International Journal of Computer Information Systems and Industrial Management Applications.
ISSN 2150-7988 Volume 3 (2011) pp. 568-577


# A Repartition Structure for Collision Detection and Deformation of Discrete Objects Based on 3D Wavelets

**Xavier Heurtebise[1] and Sébastien Thon[1]**

[1] LSIS: Laboratoire des Sciences de l'Information et des Systèmes
IUT de l'Université de Provence, Rue Raoul Follereau, 13200 Arles – France
*{xavier.heurtebise, sebastien.thon}@univ-provence.fr*

***Abstract*: In a virtual sculpture project, we would like to sculpt in real-time 3D objects sampled in volume elements (voxels). The drawback of this kind of representation is that a very huge number of voxels is required to represent large and detailed objects. Consequently, the memory cost will be very large and the user/object interaction will be slowed down. We proposed a multiresolution representation of 3D objects thanks to a 3D wavelet transform, in order to reduce the memory cost and to adapt the processing and display times with a desired level of detail. In order to allow real-time performance during the sculpting process, we propose in this paper a repartition structure, which is an octree whose each node contains maximal and minimal density for each area of the 3D object. Moreover, we combine this structure with a multiresolution collision detection, to accelerate the sculpting process during the addition and subtraction of matter into the 3D object thanks to a tool, both with the same multiresolution representation.**

***Keywords*:** virtual sculpture, discrete wavelet, voxels, collision detection, levels of detail, min/max octree.

## I. Introduction

In this paper, we present a multiresolution model based on 3D wavelets to represent a 3D object as a discrete set of volume elements (voxels). Such a discrete representation is of great use in:

- medical imaging: management of MRI or CT-scan data;
- surgical simulations: bone surgery, dental surgery;
- scientific simulations: simulation of heterogeneous systems such as semiconductor device simulation, molecular dynamics, plasma physics, and fluid mechanics;
- virtual sculpture: easy simulation of sculpture operations such as addition or subtraction of material by simply adding or removing voxels.

The major issue of discrete representations of volumes is the huge memory cost of very large 3D objects. proposed a multiresolution sculpture system based on 3D Haar wavelets [1]. A major advantage of this model is that sculpted objects can then be used as new tools, because the same model is used for both objects and tools. However, when 3D objects and tools are very large, during sculpture operations, the number of useful voxels of objects and tools can become very huge. Consequently, the sculpture operations become slower. In order to accelerate the sculpture operations, we propose in this paper several solutions:

- First, we propose a collision detection algorithm based on an octree, in order to refine the collision detection between objects and tools, both using the same model, based on 3D wavelets.
- Second, we use a min/max octree, that we called repartition structure, where each node contains maximal and minimal density for each area of the associated object.
- Third, we speed up the collision detection algorithm thanks to the repartition structure during sculpture operations.

## II. Previous work

In this paper, we would like to sculpt in real-time 3D objects with tools, both using the same model, based on 3D wavelets. In order to accelerate the sculpture operations, we would like a multiresolution collision detection algorithm. We present multiresolution representation in section II.A, in order to represent 3D objects and tools. Then, we describe in section II.B existing methods of virtual sculpture. Finally, we present in section II.C existing collision detection algorithms.

### *A. Multiresolution representations*

In this paper, we tackle the problem of virtual sculpture of a very large 3D object with a tool, both represented with spatial enumerations. Such a spatial enumeration is a set of volume elements called voxels, obtained by sampling the volume of a 3D object. It can be seen as a 3D image composed of voxels, while a 2D image is a bidimensional array composed of pixels.

To make a spatial enumeration from a 3D object, several methods have already been suggested. The simplest way is a uniform discrete spatial enumeration, by regularly sampling the object into voxels with the same size. However, a major drawback of this representation is the large number of voxels needed to represent very large objects with detailed features (for example, a 3D image in 1024×1024×1024 has more than one billion voxels). This entails three main problems. The first one is the important memory cost to store this uniform spatial

enumeration. The second one is that the display of these objects becomes slower. Finally, operations on these objects such as sculpture actions or displacements become less and less interactive.

To further improve the use of spatial enumeration, several methods of multiresolution representations have been proposed. Therefore, processing and display times are adapted with the desired level of detail. Among these methods, there are trees and 3D wavelet decomposition.

An octree can also be seen as a hierarchical representation of 3D object. The maximum level of subdivision of the octree defines the finest level of detail of a multiresolution representation. Boada et al. [1] define a section in an octree that determines the displayed nodes for a given level of detail. This method is extended to a "n-tree" by Ferley et al. [3].

The second multirésolution methods use bounding volume trees, used in collision detection. These methods propose to modify properties of the voxels, such as the size with AABB (axis-aligned-bounding-box) method [4], the orientation with OBB (oriented-bounding-box) method[5], or even the shape with sphere tree [6][7]. Thanks to these three methods, the object rendering is optimized because the original object shape can be approached with less voxels than with a simple uniform spatial enumeration, but to the cost of higher computation times.

The third multiresolution method uses wavelet transform. Wavelets are a mathematical tool for representing functions hierarchically. In our case, these functions are discrete 3D functions that define a set of voxels. Muraki [8] shows the use of 3D Haar wavelets [9] to represent a 3D object. Pinnamaneni et al. [10] build a 3D Haar wavelet decomposition from a sequence of 1D Haar wavelet decomposition in each direction of the 3D voxels grid. Wavelet decomposition allows us to display a 3D object faster according to the level of detail. It also permits to drastically cut down the memory cost, because high compression ratio can be achieved on wavelets coefficients, especially if lossy compression schemes are used. Heurtebise [11] uses same wavelet decomposition into a sequence of 1D wavelet transform in each direction of the 3D voxels grid, but he studied several wavelets, such as Haar wavelet, orthogonal Daubechies wavelet, bi-orthogonal Cohen-Daubechies-Feauveau wavelet…

### *B. Virtual sculpture*

In a voxel representation, several kinds of information can be stored, such as density of matter, color, material, hardness, elasticity… According to the kind of information, the mode of representation may be different.

A first volumetric method has been implemented by Galyean and Hughes [12]. Their model is defined with a uniform discrete spatial enumeration that contains density values. 3D tools modify the discrete potentials describing the object. Basic operations like addition or subtraction, and several tool definitions (heat gun, sand paper or color modifier) are proposed.

Ayasse and Müller [13] perform sculpture operations by the use of CSG (Constructive Solid Geometry). Complex objects are created by successive modifications of the material with a tool according to simple operations such as difference, union or intersection. However, the object and the tool are represented by simple uniform spatial enumerations. Moreover, voxels are limited to binary values (full or empty). The authors propose to reduce the computation time for each sculpture operation by using only the effective voxels according to a given displacement of the tool. However, they do not use a multiresolution representation to improve the display performance.

Raffin et al. [14] perform sculpture operation by moving the matter, into a 3D object, with a tool. Both of them are represented by a set of voxels. The authors proposed a method of diffusion of the matter by decomposing it along each axis x, y and z according the normal of collision point between the tool and the matter to be sculpted. They proposed also an algorithm to distribute the matter to surface neighbors, using a plasticity value of the object matter, in order to perform a more realistic result. The main drawback of this method is the size of the tool in order to have real-time performances.

Dewaele and Cani [15] proposed a deformable model for virtual clay. Sculpture operations are addition, subtraction and deformation of matter through the interaction with rigid tools. Their deformations, both large and small scale, mimic the effects of tools on real clay. The authors proposed two steps. The first one is the processing of the influence of static tools on the matter, by determining the displacement vector for each voxel of the matter. The second one is the displacement of the matter from each voxel of the object to its neighbours.

Angelidis et al. [16] presented sweepers, a new class of space deformations suitable for interactive and intuitive virtual sculpture. When an artist moves a tool, it causes a deformation of the working shape along the path of the tool: the authors used simple path (translation, scaling or rotation). Tools are simply shapes, subsets of 3D space. An advantage is that he artist can use shapes already created as new tools to make more complex shapes. Furthermore, more complex deformations are achieved by using several tools simultaneously in the same region. For representing shapes, Angelidis et al. [16] presented a mesh refinement and decimation algorithm that takes advantage of the definition of deformations.

In the Kizamu project, Perry and Frisken [17] use ADFs (Adaptively sampled Distance Fields) to model and to sculpt the material. A 3D object is sampled adaptively with a 3D grid according to the details of the object. Each grid cell contains a scalar specifying the minimum distance to the object shape. This distance is signed to distinguish the inside from the outside of the shape. Sculpture operations use CSG, but the Euclidean distance field used instead of density raises discontinuity problems and increases the update computation time.

Bærentzen and Christensen [18] propose the Level-Set method to deform the material. This method stores distance fields around the exterior of a 3D object. The single tool is a blob (a sphere) represented by an implicit function, which limits the sculpture capabilities.

To represent an object to be sculpted, Ferley et al. [3] also use distance fields, stored in a "n-tree" hierarchical representation where the sampling rate depends on object's details. The tool is limited to an ellipsoid defined by an

implicit function discretized to perform sculpture actions on the object: this limitation is very restrictive if the user wishes to use more complex tool shapes to perform specific sculpture actions. Raffin et al. [19] propose a hierarchical model of virtual sculpture based on an octree [20]. However, the tools, defined as voxels sets, remain parallel to the axis, which limits the orientation of the tool and the sculpture operations.

We proposed in [1] a multiresolution sculpture system based on 3D Haar wavelets. A major advantage of this model is that sculpted objects can then be used as new tools, because the same model is used for both objects and tools. Moreover, any orientation of tools [21] is possible, which does not limit the sculpture operations. Furthermore, we proposed in [22] an extension of this model, that combines octree and wavelet transform (a 3D object is roughly sampled in an octree, where each leaf containing data is thinly sampled thanks to a 3D wavelet transform), to manage very large volume datasets.

### *C. Spatial Collision Detection*

In order to deform the matter with a tool, we need to know if tools collide the matter. Two approaches can be used to determine the collision between tools and the matter: static collision detection (only the positions of tools and the matter are known at distinct instant) and dynamic collision detection (the movement of tools and object are considered: the positions of the tools and the matter are known at all instants).

The obvious approaches to collision detection for multiple and/or very detailed objects are very slow. Indeed, if we want to know the potential collision between N objects, checking the collision between every object against every other object is too inefficient to be used, because the complexity is quadratic, in $O(N^2)$. Moreover, if we want to know the potential collision between two objects with complex geometry, checking the collision between these objects each face against each other face, is itself quite slow.

Thus, considerable research has been applied to speed up the problem. Kitamura et al. [22], Hubbard [6], and O'Sullivan and Dingliana [24] used hybrid collision detection, with the following phases:

- Broad phase:
  - Phase 1: *progressive delimitation levels*. This phase restricts the subspaces with a potential by using hierarchies of space subdivisions.
  - Phase 2: *accurate broad levels*. In this phase, the approximate test is performed to identify interfering objects using a coarse representation of object shapes (such as bounding volumes).
- Narrow phase:
  - Phase 3: *progressive refinement levels*. Hierarchical approximations are suitable to well determine the object-parts in potential collision.
  - Phase 4: *exact level*. The tests use a tightly representation of object shapes to accurately identify any object parts, selected in the previous phase, that actually cause interference.

In the literature, several data structures are used to solve collision queries. We can classify these data structures in different categories depending on the criterion followed to model the workspace and objects.

For geometric models, collision, proximity and interference queries are computed by using the geometry of the two possible candidate objects. In the narrow phase at the exact level, the queries are formulated by using the geometry of objects. Dobkin and Kirkpatrick [25] proposed a collision detection algorithm between two polyhedrons in $O(\log^2 v)$ where $v$ is the average number of vertex of two polyhedrons. This method requires the convexity of polyhedrons, and only one collision point is given even if more collision points exist. Linear programming algorithms [26] allow checking collision between two convex polytopes if and only if there exists a separation plane between them. Feature based algorithms focus on the relationships between the sets of features (vertices, edges and faces) of the two polytopes. The algorithm of Lin-Canny [27] constructs the Voronoï Region (set of points closer to a feature than any other) of each feature. Then it computes the distance between the closest features of two polytopes to elucidate whether they collide. This algorithm takes $O(f)$ time, where f is the number of features. The method takes advantage of coherence because closest features will not change significantly between two consecutive frames. Simplex based algorithms treat the polytopes as the convex-hull of a point set. Gilbert et al. [28] presented the GJK algorithm, which detects collisions and gives a measure of interpenetration. Cameron [29] developed the Enhanced GJK algorithm. Volume minimization based algorithms focus on the intersection volume of two objects. Faure et al. [30] proposed an image-based method of the intersection volume between two polyhedra, using surface rasterization in three orthogonal directions. Moreover, the authors proposed the integration of pressure forces over the pixels of the intersection volume, to compute forces applied to the vertices of the polyhedra. Their method handles deformable and rigid objects without any precomputation, by combining the speed of surface-based methods [31][32] with the robustness of distance-based methods [33][34].

For space partition trees, space decomposition techniques in a hierarchical ways are considered. In the broad phase at the progressive delimitation level, the queries are often formulated by using the space partition trees. However, these trees are also used to refine collision detection between two objects, during the narrow phase at the progressive refinement levels. Examples of space partition hierarchies include regular grids of voxels [30], octrees [6][22], BSP-trees [35], kD-trees and their extensions [36][37]. The advantage of regular grids of voxels is the computation time in O(1) for the collision detection, but the major drawback is the huge memory cost of this method, for very detailed workspace. The advantage of octrees is the reduced memory cost in relation to the previous method. The computation time is proportional to the number of levels of subdivision of the octrees. For BSP-trees and kD-trees [35][36][37], the efficiency depends on the size and/or the number of levels of subdivision of the tree. The main drawback is the build of the tree in $O(N^2)$, where $N$ is the number of objects.

For bounding volumes (BV), the objects are enclosed in volumes of simple geometry such as spheres, AABBs, OBBs,

k-DOPs and convex hulls. BVs are used during the broad phase at accurate broad levels, in order to check easily and quickly collision between two objects. However, BVs are often used in hierarchical volume representations, called bounding volume trees: sphere trees [6][7], AABB-trees [4], OBB-trees [5], k-DOP trees [38] and convex hull trees [26]. BV trees are used in the narrow phase at progressive refinement levels. Klosowski [38] quantized the computation time for collision detection. They demonstrated that the thinness of BV influences the number of collision tests between each pair of BVs. Moreover, the simplicity of BV influences the efficiency of the intersection test between two BVs.

## III. Our original contribution

Many models have already been proposed in order to represent 3D objects as discrete sets of voxels. We have already proposed in previous papers a discrete representation to reduce the memory cost and display times, which become very huge when 3D objects are very large.

In this paper, we propose to solve the problem of computation times, which become very important, when 3D objects are very large and/or very detailed. For that, we propose the use of a min/max octree that we called “repartition structure”. Indeed, it allows us to easily and quickly determine the maximal and minimal densities of an area of a 3D object. The main contribution of this paper is to use this structure to enhance the collision detection between a tool and the matter, during the sculpture operations: as the tool is moved by the user, its voxels are in different local frame than the sculpture’s one. Such operations of addition and subtraction of material are then simply performed by modifying density values of voxels.

The remainder of the paper is organized as follows. In section IV, we describe the multiresolution model based on 3D wavelets, proposed in a previous paper. In section V, we describe our repartition structure. Then, in section VI, we propose a collision detection algorithm between objects and tools, both using the same multiresolution model. In section VII, we present dynamical and interactive modifications of 3D objects thanks to sculpting tools. Then, we conclude and present future works.

## IV. Multiresolution model

In a virtual sculpture project, we represent the 3D objects to be sculpted as a discrete set of voxels to easily handle subtraction, addition and displacement of matter by tools. Each voxel contains a density value coded in a byte (from 0 for an empty voxel to 255 for a full one). However, a uniform spatial enumeration is expensive in processing and display times.

Thus, we use our multiresolution model based on 3D Haar wavelets [1], based on the hierarchical structure proposed by Pinnamaneni et al. [10]. This principle can be extended to other discrete wavelet transform [11], such as orthogonal wavelets, bi-orthogonal wavelets, interpolated wavelets… For each level of details, the 1D wavelet transformation is applied in x-, y- and z-direction successively (Figure 1). For each transformation step, we have a block ‘L’ with low-resolution coefficients obtained by a low-pass filter, and a block ‘H’ with detail coefficients obtained by a high-pass filter.

We display this discrete object with the Marching Cubes algorithm [39] that provides a smooth surface instead of a set of blocky voxels. During display, they take the advantage of the multiresolution nature of the model given by the 3D wavelets to display the more appropriate level of details according to the situation (distance between the object and the point of view, needed frame rate). Moreover, we implemented a data cache [1] that improves performances by storing in memory the useful levels of detail, in order to avoid extracting the level each time we need to use it.

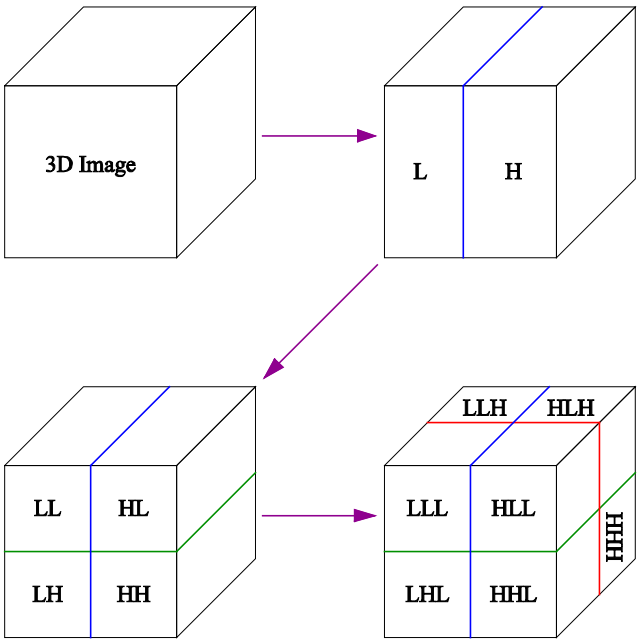



**Figure 1.** 3D Haar wavelet transform.

## V. The repartition structure

In this paper, we use this same model to represent both objects and tools. So, we will take the advantage of the multiresolution representation to accelerate the processing times, during the collision detection and the sculpting process.

Indeed, consider an area of the 3D object to be sculpted. If the density values at a given level of detail n allow us to deduce that no sculpture operation is possible in this area, we can stop the refinement processing steps of the collision detection at the level of detail n. However, the density values at the level of detail n do not allow us to define the interval of density values (from 0 to 255) at the rougher level of detail $n - 1$, except if we use the detail coefficients of the wavelet transform. So, in order to define the density repartition of an area of $N$ voxels of the 3D object at a level of detail, we need to know the following data at a finer level of detail: the density repartition of this area ($N/8$ voxels), and all the detail coefficients ($7N/8$ detail coefficients) of the wavelet transform in this area. Consequently, in order to define the density repartition of an area of $N$ voxels of the 3D object, at a level of detail, we need to know $N$ coefficients, at a finer level of detail.

In order to reduce the processing time to define the density repartition of an area of a 3D object, we propose the use of a min/max octree, called repartition structure. This structure is

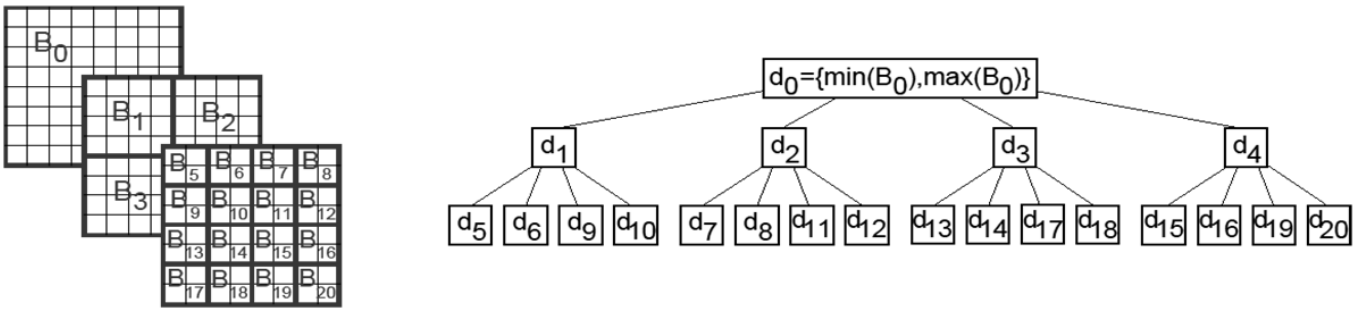



**Figure 2.** Repartition structure, in the 2D case: a quadtree (on right) contains the minimal ($\min(B_i)$) and maximal ($\max(B_i)$) values of the densities for each zones $B_i$ of the 2D image (on left).

an octree that stores the density repartition of this 3D object, at different levels of detail. Each node contains two values: the maximal and minimal densities of an area of the 3D object contained in this node. Figure 2 illustrates this structure in the 2D case thanks to a quadtree: this principle is the same in 3D thanks to an octree. As shown on figure 2, the quadtree is completely developed: the leaves are the areas, with a size of 2×2×2 voxels, and the root is the area, with a size of n×n×n voxels (with $n = 2^p$ and $p$ is the number of levels of detail of the 3D object).

Moreover, for each non-terminal node (i.e. each node different from a leaf) of this octree, the minimal density (respectively the maximal density) is the smallest (respectively the greatest) of the minimal densities (respectively the maximal densities) of all the child nodes. For example of figure 2, the minimal density of the area $B_1$ is defined by: $\min(B_1) = \min(\min(B_5), \min(B_6), \min(B_9), \min(B_{10}))$.

As the octree is completely developed with a structure set by the size of the 3D object, we can represent this octree with a vector V, whose handling is easier, faster and cheaper in memory:

- The first element, at the position 0, is the root of the octree.
- The size of the vector $V$ is $N = (n^3 - 1) / 7$, where n is the size of the 3D object.
- The size of the repartition structure is $S = 2N \times d$, where $d$ is the number of bytes to store a density value (for example, 1 for 'char', 2 for 'short int' and 4 for 'long int', on 32-bits operating system).
- The element, at the position $i < \text{INT}((N - 1)/8)$, has eight children from the position $j_1 = 8i + 1$ to the position $j_8 = 8i + 8$. The operator 'INT($x$)' correspond to the integer part of '$x$'
- The element, at the position $i > 0$ has one parent at the position $j = \text{INT}((i - 1)/8)$.
- The level of detail p of the octree contains the elements from the position $a_p = 1 + 8 + \ldots + 8p - 1$ to the position $b_p = a_p + 8p - 1$. Recursively, we have: $a_1 = 2$; if $p > 0$, $b_p = 8a_p$; if $p > 1$, $a_p = b_{p-1} + 1$.

The octree structure allows us to move easily from a node to its parent or its children into the repartition structure, as each node is set. The build of the repartition structure is easy and starts from the leaves to the root. For each leaf of the octree, we set the minimal and maximal densities thanks to the 3D objects at the finest level of detail. Once all the leaves are filled, we update the repartition structure from the leaves to the root.

During the sculpture process (see section VII), the repartition structure will be updated according to the modified voxels of the 3D object. If one voxel is modified, the minimal or maximal density of the corresponding leaf of the octree will be modified, and we update the repartition structure from this leaf to the root of the octree. However, if several voxels are modified, the update of the octree will be made in one pass, rather than density by density.

## VI. Our collision detection algorithm

In section II.C, we have presented the classification given by O'Sullivan and Dingliana [24]. For the broad phase, there exist several performing algorithms to check quickly collision between $N$ objects and to determine the candidate pairs of objects in collision. Among these algorithms, we will use the "*sweep and prune*" algorithm [40] that is very efficient. For the narrow phase, we will use the hierarchical structure of multiresolution model of the 3D objects, in order to refine progressively the collision detection from the entire object (the roughest level of detail) to the different voxels (the finest level of detail).

### *A. The complete hierarchical collision detection*

In order to refine the collision detection during the narrow phase thanks to 3D wavelet transform, we propose a very easy method, based on the collision detection thanks to octree method, with following conditions:

- Each node of the octree becomes a voxel at a given level of detail of 3D wavelet transform.
- The eight children of a node of the octree become the eight underlying voxels at a more detailed level of detail of 3D wavelet transform.
- The root of the octree becomes the voxel at the roughest level of detail of 3D wavelet transform.
- A voxel whose density value is lower than threshold $T$, is considered as empty, else it is considered as full or partially full. It is necessary that threshold $T$ is very small in relation to the maximal density.

The main drawback of this algorithm is the necessity to go to the finest level of detail of each object in order to check collision between two voxels for each object. Indeed, if two objects are very detailed, the computation time will be very high, and the collision detection can be non-interactive. In order to accelerate the collision detection, we propose to use not only a bounding volume of each voxel, but also the repartition structure (described in section V).

*Algorithm 1.* Simple collision detection algorithm between two blocks.

```
Function COLL (voxels: V_Ti, V_M; LOD: lod_Ti, lod_M)
 If V_Ti collides V_M
 Then
   Variable V'_M
   If (Op = 'subtraction of matter')
   Then V'_M = V_M
   Else V'_M = Max density – V_M

   If (V_Ti ≥ T and V'_M ≥ T) then
      If (lod_Ti = 0 and lod_M = 0) then
         Add V_Ti to the list L_Ti, and V_M to the list L_M
      Else if (lod_Ti = 0 or (lod_M ≠ 0  and size(V_M) > size(V_Ti))) then
         For each V_M[k] ⊂ V_M, with LOD lod_M-1Do COLL (V_Ti, V_M[k], lod_Ti, lod_M–1)
      Else
         For each V_Ti[k] ⊂ V_Ti, with LOD lod_Ti–1  Do COLL (V_Ti[k], V_M, lod_Ti–1, lod_M)
```

### *B. The choice of bounding volume (BVs)*

The choice of BVs for the detection collision is very crucial, because the computation time depends on this choice.

Each 3D object is sampled in a set of voxels that are cubes. If we check collision between two 3D objects, sampled in two sets of cubes, the simplest method of collision detection uses the OBB to represent each cube, in their respective orientation. However, the computation time of collision detection between two OBBs is greater than two AABBs or two spheres: 200 operations needed to perform the interference test between two OBBs versus 6 for AABBs and 8 for spheres. Therefore, we prefer to use AABB or sphere to represent each voxel in order to reduce the computation time of collision detection. If we use AABBs to represent each voxel of 3D objects, it is necessary to compute again the AABBs for each voxel when 3D object rotates. In order to allow the rotation of 3D objects, we prefer using spheres to represent each voxel during the collision detection.

However, if we want to reduce the most possible the computation time and to allow any orientation of 3D objects, we use the smallest AABB of the bounding sphere for each voxel during collision detection, in despite of the tightness of the detection. Indeed, for any orientation of 3D objects, the AABB of the bounding sphere for each voxel does not change. Moreover, the computation time of collision detection between two spheres is greater than two AABBs of two spheres: 8 operations needed to perform the interference test between two spheres versus 6 for AABBs.

### *C. Enhanced hierarchical collision detection*

We propose an enhanced hierarchical collision detection algorithm, based on our repartition structure. Consider $N_i$ the node of the repartition structure of the 3D object *i*, for each level of detail $LOD_i$. If the BVs of $N_1$ and $N_2$ have no intersection, there is no collision: refinement of collision detection is useless.

Consider the case of the BVs of $N_1$ and $N_2$ are in collision. If repartition structures give for $N_1$ or/and $N_2$ maximal densities that do not exceed the threshold *T*, there is no collision. On the other hand, if repartition structures give for $N_1$ and $N_2$ minimal densities that exceed the threshold *T*, there is necessarily collision for the voxels of each object at the finest level of detail, contained in the intersection of the BVs of $N_1$ and $N_2$. In these two cases, refinement of collision detection is useless. Else, we refine the smallest voxel in size between $N_1$ and $N_2$, if possible. However, if no refinement is possible for both $N_1$ and $N_2$, so all the voxels of two 3D objects, at the finest level of detail, are in the intersection of the BVs of $N_1$ and $N_2$.

With this method, it is no longer useful to know the 3D object for each level of detail, but only the repartition structure associated to the 3D object, which is cheaper in memory than the 3D object and its levels of detail, or even the 3D wavelet transform. Indeed, the repartition structure has a memory cost of $(n^3 – 1)/7$ where $n \times n \times n$ is the size of the 3D object (with $n = 2^p$ and $p$ is the number of levels of detail of the 3D object). Meanwhile, all the levels of detail of the 3D object have a memory cost of $(8n^3 – 1)/7$ and the 3D wavelet transform has a memory cost of $n^3$.

## VII. Virtual Sculpture

Our model can easily handle sculpture operations by simply adding or removing matter into a 3D object thanks to a tool, i.e. by modifying the density values of each voxel of the object to be sculpted. A major advantage of our method is that the tool used for virtual sculpture has the same representation than the matter. So, the user can create his or her own tools to sculpt another 3D object. The tools can take any orientation and any position in relation to the object [1].

### *A. Steps of the addition/subtraction of matter*

During the sculpture operations, such as addition and subtraction of matter thanks to a tool, a first phase allows to determine which voxels of the tool and the matter to be sculpted are in collision: this phase is called "*collision detection phase*". The following operations are performed:

- In a first step, we perform the collision detection, by using the "*sweep and prune*" algorithm to determine which tools $T_i$ collides the matter M to be sculpted.
- In a second step, we refine the collision detection between each tool $T_i$ and the matter *M* by using the algorithm 1. We obtain two lists $L_{Ti}$ and $L_M$ of voxels for tool $T_i$ and matter *M*, respectively. For this step, when we subtract the matter

*Algorithm 2.* Hierarchical collision detection algorithm between two blocks.

```
Function ENH_COLL (nodes: N_tool, N_obj; mode: Op)
 If N_tool collides N_obj
 Then
   Variable N_max, N_min
   If (Op = 'subtraction of matter')
   Then N_max = N_obj,max    and    N_min = N_obj,min
   Else N_max = Max density – N_obj,min    and    N_min = Max density – N_obj,max

   If (N_tool,max ≥ T) and (N_max ≥ T) then
      If ((N_tool is node or N_tool,min ≥ T) and (N_obj is node or N_min ≥ T)) then
         Add to the list C[tool,obj],
         all the pairs (V_tool,V_obj) ⊂ (N_tool ∩ N_obj), with LOD = 0
      Else if (N_obj is node or N_min ≥ T) then
         For each child N_tool[i] of node N_tool Do ENH_COLL (N_obj, N_tool[i], Op)
      Else if (N_tool is node or N_tool,min ≥ T) then
         For each child N_obj[i] of node N_object Do ENH_COLL (N_obj[i], N_tool, Op)
      Else if (size(N_tool) > size(N_obj)) then
         For each child N_tool[i] of node N_tool Do ENH_COLL (N_obj, N_tool[i], Op)
      Else
         For each child N_obj[i] of node N_obj Do ENH_COLL (N_obj[i], N_tool, Op)
```

with a tool, we check collision between each voxel $V_{Ti}$ of the tool $T_i$ with the dual $V'_M$ of the voxel $V_M$ of the matter $M$, whose value is equal to the maximal density minus the current density.

Then, we perform a "*sculpting phase*" whose operations are the following:

- In a first step, we apply the method of discrete rotation, presented by Heurtebise and Thon [21], only for the voxels in the intersection area, which is the AABB of the list $L_{Ti}$ of voxels for tool $T_i$. We obtain a new list $L_{R,Ti}$ of voxels of the tool after the discrete rotation.
- In a second step, for each voxel $V_M$ of the matter $M$, with $V_M$ included in $L_M$, we find which voxels $V_{Ti}$ of the tool $T_i$, with $V_{Ti}$ included in $L_{R,Ti}$, intersect it. Then we compute the new value of $V_M$ according to the selected sculpting mode and the filling percentage $D$ of the voxel $V_M$ by the voxels $V_{Ti}$ of the tool $T_i$. This voxel value is clamped to the maximal value (respectively the minimal value) during the operation of addition (respectively subtraction) of matter, because we have density values in a given interval.
  Figure 3 shows this principle in 2D where the values are given in percentage, i.e. 100% (respectively 0%) is used for the maximal density (respectively the minimal density). The principle is the same in 3D.
  - In the "*addition of matter*" mode, the filling percentage $D$ is given to the value of the voxel $V_M$ of the matter $M$, only if the filling percentage $D$ is greater than the value of the voxel $V_M$.
  - In the "*subtraction of matter*" mode, the empty percentage $D'$, equal to the difference between the maximal density and $D$, is given to the value of the voxel $V_M$ of the matter $M$, only if the empty percentage $D'$ is smaller than the value of the voxel $V_M$.
- In a third step, we modify the wavelet data according to this new voxel value. In order to accelerate this step, we can update the wavelet data after the modification of a set of voxels $V_M$ of the matter, instead of voxel by voxel.
- In a fourth step, we update the repartition structure according to the modified voxels.
- In a last step, for each level of detail, we use the Marching Cubes algorithm to rebuild the triangulated surface, only for the modified parts of the 3D object, corresponding to the matter $M$, to improve the computation time.

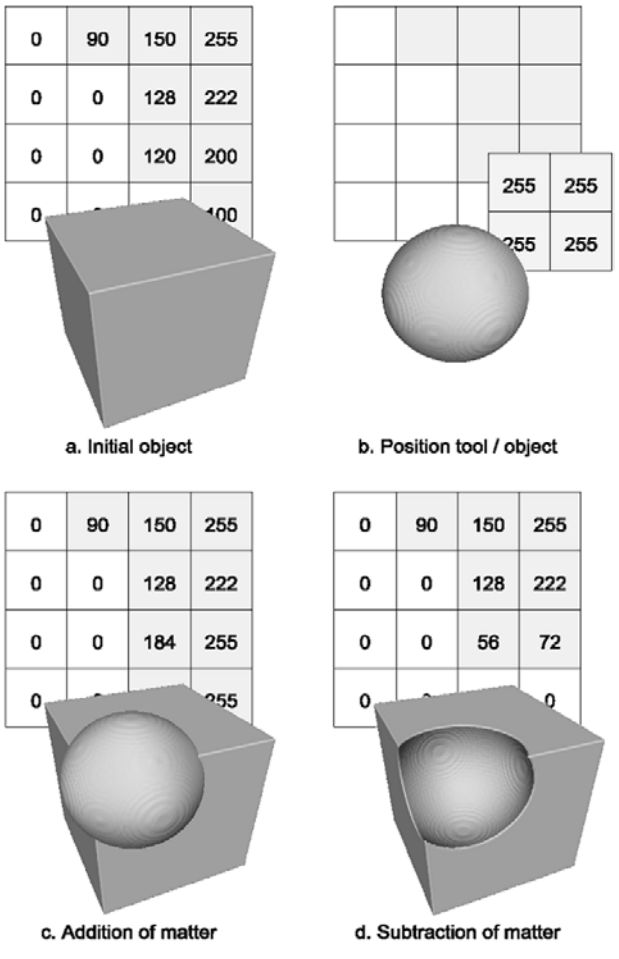



**Figure 3.** Addition (c) and subtraction (d) of matter to a cubic object (a) with a spherical tool (b).

### B. Enhancement of the addition/subtraction of matter

An enhancement of this sculpting method is the use of the repartition structure, in the algorithm 2, to apply sculpting operations not at the finest level of detail, but at a rougher level of detail, in order to reduce the computation times, if necessary. The interest is to permit to modify interactively the matter with a tool. In the second step of the collision detection phase, we stored in the list $C[T_i,M]$ the couples of voxels $(V_{ti},V_M)$ or $(V_{ti},V'_M)$ of the tool $T_i$ and the matter $M$, respectively, in collision at the finest level of detail of two objects.

### C. Experiments

The results given in this paper have been obtained on a PC with an *Intel Core 2 Duo 3 GHz with 4 GB memory, a NVidia GeForce 9800 GTX+ with 1 GB video memory, and Windows Seven 32 bits*.

To determine the performance of each algorithm in function of the size of the tool for an object in 512×512×512 voxels, we perform two kinds of tests: addition and subtraction of matter. For that, we create a filled (respectively empty) cube and a tool with a given shape (sphere), for subtraction (respectively addition) of matter. Then, we move the tool inside the 3D object with a given step and we compute the average times per frame of sculpture operations. For each algorithm, the sculpture times are given in table 1. In this table, we compare our algorithms with a virtual sculpture method without collision detection algorithm (called algorithm 0 in table 1) to refine the sculpture operations. In order to compare the collision detection algorithms, the sculpture time excludes the display times, which depends on the level of detail for display.

*Table 1.* Average sculpture times (in milliseconds) per frame, for addition and subtraction of matter, and several size of tool in $n\times n\times n = n^3$ voxels.

| **Algorithms** | **Addition of matter** | | | **Subtraction of matter** | | |
|---|---|---|---|---|---|---|
| | $32^3$ | $64^3$ | $128^3$ | $32^3$ | $64^3$ | $128^3$ |
| #0 | 28.5 | 137.7 | 1001 | 27.9 | 138.4 | 1027 |
| #1 | 38.9 | 195.9 | 1558 | 37.7 | 183.3 | 1572 |
| #2 | 29.3 | 103.8 | 782.2 | 29.2 | 102.5 | 781.2 |

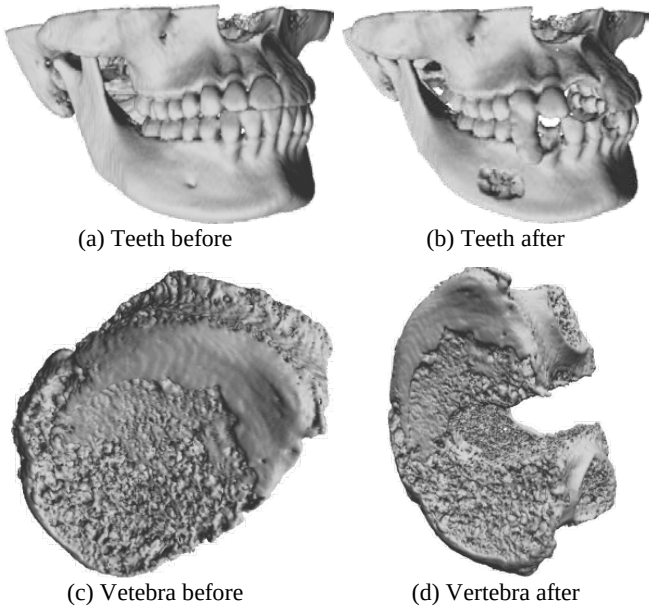
The table 1 shows that the sculpture times depend on the size of the tool and the collision detection algorithm. Indeed, if the size of the tool increases, the average number of voxels of the tool in collision with the matter increases, consequently the sculpture times increase.

Furthermore, the sculpture times, for the algorithm 1, are higher than for the algorithm 0 without collision detection. Indeed, the algorithm 1 uses a collision detection based on an octree method, where the detection is made on all the levels of detail of the object and the tool. So, the collision times become so high that the sculpture times, for the algorithm 1, is higher than in the case where no collision detection is used.

However, if we use the algorithm 2, based on our repartition structure, the sculpture times become smaller than the ones with the algorithm 1 or the algorithm 0 without collision detection. Indeed, it is not needed to detect the collision between a tool and the matter at the finest level of detail, because the algorithm 1 stops the refinement before the finest level of detail when minimal density is greater than a given threshold. Furthermore, as the repartition structure, based on an octree, is completely developed, the processing time for each displacement into the repartition structure is optimized.

On the other hand, the repartition structure associated to a 3D object is cheaper in memory than the 3D object and its levels of detail, or even the 3D wavelet transform. Indeed, the repartition structure has a memory cost of $(n^3 - 1)/7$ where $n\times n\times n$ is the size of the 3D object (with $n = 2^p$ and $p$ is the number of levels of detail of the 3D object). All the levels of detail of the 3D object have a memory cost of $(8n^3 - 1)/7$ and the 3D wavelet transform has a memory cost of $n^3$.

Consequently, the repartition structure allows us to reduce the memory cost and the sculpture times during the sculpture process, thanks to the use of detection collision algorithm based on the repartition structure.



(a) Teeth before (b) Teeth after

(c) Vetebra before (d) Vertebra after

**Figure 4.** Teeth (a) and (b), and human lumbar vertebral body L3 (c) and (d), both sculpted with several tools (sphere and cylinder, in different size between 8×8×8 and in 64×64×64), thanks to the two kinds of sculpture operation: addition and subtraction of matter. The collision detection algorithm used for these sculptures is the algorithm 2. The framerate during the sculpture operations and the display is upper than 5 FPS.

## VIII. Conclusion

We have presented in this paper a multiresolution model for virtual sculpture of 3D objects with tools, using 3D wavelet. The major drawbacks of multiresolution model [1] are the problem of computation times and the limitation of the memory size, which become very important, when 3D objects are very large and/or very detailed. In order to solve these problems, we proposed in the paper the use of a min/max octree that allows us to easily and quickly know the maximal and minimal densities of an area of a 3D object. Moreover, during the sculpture operation, we do not need to store all the levels of detail of our multiresolution model thanks to this structure. Then, our main contribution was the use of this structure to enhance the collision detection between a tool and

the matter to be sculpted, during the sculpture operations. Finally, we simply perform operations of addition and subtraction of material by modifying density values of voxels. To verify the applicability of our sculpting system, we have conducted many sculpting sessions, which have resulted in numerous interesting sculptures. Two examples are shown on figures 4(a) and 4(b): Dental Scan [41] in 512×512×167 voxels, and on figures 4(c) and 4(d): micro-CT scan of a human lumbar vertebral body L3 [42] in 244×512×512 voxels

## IX. Future work

Many improvements of our sculpture system are possible, by investigating open issues such as dynamical collision detection, other sculpture operations, realistic deformation, memory cost or computation time.

Our collision detection algorithm does not allow us to know if a tool collides the matter between two distinct instants. We plan to extend our collision detection algorithm to the spatio-temporal collision detection, in order to have realistic sculpting operation.

We will also investigate other sculpture operations, such as the displacement of the matter, and we plan to take more advantage of the levels of detail of the 3D Haar wavelet, in order to accelerate these sculpture operations. In order to obtain a realistic deformation of the matter by a tool, we plan to add other information into our model, such as viscosity or hardness.

Memory cost to store very large 3D objects in memory can be very huge. Consequently, we plan to use our repartition structure with the extended model, presented by Heurtebise et al. in [22], which combines octree and wavelets: a 3D object is roughly sampled in an octree, where each leaf containing data is thinly sampled thanks to a 3D wavelet transform.

## References


[1] X. Heurtebise, S. Thon, G. Gesquière. “Multiresolution Representation and Deformation of Wavelet-based 3D Objects”, In *Short Communication Proceedings of WSCG’06*, Plzen, Czech Republic, 2006.

[2] I. Boada, I. Navazo, R. Scopigno. “Multiresolution Volume Visualization with a Texture-Based Octree”, *The Visual Computer*, XVII(3), pp. 185-197, 2001.

[3] E. Ferley, M. P. Cany, J. D. Gascuel. “Resolution Adaptive Volume Sculpting”, *Graphical Models*, LXIII(6), pp. 459-478, 2001.

[4] G. V. D. Bergen. “Efficient Collision Detection of Complex Deformable Models using AABB Trees”, *Journal of Graphics Tools*, II(4), pp. 1-14, 1997.

[5] S. Gottschalk, M.C. Lin, D. Manocha. “OBBTree: A Hierarchical Structure for Rapid Interference Detection”, In *Proceedings of ACM SIGGRAPH*, pp. 171-180, 1996.

[6] P. M. Hubbard. “Collision Detection for Interactive Graphics Applications”, *IEEE Transactions on Visualization and Computer Graphics*, I(3), pp. 218-230, 1995.

[7] G. Bradshaw, C. O’Sullivan. “Adaptive Medial-Axis Approximation for Sphere-Tree Construction”, *ACM Transactions on Graphics*, XXIII(1), pp. 1-26, 2004.

[8] S. Muraki. “Approximation and Rendering of Volume Data using Wavelet Transforms”, In *Proceedings of the 3rd conference on Visualization'92*, pp. 21-28, 1992.

[9] A. Haar. *Zur Theorie der Orthogonalen Funktionensysteme*, Ph.D. thesis, Gottingen, Germany, 1909.

[10] P. Pinnamaneni, S. Saladi, J. Meyer. “3-D Haar Wavelet Transformation and Texture-Based 3-D Reconstruction of Biomedical Data Sets”, In *VIIP’01Proceedings of the International Association of Science and Technology for Development*, Marbella, Spain, ACTA Press, pp. 389-394, 2001.

[11] X. Heurtebise. *Représentation Multirésolution et Déformation d’Objets 3D Définis par Enumération Spatiale*, Ph.D. thesis, Arles, France, 2007.

[12] T. A. Galyean, J. F. Hughes. “Sculpting: an Interactive Volumetric Modeling Technique”, *ACM SIGGRAPH Computer Graphics*, XXV(4), pp. 267-274, 1991.

[13] J. Ayasse, H. Müller. “Interactive Manipulation of Voxel Volumes with Free-formed Voxel Tools”, In *Proceedings of the Vision Modeling and Visualization Conference*, pp. 359-366, 2001.

[14] R. Raffin, G. Thibault, G. Gesquière. “Simple and Efficient Tools for Virsculpt”, In *GRAPP’06 Proceedings of the first international conference on computer graphics theory and applications*, Lisboa, Portugal, pp. 436-440, 2006.

[15] G. Dewaele, M. P. Cani. “Interactive Global and Local Deformations for Virtual Clay”, *Graphical Models*, LXVI(6), pp. 352-369, 2004.

[16] A. Angelidis, G. Wyvill, M.-P. Cani. “Sweepers: Swept User-Defined Tools for Modeling by Deformation”, In *SMI’04 Proceedings of 2004 International Conference on Shape Modeling and Applications*, IEEE, Genoa, Italy, pp. 63-73, 2004.

[17] R. N. Perry, S. F. Frisken. “Kizamu: A System for Sculpting Digital Characters”, In *Proceedings of ACM SIGGRAPH*, pp. 47-56, 2001.

[18] J. A. Bærentzen, N. J. Christensen. “Volume Sculpting Using the Level-Set Method”, In *Proceedings of the Shape Modeling International*, pp. 175-182, 2002.

[19] R. Raffin, G. Gesquière, E. Rémy, S. Thon. “VirSculpt: A Virtual Sculpting Environment”, In *GraphiCon’04 Proceedings*, Moscow, pp. 184-187, 2004.

[20] J. A. Bærentzen. “Octree-based Volume Sculpting”, In *Proceedings of IEEE Visualization’98*, IEEE CS Press, pp. 9-12, 1998.

[21] X. Heurtebise, S. Thon. “Discrete Tools for Virtual Sculpture”, In *GRAPP’06 Proceedings of the first international conference on computer graphics theory and applications*, Lisboa, Portugal, pp. 436-440, 2006.

[22] X. Heurtebise S. Thon. “Multiresolution Representation and Deformation of very Large Volume Datasets based on Haar Wavelets”, In *GMAI ‘08 Proceedings of the 2008 3rd International Conference on Geometric Modeling and Imaging*, IEEE Computer Society, pp. 34-40, 2008.

[23] Y. Kitamura, A. Takemura, F. Kishino. “Efficient Collision Detection among Objects in Arbitrary Motion using Multiple Shape Representations”, In *Proceedings of the 12th IAPR International Conference*, Jerusalem, Israel, pp. 390-396, 1994.

[24] C. O'Sullivan, J. Dingliana. "Real-time Collision Detection and Response using Sphere-trees", In *Proceedings of Spring Conference on Computer Graphics*, pp. 83-92, 1999.

[25] D. P. Dobkin, D. G. Kirkpatrick. "Determining the Separation of Preprocessed Polyhedra - A Unified Approach", In *ICALP'90 Proceedings of the 17th International Colloquium on Automata, Languages and Programming*, London, UK, pp. 400-413, 1990.

[26] R. Seidel. "Linear Programming and Convex Hulls made Easy", In *Proceedings of the 6th annual ACM symposium on Computational Geometry*, New York, USA, pp. 211-215, 1990.

[27] M.C. Lin, J.F. Canny. "A Fast Algorithm for Incremental Distance Calculation", In *Proceedings of IEEE International Conference on Robotics and Automation*, Sacramento, CA, USA, pp. 1008-1014, 1991.

[28] D. Gilbert, D. W. Johnson, S. S. Keerthi. "A Fast Procedure for Computing the Distance between Objects in Three-Dimensional Space", In *Proceedings of IEEE International Conference on Robotics & Automation*, 4(2), pp. 193-203, 1988.

[29] S. A. Cameron. "Enhancing GJK: Computing Minimum and Penetration Distances between Convex Polyhedral", In *Proceedings of International Conference on Robotics & Automation*, pp. 3112-3117, 1997.

[30] F. Faure, S. Barbier, J. Allard, F. Falipou. "Image-based Collision Detection and Response between Arbitrary Volumetric Objects", In *SCA'08 Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Dublin, Irland, pp.155-162, 2008.

[31] G. Hirota, S. Fisher, A. State, C. Lee, H. Fuchs. "An Implicit Finite Element Method for Elastic Solids in Contact", In *Proceedings of Computer Animation 2001*, Seoul, pp. 136-146, 2001.

[32] P. Volino, N. Magnenat–Thalmann. "Resolving Surface Collisions through Intersection Contour Minimization", ACM Transactions on Graphics, XXV(3), pp. 1154-1159, 2006.

[33] B. Heidelberger, M. Teschner, R. Keiser, M. Müller, M.H. Gross. "Consistent Penetration Depth Estimation for Deformable Collision Response", In *Proceedings of VMV'04*, pp. 339-346, 2004.

[34] J. Barbič, D. L. James. "Time-critical Distributed Contact for 6-DoF Haptic Rendering of Adaptively Sampled Reduced Deformable Models", In *SCA'07 Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, San Diego, CA, USA, pp. 171-180, 2007.

[35] B. Naylor, J. Amanatides, W. Thibault. "Merging BSP Trees Yields Polyhedral Set Operations", In *Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, XXIV(3), pp. 115-124, 1990.

[36] M. Held, J. T. Klosowski, and J. S. B. Mitchell, "Speed Comparison of Generalized Bounding Box Hierarchies", Technical report, Department of Applied Math, State University of New York at Stony Brook, 1995.

[37] S. Redon. *Algorithms for Interactive Dynamics Simulation of Rigid Bodies*, Ph.D. thesis, Evry, France, 2002.

[38] J. T. Klosowski. *Efficient Collision Detection for Interactive 3D Graphics and Virtual Environments*, Ph.D. thesis, State University of New York at Stony Brook, 1998.

[39] W. E. Lorensen, H. E. Cline,. "Marching Cubes: A High Resolution 3D Surface Construction Algorithm", *Computer Graphics*, XXI(4), pp. 163-169, 1987.

[40] J. D. Cohen, M. C. Lin, D. Manocha, M. K. Ponamgi. "I-COLLIDE: an Interactive and Exact Collision Detection System for Large-scale Environments", In *SI3D'95 Proceedings of the 1995 symposium on Interactive 3D graphics*, pp. 189-196, 1995.

[41] DICOM Sample Image Sets (OSIRIX). http://pubimage.hcuge.ch:8080/

[42] G. Beller, M. Burkhart, D. Felsenberg, W. Gowin, H.C. Hege, B. Koller, S. Prohaska, P. I. Saparin, J. S. Thomsen. Vertebral Body Data Set ESA29-99-L3, http://bone3d.zib.de/data/2005/ESA29-99-L3/

## Author Biographies

**Xavier Heurtebise** is a temporary researcher in Computer Science at the University of Provence, France, since 2007.

He joined the LSIS Laboratory, Marseille, France, in 2007. His research focuses on multiresolution representation and deformation of discrete 3D objects. He received his Ph.D. in Computer Science at the University of Provence, France, in 2007. He graduated from the Ecole Nationale Supérieure de Cachan, France, in 2003 and obtained a M.S in Computer Science from the University of Provence in 2004.

**Sébastien Thon** is an assistant professor in Computer Science at the University of Provence, France, since 2002. He received his B.S., M.S. from the University of Limoges, France, as well as his Ph.D. in Computer Science in 2001. He joined the LSIS Laboratory, Marseille, France, in 2002. His main research interests include natural phenomena modeling, animation and virtual sculpture.