Interactive Volumetric Textures based on reconstruction method

Babahenini Mohamed Chaouki and Djedi NourEddine

LESIA Laboratory. Department of Computer science. Mohamed Khider University Biskra (Algeria)
Chaouki.Babahenini@gmail.com, djedi_nour@yahoo.fr

***Abstract* — In this paper we present a method for interactively rendering complex repetitive scenes. It consists to integrate reconstruction method (voxel coloring) into volumetric textures using image layers. Our approach start from a set of images of the reference element, which will be converted in a second phase into a whole of layers (2D images considered as transparent textures), those are projected and composed successively on surface defined as volumetric grid using the Z-buffer algorithm.**

The model suggested allows realistic rendering of repetitive complex scenes with lower cost of calculation due to the effective exploitation of the capacities of graphics hardware and to the fact that it takes account of the level of detail according to the distance of the observer and the vision angle, in the representation of the reference element.

***Index Terms* — Real time, Image layers, Level of details, Textures volumetric, Voxel coloring.**

I. INTRODUCTION

Volumetric textures offer realistic rendering of the repetitive complex scenes, in acceptable times (from 5 to 20 minutes) by using the ray casting. However, for applications requiring the real time, obtaining an additional profit appears not easily possible by preserving this technique in its initial form. Among the approaches of image based rendering, dealing with the problems of geometrical and temporal complexity, exist the rendering based on images layers. This technique allows realistic rendering of scenes and objects in lower calculation cost than with a traditional method. This benefit of calculation time is due to the exploitation of capacities of the graphics hardware treating, effectively, the polygons. So each layer of image must be represented by a textured polygon.

For rendering, it is enough to project and successively compose the layers on the image plan (Z-Buffer) and to obtain, thus, the final image. Each projected layer is combined with the preceding result by taking account of the density of each pixel.

The projection of a layer is faster than calculations of projection for each voxel along a ray, which allows an appreciable benefit of time.

This paper present a method for integrating the technique of coloring voxel in volumetric textures containing layers of images [1], which will make it possible to generalize the reference volume which will be initially definite from images key (real photographs or synthesized images), transform it into a whole of layers and to adopt the method of Meyer [2] to carry out one made real time by exploiting the graphics hardware capacities.

The motivation of our representation is:

- Encoding any kind of shape initially represented by a series of real photographs or synthesized images using voxel colouring technique in volume pattern.
- Providing multi-scale representation by converting obtained volumes to image layers witch will be projected and decomposed successively on surface support defined as volumetric grid.
- Using capabilities of graphics hardware, witch makes volume rendering interactive.

The remainder of this paper is organized as follow: we begin with a rapid summary of previous approaches for representing volumetric textures based on image layers (Section 2). It is follows by an overview system architecture and a detail description of interactive volumetric textures based on reconstruction method in Section 3. Results are presented in Section 4, which describes the examples shown in the figures. The final section (Section 5) presents a conclusion and some directions of future work.

II. PREVIOUS WORK

A. Initial representation

In 1989 Kajiya and Kay [3] presented the volumetric textures technique applied to fur rendering, which constitutes an efficient way of representing and rendering complex repetitive data. The principle consists in making a sample of volumetric texture; which is stored in a *volume of reference*, whose deformed copies, called "*texels*", will be mapped on a surface made up of bilinear patches.

The reference volume represents area of space sampled onto voxels. Each voxel contains information space of presence (density) and function indicating the local photometric behaviour, which consists of a local orientation and an analytical function of reflectance.

The volume of reference is then plated in a repetitive way on all surface as for a traditional surface texture, it is then deformed in order to ensure continuity between the close texels.

The rendering is done by volumetric ray tracing when a texel is crossed. A stochastic sampling of the ray is

carried out in order to reduce calculations; the local illumination is then carried out using the model of reflectance which simulates the presence of a cylinder. A ray is sent towards the source of light in order to test the shade of the current voxel.
F. Neyret [4] proposed the use of the octree to store reference volume. Any voxel in the octree simulates the photometric local behaviour of the object represented in the texel, and contains the local density and a micro-primitive modelling the local reflectance.
Volumetric texture rendering is then modified according to the multiscale data structure and to the reflectance. So it is encoding in two passes one made total which is done on the level of each texel by using a traditional algorithm of cone tracer. This algorithm course respectively space scene, space texel then space object and uses a trilinear interpolation and one made local which is done on the level of each voxel by integrating the model of local illumination of *Phong*.

B. Representation based on image layers

To minimize the time of rendering, Lacroute and Levoy[5] introduced volumetric rendering by image layers where the volumetric data are interpreted as layers and the voxels are factorized in texture and then treated in parallel. The algorithms rely on a factorization of the viewing transformation that simplifies projection from the volume to the image, which allow constructing an object-order algorithm with the same advantages as an image-order algorithm. They call the factorization the *shear-warp factorization*. The principal advantage is the benefits of time because the projection of a layer is faster than the calculation of projection for each voxel along a ray. Westermann and Ertl in 1998 [6] presented a model including diffuse lighting in the medical images; they used a single texture of NxXxY resolution. For rendering they position and turn correctly the cube in the scene and then determine the polygons forming the intersection of parallel plans to the image plan with the cube in the scene. To simulate the diffuse shading, they introduced another 3D texture whose color information corresponds to the normal in each point of volumetric texture.
To carry out volumetric textures in real time, Meyer and Neyret [2] tried to benefit from the capacity of the graphics hardware to treat textured polygons quickly. In this context, a cubic volume is represented by a series of slices covered with a transparent texture. A 3D object is thus translated into sections. As the slices are not front the observer and do not have a thickness, one is likely to see between them, thus they have defined three directions of slices and used one among it according to the point of view of the user.
Schaufler[7], introduced a new type of impostor, which has the property to limit the errors of occlusion to the user specified amount. This impostor is composed of multiple layers of textured meshes, which replace the distant geometry and is much faster to render. It captures in the model the complexity of suitable depth without resorting to a complete of the scene. The layers can be updated dynamically during visualization.
A very interesting adaptation of volumetric textures containing image layers was developed by Lengyel [8], for real time rendering of fur. As a pre-process, he simulates virtual hair with a particle system, and simplifies it into a volume texture. Next, he parameterizes the texture over a surface of arbitrary topology using “lapped textures” which is an approach for applying a texture sample to a surface by repeatedly pasting patches of the texture until the surface is covered. At runtime, the patches of volume textures are rendered as a series of concentric shells of semi-transparent medium. The method generates convincing imagery of fur at interactive rates for models of moderate complexity.
Models containing billboard [9] allow a representation of the objects containing semi-transparent textures, the rendering engine always directs towards the observer. Jakulin[10] proposes a hybrid model between the texels. He uses ordinary mesh-based rendering for the solid parts of a tree, its trunk and limbs, the sparse parts of a tree, its twigs and leaves, are instead represented with a set of slices, an image based representation. For rendering, it blends between the nearest two slicing according to the angle between the viewpoint and the normal with the slice.
With the advent of rising generation of the graphics hardware, Sénégas [9] could develop a new version of volumetric textures in real time by adding a calculation of illumination to the texels. These graphics hardware make possible to evaluate illumination in each pixel of the polygon and either at the tops of the grids. Then these values will be interpolated during the process of rasterisation which consists in filling the 2D polygon pixel by pixel. A layer of a texel is then made up of two section, one coding color (RGB) and the density (channel alpha) and the other coding the normals.

III. INTERACTIVES VOLUMETRIC TEXTURES BASED ON RECONSTRUCTION METHOD :

A. Modelling of the reference element by the voxel coloring technique

Voxel coloring technique was introduced by *Seitz* [10], it can considered as a voxel-based 3D reconstruction technique to compute photo realistic volume models from multiple color images $V_0,\ldots, V_n$.
It consists in reconstructing a scene 3D by assigning colors (radiances) to voxels (points) in a 3D volume and by guaranteeing consistent with a whole of basic images.

The initial step of the algorithm consists in breaking up the scene into a set of voxels. For that, we calculate a subdivision of space 3D in layers of voxels of uniform distance at the camera.

We define a threshold thresh that corresponds to the maximum allowable correlation error. A low value of the threshold makes it possible to have a precise reconstruction but incomplete, with a larger threshold we obtain a complete reconstruction, having some false voxels.

To be able to reconstruct the 3D scene, we used a set of synthesized images, generated by using a modified ray casting. After having computed a whole of images of depth around the objects of the scene, we can reconstruct this scene by using these images accompanied by other information such as: the position of the camera, focal angles of vision (vertical and horizontal) and the dimension of each image. The process of reconstruction is an opposite projection of the whole of pixels of each image towards the space scene discretized in voxels. This operation makes it possible to color the transparent voxels and to have thus the same scene, represented this time by a series of voxels. This process is carried out in the following way:

- Initially, we construct the grid 3D of the transparent voxels (the value of alpha is 0).
- We use information accompanying the camera to compute the matrix M of change scale between the object space and the camera space. $M = R_X \times R_Y \times T$.
- Once the direction of the ray obtained, we compute the position of the pixel projected in space scene by using the information of depth accompanying this pixel: $\vec{P} = O + \lambda \times D$. Where: O is the ray origin, D is the normalized vector of the ray direction and λ corresponds to the depth of the point. *P*.
- Then we compute the position of the corresponding voxel in the grid and we assign the color of the pixel to the found voxel, and the value alpha of the voxel is 255.

Once the creation of the reference volume by the technique of voxels colouring made, we can use the whole of voxels colored for the generation of our volume of reference containing layers. The process of passage between these two representations is illustrated in Fig. 1.

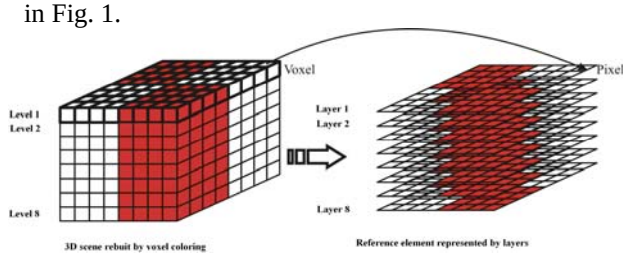


Fig. 1: The passage between a voxel representation and image layers representation.

B. Rendering by using the new model of volumetric texture

The surface can be represented by a whole of boxes built by using a surface grid. Each box is thus defined by *2N* points [2]: *N* points for the base and *N* points for the top (Fig. 2). To indicate how the box in the texel is located, it is necessary to add to the base points the textures coordinates *U*, *V*.

For the creation of a surface grid, we used sinusoidal functions (basic points), which one will be able to be perturbed by textures of *Perlin* (Perlin noise) in order to create grounds.

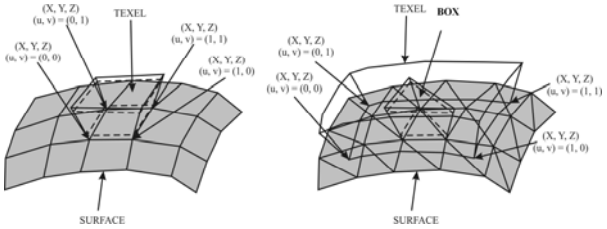


Fig. 2: Mapping texel on a surface [2].

The volume of reference is represented by a whole of *N* 2D images of size *X* ×Y to format RGBA (*Red*, *Green*, *Bleu* and the *Alpha* transparency). We can also see it like a volume of *N*×*X*×*Y* voxels having each one four components (three of colors and transparency). This volume of reference does not contain any more information on reflectance since the precalculated color is stored in the place, which implies that the render of shades and lighting is fixed during its construction.

Rendering of a scene passes by rendering of all the boxes comprising a whole of textured transparent polygons (the images layers). From the information contained in a box, we obtain the coordinates (in space scene *X*, *Y*, *Z* and in space textures *U*, *V*) of textured faces which we display with ***OpenGL*** [2]. A box is defined by its points of the base and the top. The calculation of the coordinates of the various sections is done by a linear interpolation between a point of the base and a point of the top. The face coordinates i are:

$$Ai = A_0 + \frac{i}{n}.\overrightarrow{A_0 A_{n-1}}$$

$$Bi = B_0 + \frac{i}{n}.\overrightarrow{B_0 B_{n-1}}$$

$$Ci = C_0 + \frac{i}{n}.\overrightarrow{C_0 C_{n-1}}$$

Where A_0, B_0, C_0 are coordinates of bottom layer, A_{n-1}, B_{n-1}, C_{n-1} are coordinates of top layer, and n is the number of layers.

The coordinates of the points of these faces are recomputed only when the box becomes deformed. The technique of the Z-buffer imposes that the display of transparent faces is done by order of depth. When ***OpenGL*** displays a new polygon, it treats all the pixels of this polygon like this [2]:

If (the depth of the point is further away from the eye than that of the Z-buffer) and (the point of the Z-buffer belongs to a completely opaque polygon) **Then**

The point is rejected,

else we display it by applying the following formula:

$$C_{screen} = Alpha_{point} \times C_{point} + (1 - Alpha_{point}) \times C_{screen}$$

If the point belonged to an opaque semi polygon, it is necessary to make a composition of the colors [2] by displaying the faces of the back forwards by respecting the order of depth according to the point of view.

The method of rendering all textured surfaces consists in taking each face of surface and displays the box being with the top.

When the angle of sight of the texel is very acute it is seen clearly that surface is a superposition of faces. To solve this problem, we chose to introduce two other directions of alternate sections, to use when the first gives a too degenerated quality. We choose the most adapted direction to the position from the point of view.

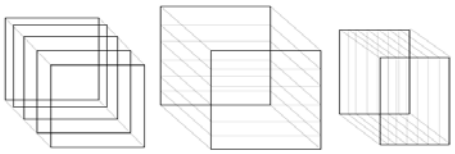


Fig. 3: A texel with its 3 directions of faces.

To know which direction is the best, we calculate the three scalar products between the axis of sight V and the three directions of the texel *NR*, *T*, *B* (Normal, Tangent, Bi-normal), and one chooses the direction whose scalar product is largest in absolute value.

C. Introduction of the multi-resolution aspect

The introduction of the multi-resolution aspect consists in reducing the number of display layers while keeping a constant visual quality, the objective being to find right balance between the realism and the computing speed.

- When the texel is seen perfectly in the axis or when the point of view deviates a little from the direction of vision, the effect of relief is not important. We could thus display only one limited number of textured faces which would be the precalculated result of "the stacking" of all the textured faces [2]. The idea is to display a variable number of layers which depends on the angle of view, and more precisely of the distance *d* (Fig. 4) which corresponds to the visible depth inside the object.

The quality criterion C_1 is defined by raising the distance d by d_0, which gives: $\frac{H \times \tan(a)}{n} < d_0$, it is deduced that $\frac{H \times \tan(a)}{d_0} < n$. Where n is the minimum number of sections which should be used to keep a quality connect constant.

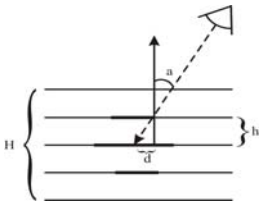


Fig. 4: The first criterion.

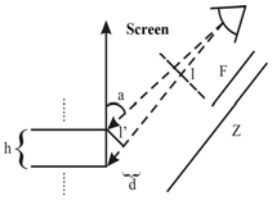


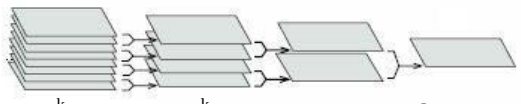
Fig. 5: The second criterion.

- The C_2 criterion consists in defining the spacing between two layers of a texel (i.e. we wants that the distance H projected in the screen is limited by d_1 (in pixels)). The distance h projected in the screen is l (see fig. 5). We want that $l<d_1$, what gives: $\frac{F \times H \times \sin(a)}{Z \times d_1} < n$

This criterion takes account of the perspective; therefore a texel being far from the point of view will have a minimum number of layers very low. This criterion also takes account of the orientation of the texel compared to the axis of slice. The minimal number of layers to be display when the two criteria are used is the maximum of the two found values (i.e.: max (C_1, C_2)).

To display the texel result with the level of detail calculated according to the preceding criterion, we precalculates intermediate images where each one is the combination of the two preceding ones:

From the $N = 2^k$ images (N being power of 2 immediately higher to n), we build $\frac{N}{2} = 2^{k-1}$ by combining them two by two according to the principle of the compositing [11], and so on until obtaining only one last image, which will be displayed according to the position and the orientation from the point of view.



$N = 2^k$ images 2^k images last image

Fig. 6: layer images compositing

IV. RESULTS AND DISCUSSION

All the computations are done on a P4 at 3.0 GHz with a GeForce FX 5200 graphic card with 128 Mb of memory. Fig. 9 represents two reference volumes generated by voxels colouring technique which uses the information of depth accompanying each pixel by an image source, once volumes obtained they are translated into a whole of layers. The image (fig. 7a) is regenerated by using six images of depth which resolution is 256x256 pixels. In this case, there are 8.597 colored voxels between 262.144 voxels (64^3) for a computing time 3,52 seconds. The images (fig. 7b) are obtained in the same way with 12 coloured key images and 128^3 voxels, and in figure 8 we have two other volumes of reference constructed by using a voxels coloring starting from a triangular grid (or cloud of points).

Fig. 7: Reference volumes obtained by using the voxels coloring technique.

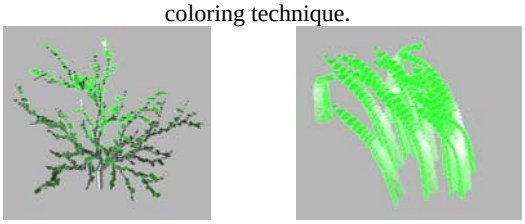


Fig. 8: Reference volumes (grass and tree) obtained from triangular meshes

Fig. 9 shows a whole of views; it represents complex terrain made up of 10.000 patches built from a *Perlin* texture equipped by trees. The frame rate is 20 to 30 fps depending on the view and the tuning of parameters. The frame rate is inversely proportional to resolution, and transitions between LODs of the same type of slices are quite unnoticeable. Rendering is carried out in an interactive time when the displaying frequency varies between 9 and 21 F.P.S and the reference volume resolution is 128^3.

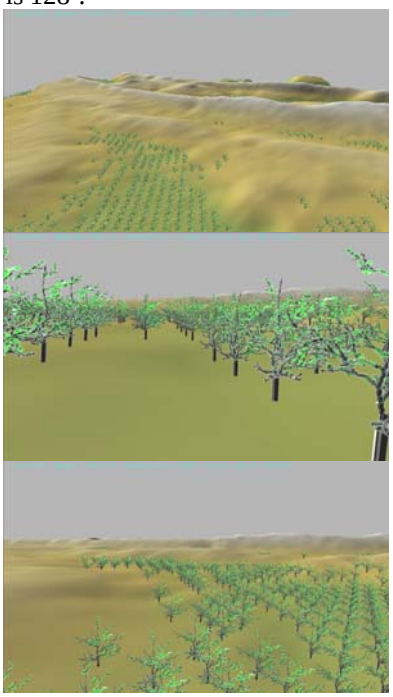


Fig. 9 : Different view position of a terrain of 10.000 patches equipped with trees

V. CONCLUSION AND FUTURE WORK

We have described a multiscale scheme for real-time volumetric textures by integrating new way to represent reference element: the voxel coloring technique, which permits to generalize the use of the texel to objects defined only by key images (real photographs or images obtained by a synthesis process). To map texel over surface support, we use technique based on image layers [2] that was particularly effective and is well adapted to visualization of volumetric textures using graphics hardware. We have shown that our results are convincing: we can move interactively above a scene which shows a continuous range reference element. However, it still remains of certain points to improve:

- The use of three directions of layers extends three times the memory capacity used to store the reference volume, what results in an increase in the space occupied in the memory textures.
- Generalization of the scene to be equipped with other types of surface and introduction of 3D textures. [12]
- Generating local illumination on texel by integrating image based rendering techniques using graphics capabilities.

REFERENCES

[1] Decaudin, P. and Neyret, F. “Rendering Forest Scenes in Real-Time”. *In Eurographics Symposium on Rendering*. 2004

[2] Meyer, A. and Neyret, F. “Interactive volumetric textures”. *In Eurographics Rendering Workshop*, pp: 157-168. 1998

[3] Kajiya, J. and Kay, T. “Rendering fur with three dimensional textures”. *Proceedings of SIGGRAPH’ 89, vol. 23, N° 3*, pp: 271- 280. 1989

[4] Neyret, F. “Modeling animating and rendering complex scenes using volumetric textures”. *IEEE Transactions on Visualization and Computer Graphics*, vol. 4, N° 1. 1998

[5] Lacroute, P. and Levoy, M. “Fast volume rendering using a shear–warp factorization of the viewing transformation” *Proceedings of SIGGRAPH ’94*, pp: 451–458. 1994

[6] Westermann, R. and Ertl, T. “Efficiently using graphics hardware in volume rendering applications”. *Proceedings of ACM SIGGRAPH*, pp: 169 - 178. 1998

[7] Lengyel, J., Praun, E., Finkelstein, A. and Hugues, H. “Real-Time Fur over Arbitrary Surfaces.” *Symposium on Interactive 3D Graphics*, pp: 227-232. 2001

[8] Jakulin, A. “Interactive Vegetation Rendering with Slicing and Blending” *Proc. eurographics (Short Presentations).* 2000

[9] Sénégas, F. Rendu de forêts en temps-réel à base de représentations alternatives. *Rapport de DEA IVR, INPG.* 2001

[10] Seitz, S. M. and Dyer, C. R. “Photorealistic Scene Reconstruction by Voxel Coloring” *Proc. Computer Vision and Pattern Recognition Conf.* pp: 1067-1073. 1997

[11] Porter, T. and Duff, T. “Compositing Digital Images” *Computer Graphics* Volume 18, N° 3, pp: 253 – 259. 1984

[12] Lefebvre, S., Hornus, S. and Neyret, F. “Texture Sprites: Texture Elements Splatted on Surfaces I3D” - *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games.* pp: 163-170. 2005