Multiresolution representation and deformation of wavelet-based 3D objects

Heurtebise Xavier Thon Sébastien Gesquière Gilles

LSIS: Laboratoire des Sciences de l'Information et des Systèmes
IUT de Provence, Rue Raoul Follerau, Route de Crau
13200 Arles – France

{xavier.heurtebise,sebastien.thon,gilles.gesquiere}@up.univ-mrs.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 important number of voxels is required to represent large and detailed objects. As a consequence, the memory cost will be very important and the user/object interaction will be slowed down. In order to allow real-time performance, we propose in this paper a multiresolution model that represents the object with more or less detailed levels thanks to a 3D wavelet transform. We use the marching cubes algorithm to display a triangular surface of the 3D object in various resolutions. To update quickly this surface during sculpture process, we propose a storage method for all the triangles that allows to rebuild only the modified parts of the 3D object. Moreover, to speed up processing and user/object interaction, we also propose a cache system to store in memory the most frequently used levels of details of the 3D object.

Keywords

Computer graphics, virtual sculpture, voxels, levels of details, 3D wavelets.

1. INTRODUCTION

In this paper, we tackle the problem of virtual sculpture of 3D objects with tools, 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, where a 2D image is an 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 spatial enumeration, by regularly sampling the 3D object into voxels with the same size. However, a major drawback of this representation is the large number of voxels needed to represent large objects with detailed features. 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 become slower. Finally, operations on these objects such as sculpture actions or displacements become less and less interactive.

Short Communications proceedings ISBN 80-86943-05-4
WSCG'2006, January 30-February 3, 2006
Plzen, Czech Republic.
Copyright UNION Agency – Science Press

To prevent these inconveniences, adaptive sampling methods have been developed. Libes [Lib91] uses an octree to gather groups of adjacent voxels having same values to reduce the number of elements stored in memory. It's very simple to use and to implement this method. Ferley [Fer02] also works on a n-tree where each cell can be divided in 27 ones. This method looks like an octree and allows to reach a high level of details. However, for an object with small details, the subdivision level of an octree or n-tree will be very high. So the processes (construction and use) will be slow.

Several other sampling methods used in collision detection propose to modify properties of the voxels, such as the size, the orientation or even the shape.

With the AABB method (Axis Aligned Bounding Boxes), Bergen [Ber97] suggests to use voxels with different sizes. Gottschalk [Got96] proposes to modify not only the size of the voxels but also their orientation, with the OBB method (Oriented Bounding Boxes). Thanks to these two methods, the object rendering is optimized because the original object shape can be approached with less voxels than with a simple uniform spatial enumeration. The modeling is finer with OBB tree than AABB tree for a same number of bounding volumes. However, AABB tree uses less memory storage than OBB tree for a same number of bounding volumes. Indeed, an OBB is represented by using 15 scalars (9 scalars for the orientation, 6 scalars for position and extent), whereas an AABB only requires 6 scalars (for position and extent). Moreover, to optimize the modeling of the 3D object, these two methods suggest to reduce overlaps between bounding boxes and to increase their filling by the object, with the less possible boxes. This optimization is expensive in processing time, so we prefer using a uniform spatial enumeration or an octree, because they are faster than AABB and OBB methods.

Liu [Liu88] and Hubbard [Hub95] propose to replace cubes by spheres in an octree to form a spheres tree, because spheres accelerate collision detection between objects. Later, Hubbard [Hub95] [Hub96] and Bradshaw [Bra04] suggest a finer modeling using spheres tree thanks to an approximated medial-axis of the 3D object, but this method is slower and more complicated than an octree.

To further improve the use of spatial enumeration, several methods of multiresolution representations have been proposed. So, processing and display times are adapted with the desired level of details. Among these methods, there are octree and 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 maximum level of details of a multiresolution representation. Boada [Boa01] defines a section in an octree that determines the displayed nodes for a defined level of details. This method is extended to a n-tree by Ferley [Fer02].

The second multiresolution method uses wavelet decomposition. Wavelets are a mathematical tool for representing functions hierarchically. In our case, these functions are discrete 3D functions that define a set of voxels. More information about wavelets will be given in the section 2.1. Muraki [Mur92] [Mur93] shows the use of 3D Haar wavelets to represent a 3D object. Pinnamaneni [Pin02] builds a 3D Haar wavelet decomposition from a sequence of 1D Haar wavelet decomposition in each direction of the 3D voxels grid. Wavelet decomposition allows to display a 3D object faster according to the level of details. 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.

The previously cited methods are about discrete representation of 3D objects. Different methods have already been proposed to sculpt these kinds of objects.

In the Kizamu project, Frisken [Fri01] uses ADFs (Adaptively sampled Distance Fields) to model and to sculpt the matter. 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 between the inside and outside of the shape. Ferley [Fer02] also uses distance fields, stored in a "n-tree" hierarchical representation where the sampling rate depends on object's details. Bærentzen [Bae02] proposes Level-Set method to deform the matter. This method stores distance fields around the exterior of a 3D object.

Raffin [Raf04] proposes a model of virtual sculpture based on a multiresolution representation: the octree. The artist can create his own tools. Each tool modify the data in uniform spatial enumeration, so it modifies the nodes of the octree. This method is more useful and easier than previous method, because it operates directly on spatial enumeration.

Finally, Ayasse [Aya01] proposes to perform sculpture operations by the use of CSG (Constructive Solid Geometry). Complex objects are created by successive modifications of the matter with a tool according to simple operations such as difference, union or intersection. The object and the tool are represented by uniform spatial enumerations. Ayasse proposes to reduce the computation time for each sculpture operation by using only the effective voxels of a movement. This method can be useful because the computation times are reduced. However, it doesn't use a multiresolution representation that could improve the display performance.

As we can see, many models have been proposed in order to represent 3D objects as discrete sets of voxels. However, there remain open issues. The three main problems are the computation time during sculpture operations and the display that must remain real-time, as well as the important memory cost. In this paper, we propose a model based on 3D wavelets that tackles these issues. Although wavelets have already been used to represent discrete 3D objects, it has never been used in a virtual sculpture context where the object must be dynamically updated. Thus, problems linked to real-time modifications of 3D wavelets have never been studied.

The use of 3D wavelets allows to solve the three main problems mentioned above.

- First, we take advantage of the multiresolution nature of wavelets to manage the interaction time between the user and the object to obtain real-time interaction.
- Second, the display of the discrete object is done thanks to the marching cubes algorithm [Lor87] that smoothes the rugged aspect of voxels, thus improving the representation of 3D objects. To avoid rebuilding the whole triangular mesh after each sculpture operation, we propose a storage method for the triangles of the mesh, which permits to modify locally and rapidly the mesh of a 3D object. Moreover, the level of details for the display is automatically selected to satisfy the conditions about user/object interaction.
- Third, our method tackles memory issues. As the memory space is limited, we propose a cache system that offers a compromise between the processing, preprocessing and display times and the memory cost. Last but not least, the wavelets representation of the object allows important compression ratio to reduce the memory cost, especially if lossy compression schemes are used.

The remainder of the paper is organized as follows: in section 2, we present our method in five parts: the multiresolution model based on 3D Haar wavelets, a display method, an automatic management of levels of details to accelerate the display, a cache system and finally some virtual sculpture operations. We conclude in section 3. Finally we expose some future works in section 4.

2. OUR MULTIRESOLUTION MODEL OF SPATIAL ENUMERATION

2.1 The Model

As the uniform spatial enumeration of a 3D object is expensive in processing and display times, we propose a multiresolution model based on 3D Haar wavelets.

On the following example, we explain Haar wavelet decomposition on a 1D case. First, consider a sequence of p values, where p is a power of two (here, $p = 4 = 2^2$):

$$X^0 = [9,7,3,5]$$

Then, by applying Haar wavelet transform, we can represent this sequence in terms of a low-resolution sequence X^1 and a set of detail coefficients Y^1:

$$X^1 = \left[\frac{9+7}{2}, \frac{3+5}{2}\right] = [8,4]$$

$$Y^1 = \left[\frac{9-7}{2}, \frac{3-5}{2}\right] = [1,-1]$$

So, by repeating these operations, we obtain several sets of coefficients corresponding to different levels of details, as shown on the following decomposition table:

level of details #	low-resolution coefficients	detail coefficients
0	[9 7 3 5]	
1	[8 4]	[1 -1]
2	[6]	[2]

So, the higher the number of the level of details, the less detailed the sequence. The sequence obtained by Haar wavelet decomposition has the same size than the original sequence. Its coefficients are the low-resolution coefficients of the last level of details and the different detail coefficients:

Original sequence: [9 7 3 5]
Final sequence: [6 2 1 -1]

The extraction of the original sequence from the final sequence uses the inverse wavelet transform:

$$X^1 = [6+2, 6-2] = [8,4]$$

$$X^0 = [8+1, 8-1, 4+(-1), 4-(-1)] = [9,7,3,5]$$

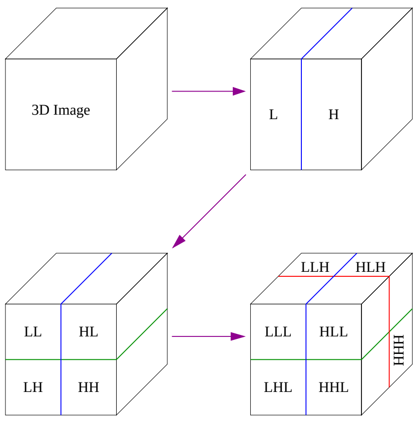

Figure 1. 3D Haar wavelet decomposition.

Similarly, we can use this wavelet transformation in a 3D case. First, the 3D discrete object is defined by a uniform spatial enumeration. Then, by using the wavelet transform we build a hierarchical structure that stores the coefficients of each level of details of this 3D object.

We use the hierarchical structure proposed by Pinnamaneni [Pin02]. For each level of details, the 1D Haar Wavelet transformation is applied in x-, y- and z-direction successively (figure 1).

For each transformation step, we obtain a bloc 'L' with low-resolution coefficients obtained by a low-pass filter, and a bloc 'H' with detail coefficients obtained by a high-pass filter.

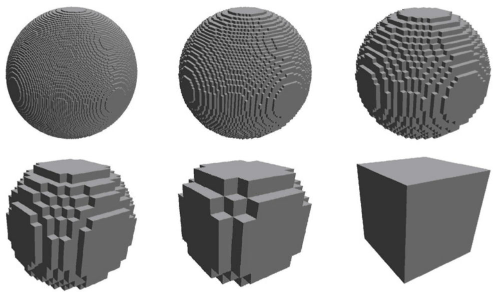
Figure 2. 3D wavelet enumeration for a sphere in 128x128x128 with 6 levels of details (from 0 to 5, from left to right, and from top to bottom).

The figure 2 shows a 3D Haar wavelet decomposition for a sphere with 6 levels of details. Each voxel contains a density value coded in a byte (from 0 for an empty voxel to 255 for a full one).

Several objects permit us to measure the influence of the size of a 3D image on the building time (for wavelet decomposition of the 3D image) and the extracting time (for extraction of a level of details from wavelet enumeration). The results given in this paper have been obtained on a PC with an AMD 3GHz, 1GB of RAM and a NVIDIA Geforce FX 5200 with 128MB video memory.

	Objects	Building (4 levels)	Extracting	
			Level 0	Level 1
64x64x64	cube	0.04599 s	0.06178 s	0.00181 s
	sphere	0.04647 s	0.06228 s	0.00184 s
	torus	0.04611 s	0.06268 s	0.00183 s

	Objects	Building (5 levels)	Extracting	
			Level 0	Level 1
128x128x128	cube	0.69834 s	0.75055 s	0.06286 s
	sphere	0.69753 s	0.75292 s	0.06272 s
	torus	0.70104 s	0.75064 s	0.06252 s

Table 1. Building and extracting times for 3D Haar wavelet enumerations.

As reported on table 1, the building and extracting times do not depend on the kind of 3D object. The building time only depends on the number N of voxels of the initial uniform spatial enumeration and on the maximum level n of details of wavelet enumeration (a is a constant depending on computer power):

$$t_{building} = a \cdot N \cdot \left[1 - 0.125^n \right]$$

The extracting time depends on the number N of voxels of the initial uniform spatial enumeration, the maximum level n of details of wavelet enumeration and the extracted level p of details (b is a constant depending on computer power):

$$t_{extracting} = b \cdot N \cdot \left[0.125^p - 0.125^n \right]$$

These properties will be useful to estimate the building and extracting times in section 2.4.

2.2 The Display

To display a 3D discrete object, there are several methods such as a display with cubes, spheres or surfels [Sze92]. Nevertheless, the rendering of these methods has a very rough aspect.

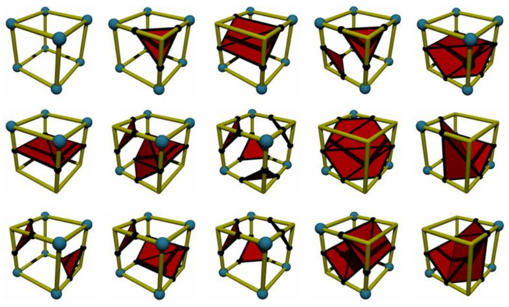
Figure 3. The 15 configurations proposed by Lorensen and Cline in 1987 [Lor87].

To obtain a smooth surface instead of a set of boxes, we use the marching cubes algorithm [Lor87] to build a triangulated surface, by using the different configurations shown on figure 3 (The voxels are the eight vertices of the cubes). For each configuration of full and empty voxels, from 0 to 5 triangles are generated.

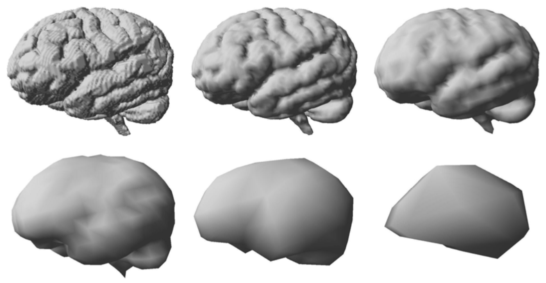
Figure 4. Marching cubes method for an object in 128x128x128 with 6 levels of details (from 0 to 5, from left to right, and from top to bottom).

In the case of a binary coding of the object, the marching cubes algorithm would lead to a rather angular rendering: the surface would always run exactly in the middle of two inside and outside voxels, thus generating angles multiple of 45°. However, as we have values between 0 and 255, we obtain a smoother set of triangles by an interpolation of voxels values.

The marching cubes algorithm is applied on each level of details of 3D wavelet enumeration, so a triangulated surface is obtained for each level of details (figure 4).

	Preprocessing times		
	Level 0	Level 1	Level 2
sphere	0.43098 s	0.06498 s	0.01113 s
torus	0.38808 s	0.05524 s	0.01092 s
brain	0.49629 s	0.08615 s	0.01310 s

	Display times		
	Level 0	Level 1	Level 2
sphere	0.03294 s	0.00824 s	0.00221 s
torus	0.01843 s	0.00473 s	0.00113 s
brain	0.05552 s	0.01268 s	0.00276 s

	Number of triangles		
	Level 0	Level 1	Level 2
sphere	147 968	36 672	9 160
torus	72 368	17 800	4 584
brain	296 416	68 124	13 968

Table 2. Preprocessing and display times and number of triangles of the triangulated surface for 3D Haar wavelet enumerations of 3D objects in 128x128x128.

However, the marching cubes algorithm is a rather slow process. Using this algorithm to compute the triangulated surface of a whole 3D object each time it needs to be displayed would be too time consuming to achieve real-time display (table 2). The more detailed the 3D object, the higher the computation time, as a lot of triangles are computed. In order to improve computation time during sculpting process and to have real-time interactions between a user and the 3D object, we propose the following strategy: we compute the marching cubes on the whole object only once, as a preprocessing step before sculpting. Then, during the sculpting process, the marching cubes algorithm is only applied to the voxels modified by the sculpting tools. Thus, we only compute the new triangles of the surface locally altered by the tool.

In order to modify locally and rapidly the triangulated surface during sculpting process, we propose a method to store the triangles of the surface, which permits to easily find and modify the triangles for the modified parts of the 3D object.

This method uses a data structure (figure 5), composed by:

- A linked list for the storage of triangles of the surface for each block of 8 voxels.
- A 3D matrix of pointers to improve the access to the linked list. When the triangulated surface is created, the 3D image is processed by block of 8 voxels. For each block, a piece of surface is created with triangles (from 0 to 5) thanks to the marching cubes algorithm. For 0 triangle, the pointer of the 3D matrix is *NULL*. Else, it point to a item in the linked list containing the triangles data.

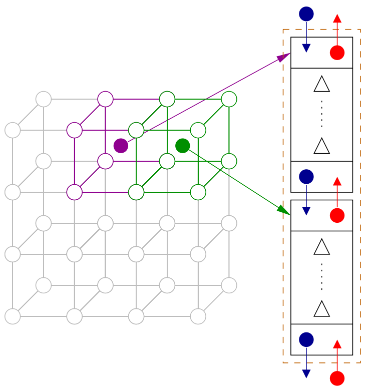

Figure 5. Display structure with a 3D matrix of pointers (on the left) to a linked list (on the right) for the storage of triangles of the surface.

2.3 Automatic Management of the Levels of Details

In order to accelerate the display of a 3D object defined with our model, we take advantage of its multiresolution nature given by the wavelets. We propose an automatic management of the levels of details that selects the more adequate level of the 3D object to display. For that, we use the three following criteria:

- If the user interacts with the 3D object, we define a frame rate N_1, according to the power of the computer, to display the 3D object in real-time. Consequently, the greater the frame rate N_1, the faster the display.
- If the user doesn't interact with the 3D object, we define a frame rate N_2, according to the power of the computer, to display the 3D object. In fact, it's unnecessary to display object in real-time, but the user can interact rapidly with the object. So, the frame rate N_2 is smaller than N_1.
- Finally, we rectify the level of details obtained by respecting the previous conditions, according to the proximity between the user and the object. So, the greater the distance between the user and the object, the more rough the level of details, because the details become invisible for the user. For a very great distance (right of figure 6), the coarser level of details is displayed. On the contrary, when the distance is small (left of figure 6), the most detailed level of details is displayed.

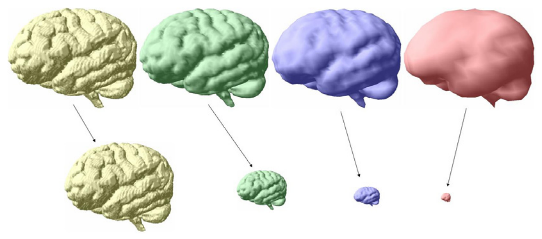

Figure 6. Effect of the proximity between the viewpoint and a 3D object in 128x128x128 with 4 levels of details (from 0 to 3, from left to right).

2.4 The Cache System

Thanks to the automatic management of the levels of details, the display time is improved. But, as the memory space is limited, we can't store in memory all the levels of details of the 3D wavelet enumeration. However, in order to preserve access performance to a given level of details of the 3D object (to modify it, to display it, etc.), it is crucial to avoid to extract this level each time we need to use it. Consequently, we propose a cache system method that offers a compromise between memory usage and the performances of processing, preprocessing and displaying.

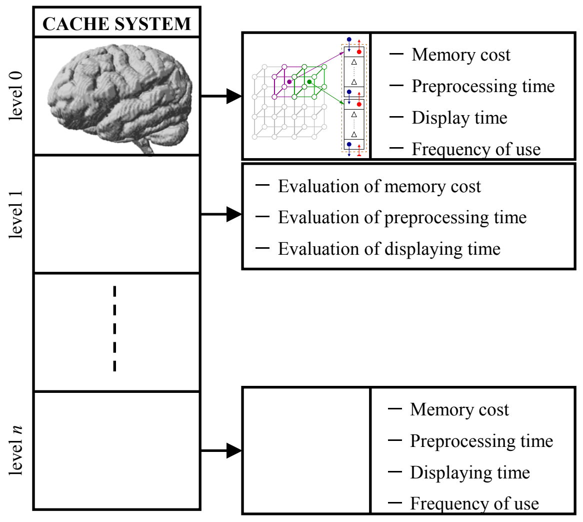

Figure 7. Data structure of the cache system.

The principle of our cache system is to store the most frequently used levels of details of the 3D object. So, we propose the storage of levels of details in memory space, according to several criteria:

- maximum memory size allowed;
- frequency of use of each level;
- processing, preprocessing and display times.

The cache system (figure 7) uses a data structure to store the levels of details of the 3D object. The size of this structure depends on the maximum level of details of the 3D wavelet enumeration. Each level of this structure corresponds to one level of details of the multiresolution representation. For each level of details, this structure is composed of:

- if the level of details exists:
  - a 3D image;
  - a data structure for displaying the object ;
  - the memory cost;
  - the preprocessing and display times;
  - the frequency of use of the level of details;
- else:
  - the evaluation of the memory cost;
  - the evaluation of the preprocessing and display times;

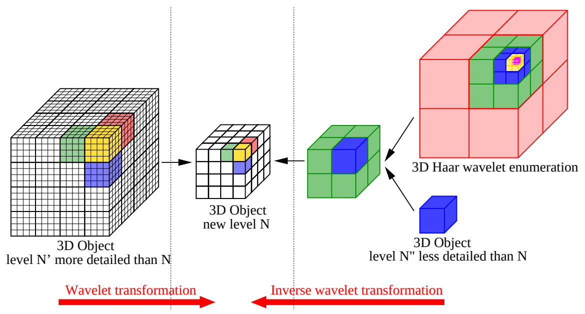
3D Object
level N' more detailed than N
3D Object
new level N
3D Haar wavelet enumeration
3D Object
level N" less detailed than N
Wavelet transformation
Inverse wavelet transformation

Figure 8. Building the new level from a more detailed level (on the left) or from a less detailed level (on the right).

When a new level has to be added to the cache structure, it is crucial not to extract it from the wavelet enumeration, but to extract it from a level already present in the cache, as less reconstruction steps are needed. For that, the new level is built from:

- the first existing more detailed level by using the wavelet transform only for the low-resolution coefficients;
- the first existing less detailed level by using the inverse wavelet transform.

We automatically choose between these two methods the fastest one to rebuild the new level according to the different existing levels in the cache system (Figure 8).

2.5 The Virtual Sculpture

In this paper, we propose a simple case of sculpture of the matter: adding and subtracting voxels to the 3D wavelet enumeration. 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. Other tools and the interaction with other representation (like implicit functions, etc.) will be studied in future works.

In this paper, the tool can only be positioned in translation relatively to the matter, without rotation. Rotation issues will be studied in future works. In the case of a tool in translation, a collision test is made between the bounding boxes of the tool and the matter to speed up the sculpture time. If there is a collision, the following operations are performed:

- find the voxels of the matter and the tool which are in the collision zone;
- find which voxel of the tool intersects which voxel of the matter in this zone.

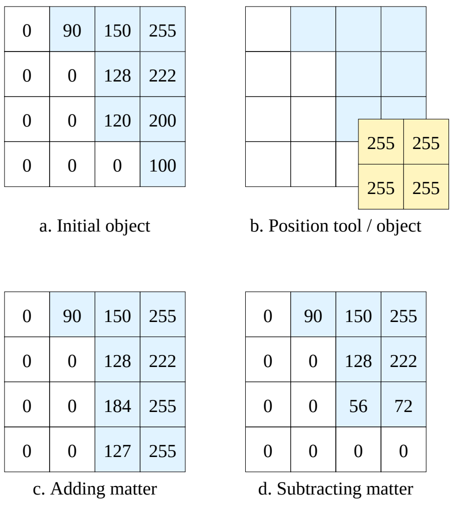

Figure 9. Adding or subtracting matter to an object with a tool.

If there is intersection between voxels of the matter and the tool in the collision zone, the filling percentage of the voxel of the matter by the one of the tool is computed. It's the ratio between the volume of the intersection between the two voxels and the volume of the one of the matter. The two following modes are illustrated in 2D on figure 9:

- in the "Adding matter" mode, if the voxel of the tool isn't null, the filling percentage is added to the value of the voxel of the matter. If this value becomes greater than 255, it is put to 255;
- in the "Subtracting matter" mode, if the voxel of the tool isn't null, the filling percentage is subtracted to the value of the voxel of the matter. If this value becomes negative, it is put to 0.

When the computation is made on all the voxels of the matter in the collision zone, only the images for the levels of details existing in the cache system and the 3D wavelet enumeration are updated.

Moreover, for each level of details existing in the cache system, the triangulated surface is rebuilt only for the modified parts of the 3D object, to improve the computation time. First, the coordinates of the modified voxel are used to access, thanks to the 3D matrix presented on figure 5, to the items of the linked list that are deleted. Then, the new triangles are computed and the linked list as well as the 3D matrix are updated.

Several examples of sculpture, in 128x128x128, are illustrated on figures 10, 11 and 12 :

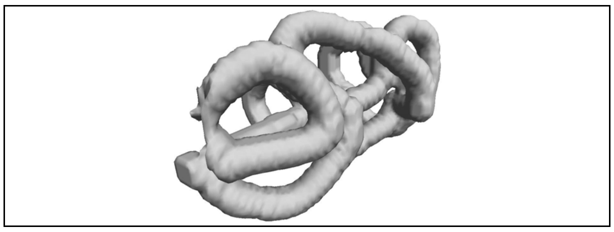

Figure 10. A random sculpture built with a spherical tool in less than 1 minute.

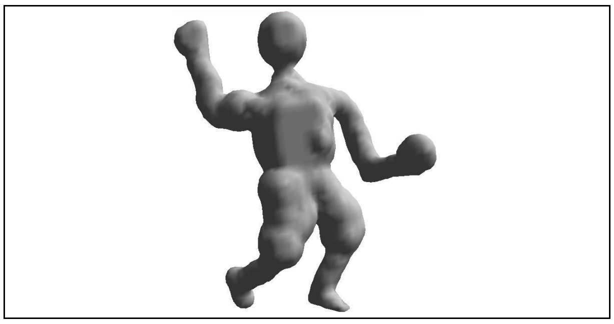

Figure 11. A sportsman sculpted with 3 spherical tools at different sizes in less than 5 minutes.

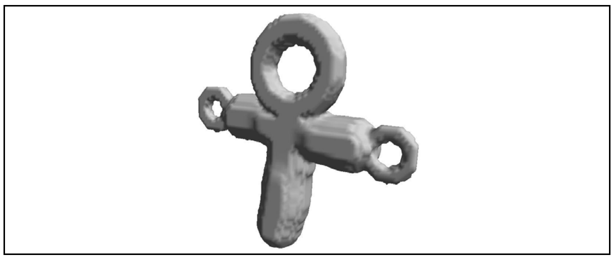

Figure 12. A relic sculpted with a spherical tool and two ring tools in less than 1 minute.

3. CONCLUSION

We have presented in this paper a model of 3D object that allows real-time virtual sculpture on an average PC. This object is represented as a set of voxels, but we avoid speed and memory issues inherent to this kind of representation thanks to a multiresolution approach based on 3D Haar wavelets. In order to enhance real-time performance, we also have developed several structures, as follows:

- A management of the interaction time between the user and the object to obtain real-time interaction.
- A storage method for the triangles of the mesh, generated by the marching cubes algorithm, to

modify locally and rapidly the mesh of a 3D object after each sculpture operation.

- An automatic management of the level of details for the display to satisfy the conditions about user/object interaction.
- Finally, a cache system to offer a compromise between the memory cost and the processing, preprocessing and display times.

To verify the applicability of our sculpting system, using simple sculpture tools, we have conducted many sculpting sessions which have resulted in numerous interesting sculptures. Some sculptures examples are shown on figures 10, 11 and 12, and several other examples can be seen on http://www.iut-arles.up.univ-mrs.fr/thon/.

4. FUTURE WORKS

Many improvements of our sculpture system are possible, as interaction, speed and memory cost will always be challenging issues.

Concerning interaction, we plan to manage the rotation of tools relatively to the matter. Moreover, more sculpture operations will be added. We will also improve the realism of sculpture actions, by adding parameters to the voxels to imitate physical behavior.

In order to accelerate the display when the user interacts with the matter, different display methods will be investigated.

Finally, we plan to reduce the memory cost of the sculpted object by taking advantage of the important compression ratio allowed by the wavelet representation.

5. REFERENCES

[Aya01] Ayasse, J., and Müller, H. Interactive Manipulation of Voxel Volumes with Free-formed Voxel Tools. In Proceedings of the Vision Modeling and Visualization Conference 2001, pp.359-366, 2001.

[Bae02] Bærentzen, J.A., and Christensen, N.J. Volume sculpting using the Level-Set method. Shape Modelling International 2002, IEEE Computer Society, pp.175-182, 2002.

[Ber97] Bergen, G.V.D. Efficient collision detection of complex deformable models using AABB trees. Journal of Graphic Tools, 2(4), pp.1-13, 1997.

[Boa01] Boada, I., Navazo, I., and Scopigno, R. Multiresolution volume visualization with a texture-based octree. Visual Computer, 17, pp.185-197, 2001.

[Bra04] Bradshaw, G., and O'Sullivan, C. Adaptive medial-axis approximation for sphere-tree construction. ACM Transactions on Graphics, 23(1), pp.1-26, 2004.

[Fer02] Ferley, E. Sculpture virtuelle. Ph.D. thesis, Institut National Polytechnique de Grenoble, 2002.

[Fri01] Frisken, S.F., and Perry, R.N. Kizamu: a system for sculpting digital characters. Proceedings of ACM SIGGRAPH 2001, pp.47-56, 2001.

[Got96] Gottschalk, S., Lin, M.C., and Manocha D. OBB-Tree: A hierarchical structure for rapid interference detection. In Proceedings of ACM SIGGRAPH'96, pp.171-180, 1996.

[Hub95] Hubbard, P. Collision detection for interactive graphics applications. Ph.D. thesis, Dept. of Computer Science, Brown University, 1995.

[Hub96] Hubbard, P. Approximating polyhedra with spheres for time-critical collision detection. ACM Transactions on Graphics, 15(3), pp.179-210, 1996.

[Lib91] Libes, D. Modeling dynamic surfaces with octrees. Computer & Graphics, 15(3), 1991.

[Liu88] Liu, Y., Noborio, J., and Arimoto, S. Hierarchical sphere model (HSM) and its application for checking an interference between moving robots. In Proceedings of the IEEE International Workshop on Intelligent Robots and Systems, pp.801-806, 1988.

[Lor87] Lorensen, W.E., and Cline, H.E. Marching Cubes: a high-resolution 3D surface construction algorithm. Computer Graphics, 21(4), pp.163-169, 1987.

[Miz98] Mizuno, S., Okada, M. and Toriwaki, J. Virtual sculpting and virtual woodcut printing. The Visual Computer, 14(2), pp.39-51, 1998.

[Mur92] Muraki, S. Approximation and rendering of volume data using wavelet transforms. In Proceedings of Visualization '92, Boston, pp.21-28, 1992.

[Mur93] Muraki, S. Volume data and wavelet transforms. IEEE Computer Graphics and Applications, 13(4), pp.50-56, 1993.

[Pin02] Pinnamaneni, P., Meyer, J., and Saladi, S. Remote transformation and local 3-D reconstruction and visualization of biomedical data sets in Java3D. In Proceedings of Electronic Imaging Science & Technology Visualization and Data Analysis Conference, San Jose, CA, pp.44-54, 2002.

[Raf04] Raffin, R., Gesquière, G., Remy, E., and Thon, S. VirSculpt: a virtual sculpting environment. GraphiCon '04 Proceedings, pp.184-187, 2004.

[Sze92] Szeliski, R., and Tonnesen, D. Surface modeling with oriented particle systems. Computer Graphics, 26(2), pp.185-194, 1992.