Hybrid visualization for maxillofacial surgery planning and simulation

Carlos Buchart
CEIT and Tecnun (University of Navarra)
cbuchart@ceit.es

Gaizka San Vicente
CEIT
gsanvicente@ceit.es

Aiert Amundarain
CEIT
aamundarain@ceit.es

Diego Borro
CEIT and Tecnun (University of Navarra)
dborro@ceit.es

Abstract

Currently, many visualization methods are used in computer assisted medicine. It is commonly considered that a unique visualization scheme makes difficult the interaction and limits the quality and quantity of the information shown. In this paper we study the specific requirements of a maxillofacial surgery simulation tool for facial appearance prediction. The different stages of the application lead to present medical information in different ways. We propose to adapt visualization techniques to give a more suitable answer to these needs: a hybrid volumetric and polygonal visualization for the planning stage, as well as a scheme for surgery definition in 2D. Finally, we propose the use of mesh visualization for the simulated model, which previously requires the 3D reconstruction of the surface in order to be visualized.

1. Introduction

In the last decades, computer science has given a new dimension to medical information, enabling the prediction and simulation of surgical treatments. Indeed, computer-aided surgical simulation (CASS) is widely used in medicine since it allows saving much time in the surgery planning process. One of the many developed applications that uses this techniques is the maxillofacial surgery planning since it appears to be at least as good as traditional methods [20], [19].

Maxillofacial surgery treats abnormalities, injuries, and diseases in the head by bone fragment repositioning, restoration of bone defects, and implant insertion. In this field there is a huge demand from surgeons to be able to predict, before performing the surgery, the new facial outlook of the patient. Actually, there is a growing interest in CASS since it helps during surgery planning, and improves the communication between dentists, orthodontists and surgeons. For some application examples in this area we refer the reader to the following research works [15], [3], [21].

The medical information acquired in the diagnostic and planning phase can be very varied; nonetheless, one of the most common techniques is the computed tomography (CT). This source of medical images is usually stored in DICOM (Digital Imaging and Communication in Medicine) format, which is a widely standard used by the medical community. These images encode the volumetric information of the head using pixels that are given in sets of 2D slices. The color of the pixels is expressed in gray levels and they allow identifying different tissues because of their different densities. This procedure is called *segmentation*. After the segmentation, a 3D reconstruction of the different issues is accomplished.

The essential steps of a computer tool capable of planning and simulating maxillofacial surgeries are the following (see Figure 1): read the DICOM files, make the segmentation, reconstruct the 3D model, make the surgery planning, and simulate the surgery.

The specific requirements pointed out by the medical personnel/staff for this tool were:

- Show user-defined specific tissues like soft tissue and hard tissue.
- Show region of interest (ROI) and allow navigation through the whole data.
- Easy and interactive surgery planning module.
- Fast and accurate simulation module.

According to the requirements of the simulation module we developed a simulation module based on mass-spring models (MSM). In a word, MSM assumes that mass is discretely concentrated in mass-points which are connected by springs to each other. Those springs represent the elastic

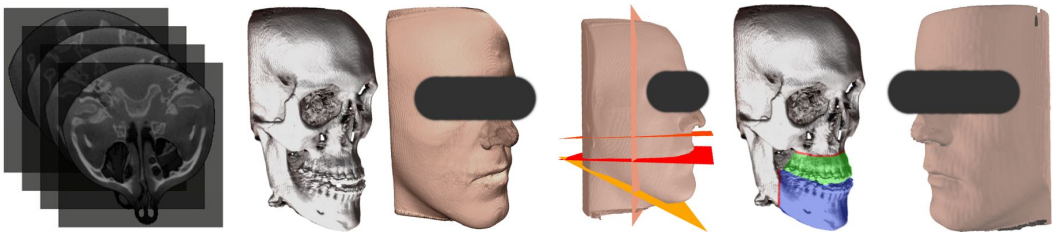

Figure 1. (a) DICOM images, (b) hard and soft tissue before surgery,(c) surgery planning over soft tissue, (d) pieces identified for simulation, and (e) simulated model

behavior of the deformable body. Generally, MSM are considered as fast simulating methods but with the drawback of not being accurate. Thus, we developed a cubical MSM derived from the linear elastic formulation of the finite element method [18]. This way we fulfilled the requirement of little computation time and plausible simulation accuracy.

In the following sections, we are going to focus our attention mainly on the graphical aspect of the computer tool, describing the visualization techniques used in the tool.

2. Hybrid visualization pipeline

In this section, we present the complete visualization pipeline developed to meet the previously mentioned requirements. Our main goal is to provide the surgeon with an inspection tool of the medical data, including a basic navigation through the scanner images, and the 3D reconstruction of the model and its different tissues (such as bone, skin or both). It may also allow the surgeon to inspect with more detail a region of the patient head or add external interaction elements.

Volumetric visualization is a powerful tool in medical data exploration. Among the different volumetric rendering techniques developed, we can point out the full volume rendering (Engel et al. [5], Kaufman & Mueller [8] and Pfister [17] make an extensive overview of this topic). Full volume rendering casts light rays through the volume and composites the final image using the light accumulated. The way samples are accumulated into the final color determines the final image; for example, X-ray images can be obtained with a Σ operation while for more realistic images, material properties can be mapped from user defined transfer functions.

The pipeline is shown in Figure 2 and has the following parts:

- **Data storage:** the volumetric data is loaded into a 3D texture and gradients are precomputed.
- **Transfer function definition:** it is used to assign different visual properties to each tissue, such as material and opacity.
- **Cutting planes definition:** user can define a plane to inspect the interior of the data.
- **Proxy geometries rendering:** this part is used for both rendering the whole dataset or to define regions of interest.
- **Ray casting:** light is accumulated along a ray through the volume, resulting in the color displayed.
- **Composition with polygonal objects:** finally, the image from the ray caster is composite with the polygonal meshes, such as the tools or the interface.

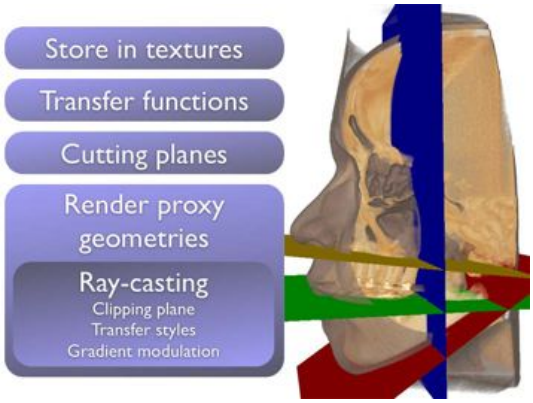

Figure 2. Hybrid visualization pipeline overview.

2.1 Storage

The patient medical data comes from Icat images, which consist in 555 slices with a dimension of 400×400. To

store all this information, along with precomputed gradients needed in some visualization stages, 1.32GB of video memory are needed, which is not currently available in commodity graphic hardware. Another problem is the number of samples needed for a proper visualization; the higher resolution has the model, the more samples are needed, which has a negative impact in the frame rate and user interaction.

In order to reduce the size of the data, and since DICOM voxels have only 11 significative bits of information and gradients are used only as an approximation of the normal, we reduce the precision from 32 to 16 bits for each component (16 bits for density, and 3×16 bits for gradient), moving the memory requirements down to 660MB. Although this is enough in some modern graphic cards, it does not fit yet in most common ones. One possible solution is to store the voxel density in the gradient magnitude, reducing 25% the size of the dataset. Another possible solution is to use octrees codifications, for which some GPU implementations have been presented, such as Lefebvre et al. [11]. The main drawbacks are that octrees do not guarantee a constant size since they depend on data values (some models can still not fit into memory), and at the same time they introduce an overhead which may be too expensive: additional fetches to read the octree structure. In addition, both solutions still are affected (although octrees less than density packaging) by the number of samples needed for a proper visualization. Since interaction is crucial, we chose to reduce the resolution of the dataset by two ($277 \times 200 \times 200$). This reduced dataset, with only 82.5MB, is stored in a 3D texture with four components (RGB channels for the gradient, and the alpha channel for density), using 16 bits floating point precision.

In the validation process of the software, surgeons were very satisfied with the images quality, and interaction was not sacrificed. Nevertheless, as graphic hardware is becoming more and more powerful, we plan to incorporate lossless octree codification into the volumetric visualization as a future work when it does not reduce the performance under interactive rates.

In order to simplify the simulation of the surgery and the definition of the reference planes, it is assumed that the patient's head orientation is perpendicular to the camera. So, just after the model is loaded, it must be manually aligned in order to set its correct orientation. This alignment consists in a rotation around the Z axis and a translation in the X axis. Such operations are performed directly in the GPU memory using the OpenGL 2.0 extension render to 3D texture: each re-oriented slice is rendered as a textured quad, using linear interpolation to avoid aliasing. A temporary 3D texture is needed, since if it is rendered in-situ, interpolation uses data from both the previous, already reoriented, and the next, unmodified slice, causing a ghost image; a temporary texture avoids the new reoriented image overwrites the original data, thus the ghost effect is not produced.

2.2 Ray casting

Before explaining the rest of the parts of the visualization pipeline, we will explain the ray casting part since here are exposed key concepts about volume rendering. As it was mentioned before, full volume rendering consist in casting light rays through the volume and using the light accumulated to composite the final image. For a more general and useful visualization, the accumulation function needs to approximate light absorption, occlusion and emission when the ray casts the volume data. Equation 1 describes the accumulation function $I(x)$ along the ray x for a back to front mapping. α and c are the opacity and the material for each sampled point. L is the number of steps along the ray, which must satisfy the Nyquist-Shannon theorem, i.e., L must be as minimum twice the number of voxels the ray will sample.

$$I(x) = \sum_{l=0}^{L} \left(c_l \alpha_l \prod_{j=0}^{l-1} (1 - \alpha_j) \right) \tag{1}$$

The ray $I(x)$ is generated using the entry-exit points approach of Krüger & Westermann [10]. The proxy geometry of the volume data is rendered in two steps, back (exit points) and front (entry points) faces of the geometry. The entry-exit points are computed using the per-vertex color of the proxy geometry. In general, the proxy geometry is the volume bounding box, but it can be used to change the rendered volume or to deform it. In Section 2.5 we make use of this property to create regions of interest and clipping planes.

Finally, it is important to note that each ray is a pixel of the rendered image, so the ray-caster can be implemented directly in the GPU using a fragment shader. Our implementation has shown interactive frame rates (15fps) even at 1280×1024 using a NVIDIA 9800 with 512MB of memory. As an extra, we have also implemented a stereoscopic visualization which runs at interactive times at 1024×768.

2.3 Transfer function definition

One way to visualize distinct tissues is the use of different materials associated to the correspondent range of densities of each tissue. These materials provide a set of properties such as color and opacity, which are mapped from the volumetric data using transfer functions. If only the scalar value of the voxel (in our case the tissue density) is mapped, then they are called 1D transfer functions. The problem

is that they do not provide enough material information in many cases, produce highly artificial images and, in order to get more realistic results, light computation must be included in the ray caster. Other volume information can be used to improve color mapping, such as gradient or curvature. This leads to multidimensional transfer functions that show better results but at the expense of a harder and less intuitive creation of the transfer function by the user (even using modern design techniques such as the widget based proposal of Kniss et al. [9]), which is beyond our usability requirements.

Figure 3. Transfer styles can represent both specular and diffuse materials.

Bruckner & Gröller [1] have proposed the use of transfer styles to add more information preserving the simplicity of 1D transfer functions. Transfer styles use sphere maps to create an image-based illumination model. The main idea is to associate a sphere map to each material in the same way a set of properties are assigned with transfer functions, and then capture color variations as a function of eye-space normal direction. In this way, two problems are solved at once: material information and the illumination model (see Figure 3). Figure 4 shows alternative uses of transfer styles to incorporate other illustrative techniques such as colored borders (where the sphere map of the style has a colored ring in its border, [1]). We have also designed specific styles to add transparency in the region of interest only for certain orientations (in this case the sphere map opacity is set to zero in that zone). In total, our tool offers the surgeon four built-in visualization styles (exterior appearance (skin only), hard tissue and translucent soft tissue over hard tissue) where each of them comes with different materials for better data understanding (such as photo-based images of soft and hard tissue, border-colored styles, highly specular materials and translucent styles).

When working with material sets, each material M_i is

Figure 4. Other applications of transfer styles for illustrative rendering: the left image shows borders colored and the right image transparency of a region of interest.

defined inside a density range $[x_i^0, x_i^1]$ and has associated a transfer style S_i and an opacity level α_i. $\alpha(x)$, is then a step opacity function, easy to define and very intuitive but which may produce some aliasing artifacts (Figure 5).

Figure 5. Differences between a step linear opacity function (left) and piecewise function based on Bézier curves (right).

To reduce these artifacts, we have developed an opacity level smoother in the transfer function definition. This operator replaces the step function with a piecewise Bézier curve $A(x)$ with control points X_i and Y_i, modulated by a softness intensity parameter β (Equation 2). A value $\beta = 0$ leads to the original step opacity function. As the opacity function is computed only when the transfer function changes, the additional cost of computing the Bézier curve is negligible.

$$
\begin{aligned}
&\tilde{x} = x - x_i^0 \\
&\Delta_i = x_i^1 - x_i^0 \\
&X_i^0 = \{0, 0, -\tfrac{1}{2}, \tfrac{1}{2}\} \\
&Y_i^0 = \{1-\beta, 1, 1, 1\} \\
&X_i^1 = \{\tfrac{1}{2}, \tfrac{3}{2}, 1, 1\} \\
&Y_i^1 = \{1, 1, 1, 1-\beta\}
\end{aligned}
\tag{2}
$$

$$
A_i(x) = \alpha_i * \begin{cases} bezier(\tilde{x}, X_i^0, Y_i^0) & \tilde{x} < \Delta_i/2 \\ bezier(\tilde{x}, X_i^1, Y_i^1) & \tilde{x} >= \Delta_i/2 \end{cases}
$$

One of the key advantages of multidimensional transfer functions is the possibility to increase visual differentiation between materials using the gradient as another lookup value, but at the expense of a more complicated user interaction, as mentioned above. Instead, we have used a gradient-magnitude opacity modulation to remark density changes [12] (Figure 6). One common technique to estimate the gradient is the central differences method (Equation 3). As it has been mentioned above, the gradient estimation is performed in a pre-process and stored in the volume texture.

Figure 6. Volume rendering with and without gradient modulation. Note that gradient modulation avoids thicker regions cause unnecessary light absorption.

$$
\begin{aligned}
f_x(x,y,z) &= f(x+1,y,z) - f(x-1,y,z) \\
f_y(x,y,z) &= f(x,y+1,z) - f(x,y-1,z) \\
f_z(x,y,z) &= f(x,y,z+1) - f(x,y,z-1)
\end{aligned}
\tag{3}
$$

2.4 Regions of interest and clipping planes

We have extended the [10] proposal to visualize regions of interest and to apply cutting planes, by changing the proxy geometry rendered to the desired region or clipping it with the cutting planes. The main restriction this method has is that the proxy geometry must be convex in camera-space, since only one pair of entry-exit points is possible per fragment. If more complex geometries are rendered, a surface ordering technique is mandatory ([16]), together with a multipass of the ray-caster. In our visualization module, no such complex concave interest regions are needed, so basic proxy geometries, such as boxes or spheres, are used (Figure 7).

Proxy geometries must contain information about the entry and exit points of the rays in the form of colors, and it complicates the use of user defined regions. To solve this, we have developed a vertex shader to simplify the generation of proxy geometry colors based on the bounding box of the proxy geometry. Let B be the bounding box and C the center of the proxy geometry for a volume of dimensions $W \times H \times D$, the color $\langle R, G, B \rangle$ of a vertex of the proxy geometry in world coordinates is given by:

$$
\begin{aligned}
R &= \left((x + C_x) * \left(B_x^{max} - B_x^{min}\right) + B_x^{min}\right)/W \\
G &= \left((y + C_y) * \left(B_y^{max} - B_y^{min}\right) + B_y^{min}\right)/H \\
B &= \left((z + C_z) * \left(B_z^{max} - B_z^{min}\right) + B_z^{min}\right)/D
\end{aligned}
\tag{4}
$$

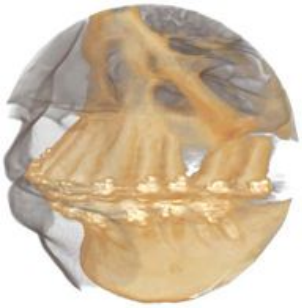

Figure 7. Spherical region of interest around the mandible.

In addition, we included a single cutting plane as a ray tracer parameter to reduce the complexity of proxy geometry modifications (single cutting planes are the most common when inspecting the data). This parameter is used to change the starting or ending point of the ray, depending on the plane orientation to the camera (Figure 9).

Finally, we have found proxy geometries useful for volume immersion too (Figure 8). It is necessary to apply some modifications to the traditional rendering with proxy geometries, because they are clipped against the near-Z plane when the camera approaches, so no start points for rays are drawn in the clipped area. To solve this issue, the intersection of the proxy geometry with the near-Z plane is computed and rendered along with the proxy geometry. In this way, start points are available to the ray-caster. It is basically the same concept applied to cutting planes, but the

main difference is that cutting planes imply the computation of a new proxy geometry, while immersion makes use of the pipeline to clip the proxy geometry, and only calculates the intersection polygon. And as in our case proxy geometries are convex (boxes, spheres), any intersection with a plane is also convex; thus only the intersection points are required to be computed, and the near-Z polygon can be constructed sorting these points around their centroid and the camera direction.

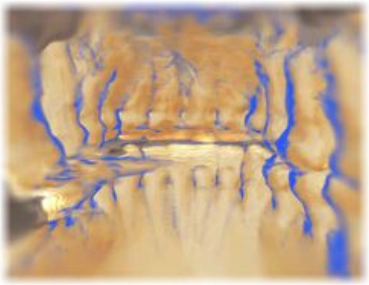

Figure 8. View from inside the volume.

2.5 Volumetric and polygonal visualization

Although volumetric visualization provides great information about patient data, it lacks of a natural way to easily introduce additional elements such as interaction objects and the planning definition. The main problem lies in the ray-marching algorithm, which is independent of the environment: the whole volume is casted regardless occlusions caused by objects which intersect with it. In general, polygons must be taken into consideration during the ray-casting using; for example, two ray-casters that work one with the volume data and the other with polygons, mixing the result after each step (Levoy [13]). An easier solution is showed by Nes [16] combining the natural entry-exit points of [10] with a surfaces sorting technique (Everitt [6]) in a multipass process, where surfaces are rendered alternatively as entry and exit proxy geometries. After each pass, surfaces are also rendered and composited into the final image.

We noted that if no transparent objects are used, this method can be simplified to skip the sorting phase, and only one pass for each type of data is needed. Instead of render several times all the objects and peel out ray-caster layers, all the objects including the backfaces of proxy geometry, are rendered in the same pass with the depth test enabled. In this way, ray end points are obtained. After it, and without clearing the depth buffer, frontfaces of the proxy geometry are rendered as usual (see Figure 9). This provides a natural and unipass scheme for mixing opaque polygons and volumetric data.

Figure 9. Occlusal reference plane cutting the volume set, and its interaction gizmo.

3. Surgery planning

The planning process consists in the definition of the different osteotomies specified by the surgeon. If the graphical representation of the surgery is exactly the same as the actual surgery, the surgery planning tool is going to be very complicated. Thus, it is very common to make some simplifications in this step; considering that the influence of these assumptions on the result has to be small compared with the precision error of the surgery.

Though our aim is to predict the aesthetic outcome of the surgery, the osteotomy details that are not going to have a direct effect on the skin of the patient can be simplified. Consequently, the surgery planning, including osteotomies, can be specified using few planes, since they provide enough information about which piece must be moved and its direction, without introducing details which are irrelevant in our problem.

Working with 3D objects is still a difficult task, so we have developed a simpler planning interface where the user defines the planes as straight lines in a 2D visualization (the planes are defined using their projections) at the time that the 3D visualization of the planning can be seen in a secondary visualization, by the use of a mixed volumetric and polygonal rendering. This interface has two different visualization modes to make easier the plane definition: an X-ray like image and a semi-transparent view of the head (both of them are performed using the volumetric visualization). This way, the planning process becomes very intuitive and avoids the problem of using 3D plane definitions, preserving the visual feedback of their position in 3D (Figure 10).

4. Visualization of simulation results

The MSM used in the simulation (cf. Section 1) starts with a configuration equivalent to the uniform grid structure of the data. Unfortunately, this equivalence is no longer true as after simulation the points are not distributed in a grid any longer. These conditions are not ideal for a visualization of

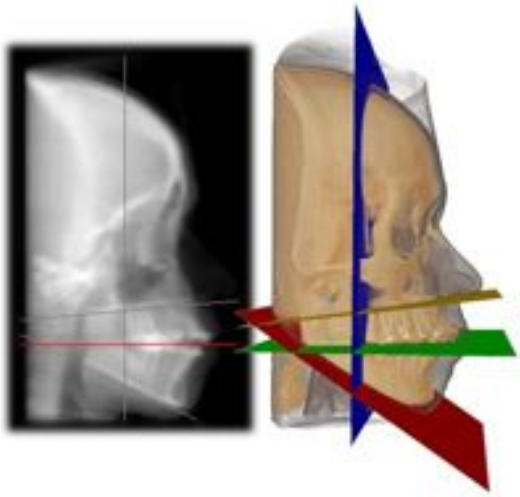

Figure 10. 2D planning showing osteotomies over an X-ray visualization and 3D planning visualization.

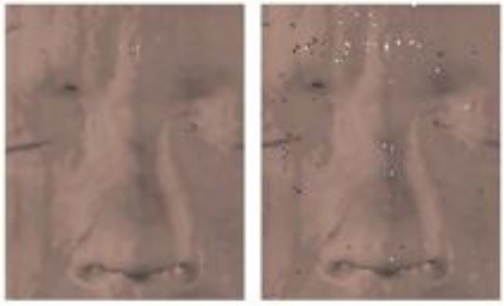

Figure 11. Face reconstruction with and without point perturbation. Without perturbation, points are aligned in a grid, and they are not guaranteed to be correctly reconstructed by Delaunay triangulations.

the results in the same way original data does. The volumetric visualization used in the planning stage is no longer valid because the data is no uniformly grid structured. A point-based visualization could be appropriate for small displacements, but in areas without information, holes may appear when using point visualization. However, since point organization (vertex indices) is not lost in the mesh, a polygonal model of the skin obtained from the pre-simulated points cloud can be used to visualize post-simulated results.

The grid of points used for simulate the surgery, in order to reduce simulation time, has a resolution too low to be used for proper reconstruction and visualization. Instead of this, we extract two models from the dataset, a reduced model for simulation, and a high quality point set of the skin for visualization. Low resolution simulation results are used to morph the high resolution skin.

Traditional iso-surface extraction methods, such as Marching Cubes [14], do not suit well in this morphing problem since they reconstruct a geometry which is not associated with the grid structure that the simulation module uses (i.e., the vertices of the reconstructed geometry are not the vertices of the grid). On the other hand, we have obtained good results working with a surface reconstruction from point sets. More specifically, we have used the method proposed by Buchart et al. [2], called GPU Local Triangulation (GLT), which has shown both good results and performance in our application.

GLT is based on the work of Gopi et al. [7], and performs local Delaunay triangulations in a parallel fashion especially designed to fit GPU requirements. In general, [2] divides the problem in small sets and processes each one independently in the GPU, then it creates triangle fans around each point, and joins them to obtain the global triangulation.

Reconstructions based on Delaunay triangulations [4] guarantee optimal and unique triangulation around each point given no more than two neighbors of a point lies in the same circumcircle with it. Point sets obtained from uniform grids, as our case, do not satisfy this condition since the points are placed in the corners of cubes, so four points may lie in the same circumcircle. To solve this problem we introduce a small random perturbation to the position of the points so they loose their grid structure but preserving the relationship between them. If perturbation is too high (more than the half of the reconstruction radius), points can cross and their relation is lost. If it is too small, numerical errors can still affect the results. Experimentally we established this perturbation to a 5% translation from their original position. After the reconstruction process we discard the perturbation, so the original points are used as mesh vertices.

Our tests have shown reconstruction times between 5 and 10 seconds, depending of the patient, while skin point sets have been around 130K and 140K points. The highest computation times corresponds to patients with more pronounced deformities, or more complex head geometries.

As a result of the noise at top and bottom of the model, the normal propagation phase of the reconstruction may produce undesired results. To overpass this situation, we have replaced normal propagation with a global normal reorientation: starting with the assumption that a head is homomorphic to a sphere, we force all normals to face out the head. Here, some zones have been incorrectly reoriented (e.g. below the eyebrows and the lower lip). Normals are then grouped in uniform regions, which frontiers are rings where normals have an abrupt change. Starting with a known normal (such as the tip of the nose), regions

are classified as correct or incorrect. Incorrect regions are then re-oriented. Note that this method works well for the head, but very noisy regions, such as the neck bottom or the top of the head, may still remain incorrect. As these are not zones of interest, we do not perform any further action on them.

5. Conclusions

In this paper, a hybrid visualization pipeline for maxillofacial surgery has been presented. A volumetric visualization pipeline has been developed and different imaging algorithms have been implemented to get a useful visualization for surgeons. As the volumetric visualization is not flexible enough to fulfil all the requirements of the surgeon, we have completed the tool with additional visual functionalities. Mixing the volumetric visualization with polygonal rendering and allowing basic geometric operations on the model, such as regions of interest and cutting planes, has provided a better and more intuitive tool for surgeons.

A custom 2D planning approach with 3D interactive visualization of the planning has been presented as well. This planning is based on the fact that many surgery details do not have influence on the external appearance of the patient, and therefore can be simplified, reducing the planning complexity without reducing the simulation quality. This fact has been remarked by the surgeons that have tested the application, because the time they spend in the planning stage has been reduced.

Finally, simulation results have been visualized using a GPU surface reconstruction algorithm. In this issue, special attention has been paid to the medical data representation in order to avoid wrong reconstructions caused by the Delaunay triangulation method.

References

[1] S. Bruckner and M. E. Gröller. Style transfer functions for illustrative volume rendering. *Computer Graphics Forum*, 26(3):715–724, 2007.

[2] C. Buchart, D. Borro, and A. Amundarain. Gpu local triangulation: an interpolating surface reconstruction algorithm. *Computer Graphics Forum*, 27(3):807–814, 2008.

[3] M. Chabanas, Y. Payan, C. Marecaux, P. Swider, and F. Boutault. Comparison of linear and non-linear soft tissue models with postoperative ct scan in maxillofacial surgery. In *International Symphosium in Medical Simulation*, pages 19–27, 2004.

[4] B. Delaunay. Sur la sphere vide. a la mémoire de georges voronoi. *Bulletin of Academy of Sciences of the USSR*, 7:793–800, 1934.

[5] K. Engel, M. Hadwiger, J. M. Kniss, A. E. Lefohn, C. R. Salama, and D. Weiskopf. Real-time volume graphics. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Course Notes*, page 29, New York, NY, USA, 2004. ACM.

[6] C. Everitt. Interactive order-independent transparency. Technical report, NVIDIA Corporation, 2001.

[7] M. Gopi, S. Krishnan, and C. T. Silva. Surface reconstruction based on lower dimensional localized delaunay triangulation. *Computer Graphics Forum*, 19(3):467–478, 2000.

[8] A. Kaufman and K. Mueller. *Overview of Volume Rendering (The Visualization Handbook)*, chapter 7, pages 127–174. Elsevier Academic Press, 2005.

[9] J. Kniss, G. Kindlmann, and C. Hansen. Multidimensional transfer functions for interactive volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):270–285, 2002.

[10] J. Kruger and R. Westermann. Acceleration techniques for gpu-based volume rendering. In *VIS '03: Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, page 38, Washington, DC, USA, 2003. IEEE Computer Society.

[11] S. Lefebvre, S. Hornus, and F. Neyret. *Octree Textures on the GPU (GPU Gems 2)*, chapter 37, pages 595–613. Addison-Wesley, 2005.

[12] M. Levoy. Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, 8(3):29–37, 1988.

[13] M. Levoy. A hybrid ray tracer for rendering polygon and volume data. *Computer Graphics and Applications, IEEE*, 10(2):33–40, Mar 1990.

[14] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH 1987 Proceedings*, pages 163–169. ACM, 1987.

[15] W. Mollemans, F. Schutyser, N. Nadjmi, F. Maes, and P. Suetens. Predicting soft tissue deformations for a maxillofacial surgery planning system: From computational strategies to a complete clinical validation. *Med Image Anal*, 11(3):282–301, 2007.

[16] G. Nes. Mixed volume/mesh rendering using depth peeling. Student Project in the Course INF219: Project in Visualization, Fall 2007.

[17] H. Pfister. *Hardware-Accelerated Volume Rendering (The Visualization Handbook)*, chapter 11, pages 229–258. Elsevier Academic Press, 2005.

[18] G. San Vicente, C. Buchart, D. Borro, and J. T. Celigüeta. Maxillofacial surgery simulation using a mass-spring model derived from continuum and the scaled displacement method. *International Journal of Computer Assisted Radiology and Surgery*, 4(1), 2009.

[19] J. Xia, J. Gateno, J. Teichgraeber, A. Christensen, R. Lasky, J. Lemoine, and M. Liebschner. Accuracy of the computer-aided surgical simulation (cass) system in the treatment of patients with complex craniomaxillofacial deformity: A pilot study. *Journal on Oral Maxillofacial Surgery*, 65(2):248–254, 2007.

[20] J. Xia, C. Phillips, J. Gateno, J. Teichgraeber, A. Christensen, M. Gliddon, J. Lemoine, and M. Liebschner. Cost-effectiveness analysis for computer-aided surgical simulation in complex cranio-maxillofacial surgery. *Journal on Oral Maxillofacial Surgery*, 64(12):1780–1784, 2006.

[21] S. Zachow, H. C. Hege, and P. Deuflhard. Computer-assisted planning in cranio-maxillofacial surgery. *Journal of Computing And Information Technology - Special Issue on Computer-based Craniofacial Modeling and Reconstruction*, 14(1):53–64, 2006.