry.

Signatures for k-partitions (and not for classes, like in section 5.2.1) discriminating left and right are possible only if points coordinates of clouds are always expressed in coordinates system with the same orientation (for instance: with your head at $(0,0,0)$, your feet at $(0,0,-1)$, and looking at $(1,0,0)$, you have always the point $(0,1,0)$ on your left). Assuming that, the simplest signature for a $k \geq 4$-partition of a cloud $\mathcal{C}$, discriminating left and right, is as follows: consider 4 classes $C_1, \ldots C_4$ of $\mathcal{C}$, with gravity centers $G_1, \ldots G_4$, affinely independent (*i.e.* G_1, G_2, G_3, G_4 must be non coplanar, and even ”far from coplanar”). Then the determinant det(G_1, G_2, G_3, G_4), where G_i are expressed in homogeneous coordinates, is $3! = 6$ times the signed volume of the tetrahedron G_1, G_2, G_3, G_4, and it is the searched signature: it will discriminate left and right, whatever the choice of the natural coordinates systems. After matching this k-partition of $\mathcal{C}$ with the k-partition of $\mathcal{C}'$, we compute det(G_1, G_2, G_3, G_4) and det$(\sigma(G_1), \sigma(G_2), \sigma(G_3), \sigma(G_4))$, where σ is the bijection induced by the minimal matching: it they have the same sign, they have equal orientation, otherwise opposite. Thus it is possible to not confuse left and right, if needed.

5.6 Signatures for k-Partitions

So it is possible to define signatures for k-partitions: for instance, the set of all distances between the k gravity centers of the k-partition, the set of signed (or unsigned if we want to recognize modulo symmetry) volumes of tetrahedra. Once two k-partitions have been matched, it makes sense to compute the distance between two such signatures. Again, we did not need such sophisticated tools.

5.7 Recognition Examples

To test the recognition capability of the E-skeleton, we made a reference library with eight carpal bones. We then compare 4 exams (32 bones) of various patients with library E-skeletons. Table 2 is an excerpt of the distance matrix for hamatum, capitatum and lunatum for the five patients (values have been rounded). It is very easy to see on the matrix that bones have been correctly recognized.

Another test involved upper and lower jaws of an adult: all teeth have been compared, and left and right teeth have been matched correctly (upper and lower teeth are not symmetrical). For one tooth (an incisive), the symmetrical tooth has been identified successfully, but with a greater distance: it was due to a real pathology on the tooth.

The same test was also performed on left and right femur of an adult: due to the chirality independence of the class signature, the two bones were matched.

6 Extraction of Relevant Parameters

From the E-skeleton, it is possible to extract other parameters than those used for the signature. For example, an analysis editor was developed to find correlation between organs (these correlations are called *allometries*), and between organs and various criteria like age, gender or genetic parameters. This editor allowed to measure the well-known correlation between the shape of the carpal bones and the age of the patient.

More precisely, the hamatum may have a protuberance called *hamulus* that appears when bones are matured. The E-skeleton shows this protuberance by the comparison between main axis of the E-skeleton at levels 1 and 2: for a young patient, main axis of the two ellipsoids at level 2 are roughly orthogonal to the main axis of the level one. For an adult, on the contrary, the angle of one of the two ellipsoids (which represents the hamulus) is about 20 degrees.

This editor also reveals that some correlation exists between hamulus maturation and the gender of the patient. Many other criteria were tested and eliminated thanks to the flexibility of the tool. Further details may be read in [10].

In this example, proposals of correlation tests were made by an expert (physician) but it should be possible, if enough data samples were available, to automatically find relevant correlations with data analysis methods, or even artificial intelligence techniques.

7 Conclusion

This model was integrated in a commercial medical application called *Corpus 2000* developed by the MEB team at CIRAD (http://www.cirad.fr/) for viewing and analyzing biological entities. This software is available on various platforms (presently SGI IRIX, HP UX and Windows NT4/95/98).

The E-skeleton model presented was used in a tree modeler, which proves that this model may be used in other application domains. The E-skeleton may be considered as a generic model, that combines compacity of the data structures and multi-level representation, interactive manipulation and visualization, and capacities of

Table 2: Distance matrix for hamatum, capitatum and lunatum

	H1	H2	H3	H4	H5	C1	C2	C3	C4	C5	L1	L2	L3	L4	L5
H1	0	8	3	3	3	67	65	64	63	65	54	53	52	53	53
H2	8	0	8	7	7	69	67	66	66	67	54	53	52	54	53
H3	3	8	0	4	4	67	65	64	63	65	53	52	52	52	52
H4	3	7	4	0	1	68	66	65	64	66	53	53	51	53	53
H5	3	7	4	1	0	68	66	65	65	66	53	53	52	53	53
C1	67	69	67	68	68	0	4	5	6	3	68	66	68	68	66
C2	65	67	65	66	66	4	0	6	7	3	66	64	65	65	64
C3	64	66	64	65	65	5	6	0	3	5	68	66	68	67	66
C4	63	66	63	64	65	6	7	3	0	6	67	65	66	66	65
C5	65	67	65	66	66	3	3	5	6	0	67	65	66	66	65
L1	54	54	53	53	53	68	66	68	67	67	0	5	2	3	4
L2	53	53	52	53	53	66	64	66	65	65	5	0	4	3	1
L3	52	52	52	51	52	68	65	68	66	66	2	4	0	2	3
L4	53	54	52	53	53	68	65	67	66	66	3	3	2	0	2
L5	53	53	52	53	53	66	64	66	65	65	4	1	3	2	0

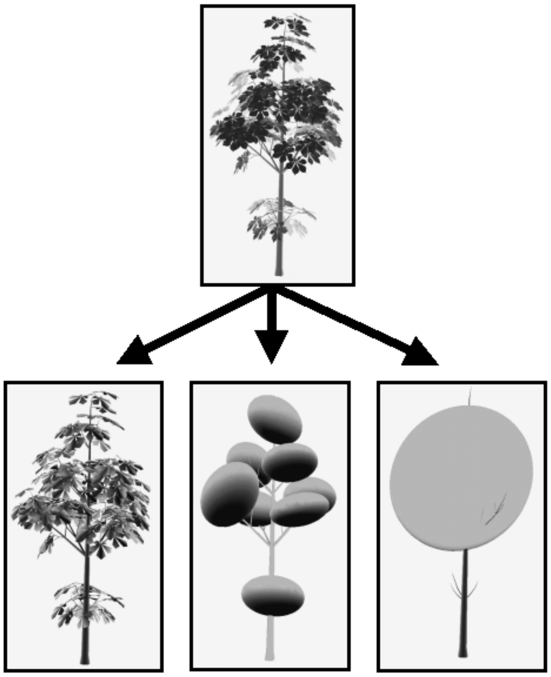

Figure 9: Multi-level representation of a tree.

automatic matching. Depending on the application domain, it also helps experts to find relevant parameters or correlations.

Further developments could involve:

- extension of geometric primitives for surface reconstruction;
- improvement of the E-skeleton by inserting a spring-mass pair for each ellipsoid or primitive, allowing extensions towards dynamic models or physically-based models;
- automatic detection of correlations or allometries inside data;
- automatic allometry-based synthesis of organs;
- construction of an atlas of anatomical shapes at several ages, including shapes with pathologies to be recognized;
- applications in CAD, like feature-based recognition and modelling;
- use of feature recognition to help segmentation of original images.

References

[1] A. Jaklič A. Leonardis and F. Solina. Superquadrics for segmenting and modeling range data. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 19, pages 1289–1295. IEEE, November 1997.

[2] A. Sourin A. Pasko, V. Adzhiev and V. Savchenko. Function representation in geometric modelling : concepts, implementation and applications. In *The Visual Computer*, pages 429–446. Springer-Verlag, 1995.

[3] F. Banégas. *Caractérisation et reconstruction de solides tridimensionnels par squelette ellipsoidal*. PhD thesis, Ecole Nationale des Mines de St-Etienne, 158 cours Fauriel, 42023 St Etienne cedex 2, France, 2000.

[4] A. Barr. Global and local deformations of solid primitives. In *Computer Graphics*, volume 18, pages 21–30. ACM Press, 1984.

[5] I. Biederman. Recognition-by-components: a theory image human understanding. *Psychological review*, 94:115–147, 1987.

[6] E. Bittar, N. Tsingos, and M.P. Gascuel. Automatic reconstruction of unstructured 3d data: combining a medial axis and implicit surfaces. In F. Post and M. Gobel, editors, *Eurographics '95*, volume 14, 1995.

[7] J.F. Blinn. A generalization of algebraic surface drawing. In *ACM Trans. on Graphics*, volume 1, pages 235–256. ACM Press, 1982.

[8] J. D. Boissonnat. Geometric structures for three-dimensional shape representation. In *ACM Transactions on Graphics*, volume 3, pages 266–286. ACM Press, 1984.

[9] J. D. Boissonnat. Shape reconstruction from planar cross sections. In *Computer Vision, Graphics and Image Processing*, 44, 1988.

[10] F. Canovas, F. Banégas, C. Cyteval, M. Jaeger, A. Dimeglio, C. Sultan, and F. Bonnel. Carpal bone maturation assessment by image analysis from ct-scans. *Journal of radiology*, 2000. to be published.

[11] L.D. Cohen. On active contour models and balloons. *Computer Vision Graphics Image Proceedings*, 53:211–218, 1991.

[12] T. Cormen, C. Leiserson, and R. Rivest. *Foundations of Computer Science*. MIT Press, Cambridge, Massachusets, 1990.

[13] P. Bertolino D. Attali and A. Montanvert. Using polyballs to approximate shapes and skeletons, 1994.

[14] G. Danuser and M. Stricker. Parametric model fitting : From inlier characterization to outlier detection. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 20, pages 263–280. IEEE, March 1998.

[15] E. Diday. Une nouvelle méthode en classification automatique et reconnaissance des formes : la méthode des nuées dynamiques. In *Rev. Statist. Appl.*, volume 19, pages 19–33, 1971.

[16] E. Ferley, M.P. Gascuel, and D. Attali. Skeletal reconstruction of branching shapes. In *Implicit Surfaces '96*, Eindhoven (The Netherlands), October 1996.

[17] Z. Galil. Efficient algorithms for finding maximum matching in graphs. *Computing Surveys*, 18(1), march 1986.

[18] F. Glover. Tabu search; part I. *ORSA Journal on Computing*, 1(3):190–206, 1989.

[19] F. Glover. Tabu search; part II. *ORSA Journal on Computing*, 2(1):4–32, 1989.

[20] H. Hoppe. Progressive meshes. In *Siggraph '96*, pages 99–108. ACM Siggraph, 1996.

[21] R.E. Kass and A.E. Raftery. Bayes factor. *Journal of the American Statistical Association*, 90:773–795, 1995.

[22] W.E. Lorensen and H.E. Cline. Marching cubes: a high resolution 3d surface algorithm. In *Computer Graphics*, volume 21, pages 163–169. ACM Press, july 1987.

[23] B.S. Morse, S.N. Pizer, and C.A. Burbeck. General shape and specific detail: context-dependent use of scale in determining visual form. In *Second International Workshop on Visual Form*, pages 374–383. World Scientific, 1994.

[24] S. Muraki. Volumetric shape description of range data using "blobby model". In *Computer Graphics*, volume 25, pages 227–235. ACM Press, July 1991.

[25] M. Näf, O. Kübler, R. Kikinis, M.E. Schenton, and G. Székely. Characterization and recognition of 3d organ shape in medical image analysis using skeletonization. In *IEEE Workshop on Mathematical Methods in Biomedical Image Analysis*, San Francisco (USA), June 1996.

[26] I. Pitas and A.N. Anastasios. Morphological shape decomposition. *IEEE PAMI*, 12(1):38–45, 1991.

[27] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C, the Art of Scientific Computing*. Cambridge University Press, 1992.

[28] A. Ricci. A constructive geometry for computer graphics. *The Computer Journal*, 16:157–160, May 1973.

[29] J. Rossignac and P. Borrel. Multi-resolution 3d approximation for rendering complex scenes. In B. Falcidieno and T.L. Kunii, editors, *Modelling in Computer Graphics*, pages 455–465. Springer Verlag, 1993.

[30] W. Schroeder, J. Zarge, and W. Lorensen. Decimation of triangle meshes. In *Siggraph '92*, volume 26, pages 65–70. ACM Siggraph, July 1992.

[31] S. Sclaroff and A. Pentland. Generalized implicit functions for computer graphics. In *Computer Graphics*, volume 25, pages 247–250. ACM Press, July 1991.

[32] S. Sclaroff and A. Pentland. Model matching for correspondance and recognition. *IEEE PAMI*, June 1995.

[33] A.J. Scott and M.J. Symons. Clustering methods based on likelyhood ratio criteria. *Biometrics*, 27:387–397, 1971.

[34] E.P. Simoncelli. A rotation invarariant pattern signature. In *IEEE International Conference on Image Processing*, volume 3, pages 185–188, Lausanne (Switzerland), September 1996.

[35] B. Soroka, R. Andersson, and R. Bajcsy. Generalized cylinders from local aggregation of sections. In *Pattern Recognition*, volume 13, 1981.

[36] G. Taubin. Distance Approximation for Rasterizing Implicit Curves. *ACM Transactions on Graphics*, 13:3–42, 1994.

[37] G. Taubin, F. Cukierman, S. Sullivan, J. Ponce, and D.J. Kriegman. Parameterized families of polynomials for bounded algebraic curve and surface fitting. *IEEE PAMI*, 6:287–303, March 1994.

[38] C. Lefèvre V. Burdin, C. Roux and E. Stindel. Modeling and analysis of 3-d elongated shapes with application to long bone morphometry. In *IEEE Transactions on Medical Imaging*, volume 15, pages 79–91. IEEE, February 1996.

[39] O. G. Okunev V. V. Savchenko, A.A. Pasko and T. L. Kunii. Function representation of solids reconstructed from scattered surface points and contours. In *Eurographics '95*. Blackwell Publishers, 1995.