CEIG - Spanish Computer Graphics Conference (2012)
Isabel Navazo and Gustavo Patow (Editors)

An application to interact with 3D models reconstructed from medical images

F. Paulano[1], R. Pulido[1] and J.J. Jiménez[1]

[1]Computer Science Department, University of Jaén, Spain

Abstract

Although the reconstruction of 3D models from medical images is not an easy task, there are many algorithms to perform it. However, the reconstructed models are usually large, have a lot of outliers and have not a correct topology. For that reason, the algorithms used to interact with them must be fast and robust. In this paper, we present an application that enables the interaction with models reconstructed from medical images. The application uses Marching Cubes to generate triangle soups from the medical images. Then, the user can define models by selecting a set of triangles. Once the models have been defined, the application allows to interact with them. In addition, a detailed collision detection is calculated between the models in the scene in order not only to avoid that models in the scene collide, but also to determine which triangles are overlapping. In addition, the calculation of distances and nearest points provides visual aid when the user is interacting with the models.

Categories and Subject Descriptors (according to ACM CCS): I.3.8 [Computer Graphics]: Applications—

1. Introduction

Nowadays, it is very common to work with polygonal models of the human body. In most cases, these models are generated from scans of real patients, since it allows to customize the simulation. However, the reconstruction of 3D models from medical images is not an easy task. Usually, the reconstructed models are not topologically correct or even they have no topological information. In addition, these models are huge and have a lot of outliers in most cases. Once the models are reconstructed, it is necessary to develop tools that allow to interact with them. These tools should include, among others, methods of visualization, picking, selection of parts, elimination of outliers and collision detection [dB03]. Because of the complexity of the reconstructed models, these methods must be fast and robust.

Applications that are usually used to work with medical images are limited to 3D visualization and they usually only allow to rotate the camera and to remove outliers if it is necessary. Therefore, this type of applications do not enable interaction. They do not allow the user to select models independently and move or associate them. It is necessary to develop an application that lets the user interact with a model individually. In this paper, we present an application to interact with 3D models reconstructed from medical images. This can be used in the field of education so users can translate and rotate individual body parts to examine them or can use proximity queries and collision detection methods in order to help the user to place body parts in their correct position.

2. Previous work

In recent years, some papers have been published for working with volumetric data generated from medical images. In [BnK96], authors extract small structures from CT scans by using iso-surface generation and spatial region growing techniques. In [BD02], a toolkit to visualize large volumetric data sets is presented. To this end, the toolkit makes use of 3D clip-textures. In [RBG07], authors, instead of using the traditional transfer function specification, use semantic layers to define the mapping of volumetric attributes, specified by rules evaluated with fuzzy logic, to a specific visual style. In [ENS*12], a framework to check topological properties of isosurfaces is extended. There are some research groups in Spain that have built other platforms to manage models reconstructed from medical images. The CEIT has developed an application, called Viewit Spine, to help the surgeons to place implants [CEI11]. The MoViBio group in the UPC has implemented BioMedIGU [AVT05], a portable library to create interfaces for biomedical imaging software.

DOI: 10.2312/LocalChapterEvents/CEIG/CEIG12/041-044

3. Overview of the application

The graphic user interface is mainly composed of four canvas and a hierarchy tree. The main canvas (Figure 1, A) allows to move the camera and the other three show static views: side, front, and raised view (Figure 1, B). All canvas can exchange their position dynamically by clicking any of the auxiliary canvas or using the buttons on the top of the window (Figure 1, C). On the left side of the application window, a hierarchy tree shows the relation between the models and the model that is currently selected (Figure 1, D). At the top of the window there is a toolbar that allows to load medical images, define models and their relationship, remove unnecessary parts, and enable the calculation of nearest points and overlapping triangles (Figure 1, E). The application generates a triangle soup representing the model. Models must be identified and defined using an area selector and relationships between models must be defined. In this step, outliers can be removed by using the selector. Once a model is identified, it can be selected, translated and rotated to place it correctly in the scene. To simplify this step, the application calculates the nearest points and the overlapping triangles between every two interrelated models. The application has been implemented in C++ using the gcc compiler and following a MVC architecture. The user interface has been developed using the Qt 4.7 libraries [BS06]. Therefore, the implemented software can be compiled for various platforms. The QtOpenGL module has been used to implement 3D graphics. Furthermore, the application uses VTK [SML06] to reconstruct the 3D models and PQP [LGLM99] to calculate a detailed collision detection.

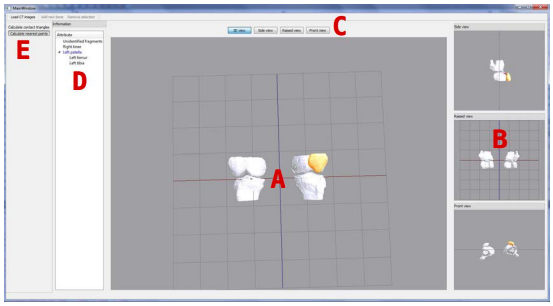

Figure 1: *Screen capture of the application. A - Main canvas. B - Secondary canvas. C - Buttons to exchange canvas. D - Hierarchical tree of the scene. E - Toolbars.*

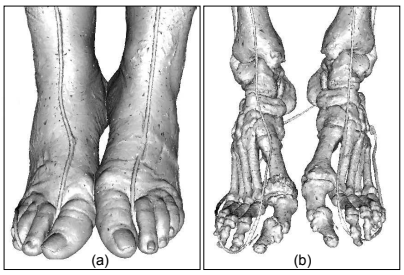

Figure 2: *Iso-surface reconstruction from volumetric data by using Marching Cubes. Feet reconstruction from both (a) skin tissue and (b) bone tissue iso-values.*

4. Model reconstruction from medical data

Model reconstruction from medical data is a difficult task due to the complexity of the human body. In addition, the results depend on several issues involved in the process.

4.1. Medical images

There are different sources such as Computed Tomography (CT), Magnetic Resonance Imaging (MRI) or Ultrasonic (US) techniques [Sac04] that provides 3D datasets and values. Each type of medical image presents special features that enable the classification of different tissue types such as bone, muscle or fat. Our application allows to visualize 2D images and works with 3D medical image series and it has been tested with DICOM datasets and images from The Visible Human Project repository [NLM86].

4.2. 3D Model reconstruction

In recent years several approaches to reconstruct 3D surfaces from medical images have been presented. Some techniques have been adapted from popular methods in computational geometry such as Delaunay triangulation and Voronoi diagram [LYG*09]. Other techniques are based on Hierarchical Space Subdivision Schemes [BHGS06]. Several of these approaches could be combined to obtain geometrically valid surfaces. The extraction of contours, using a hierarchical spatial decomposition, can be complemented with a table of patterns to triangulate these contours [PJP12]. Our application makes use of the Marching Cubes algorithm [LC87]. Marching Cubes is a fast and simple method that allows to generate large polygonal datasets from volumetric data with high resolution (see Figure 2). The algorithm requires a high computational cost and holes can appear on the generated iso-surface due to ambiguity cases. However, the presented application is able to deal with surfaces that present holes by using the interaction techniques described below.

5. Interaction

Once the 3D model has been reconstructed, the interaction is enabled. For that reason, it has been necessary to implement registration, collision detection, picking and multiview methods. Due to the size and the topology of the reconstructed models, those methods must be fast and robust.

5.1. Registration of the models

Due to the fact that the reconstructed models are triangle soups, there is no information about the model to which each

triangle belongs. To solve this deficiency, an area selector has been implemented. This tool allows to select a set of triangles and to define a model from them. When a model is defined, the application allows to relate it to another previously defined model. The area selector can be also used to remove triangles from the scene (Figure 3). Each time the user makes a selection, the scene has to be repainted to draw the selector. To solve this problem, we avoid repainting the entire scene by using a frame buffer. In the rendering function, the content of the frame buffer is drawn in a 2D texture which is located in the background of the scene. Moreover, to determine which triangles have been selected they are projected in the plane determined by the selector. Thus, it is only necessary to resolve a triangle-rectangle test for each triangle in the scene. Moreover, as the intersection is calculated in 2D, it is easy to implement new shapes for the selector.

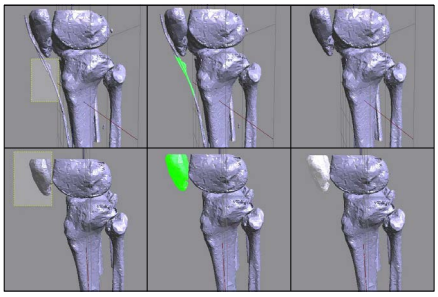

Figure 3: *The area selector is used to remove unnecessary parts (top) and to define models (bottom).*

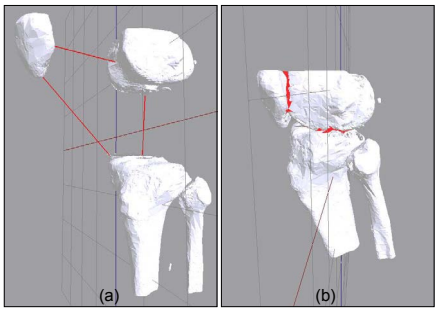

Figure 4: *a) Calculation of the nearest points between some models. b) Overlapping triangles are displayed in red.*

5.2. Collision detection

In order to implement the interaction between the different models, we have not only to calculate the collision but also to generate a realistic response. Since the reconstructed models are triangle soups, it is necessary to use algorithms that can work with that type of models. Distances and nearest points between two models can help the user to place models that are not correctly positioned (Figure 4). Some algorithms have been tested [PJPO12] to choose the best suited to this problem. In this work, algorithms have been tested with models with a complexity of up to one million triangles. The results shown that the PQP library [LGLM99] is quite fast and robust. This library uses swept sphere hierarchies to perform collision detections. Moreover, PQP is able to calculate different collision detection parameters, such us distances, overlapping triangles, or closest points.

5.3. 3D picking

After the models are defined, the application allows to select them by picking. Each selected model can be rotated and translated by using the mouse. These transformations are performed independently of the other models in the scene. In order to pick objects, we have implemented a method that consist of casting a ray from the observer that pass though the mouse position and the calculation of the collision with all the objects in the scene by using the PQP library. In this way, the data structures previously constructed are used. To measure the efficiency of the method, some tests have been performed. As shown in table 5.3, the method calculates the ray picking in a reasonable good time, even when there are several hundred of thousand of triangles in the scene. Moreover, since it is based on PQP, the method is also robust.

Triangles in the scene	Picking time (s)
58206	0.0671
253696	0.2868
428411	0.3845
478462	0.4231
513658	0.4608
546308	0.4949
577827	0.5206

Table 1: *Picking time using the ray picking method based on PQP (Intel i7 2,80GHz, 4GB RAM, GeForce GTS 240)*

5.4. Multi-canvas

The 3D view provides extra information to the user. However, doctors are used to have a 2D view of the patient area. For this reason, the application includes four canvas: a canvas with a free camera and three canvas with static cameras. The first one is the main canvas and the other three canvas represent a raised, side, and front view respectively. Instead of implementing multi-canvas using several glViewport definitions [Shr10], we decided to used four QGLWidget to implement the four canvas. In this way, we can take advantage of all the functionality that is already implemented in the QtOpenGL module. Therefore, the application uses four QGLWidget instances to implement the four canvas [BS06]. In comparison with other approaches, the use of QGLWidget instances allows to easily implement the mouse events for each canvas. Each QGlWidget has an associated QDockWidget that places it in the application window. The four

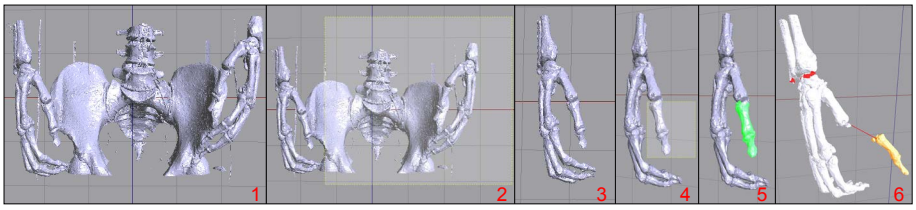

Figure 5: *Example of using the application. 1- Triangle soup generated from CT. 2- Selection of some unnecessary triangles. 3- Scene after removing all unnecessary triangles. 4- Selection of a model. 5- Selected model. 6- Interaction with defined models.*

canvas can exchange their positions dynamically by clicking them or by pushing the buttons located at the top of the window. To achieve this, an own camera class was implemented.

6. Conclusions

In this work, an application to interact with 3D models reconstructed from medical data has been developed. Unlike other existing applications that enable 3D visualization of medical images, the presented application can interact with the 3D models in terms of geometry because it previously performs a reconstruction. Moreover, the application enables a real-time interaction, although the reconstruction generates large models with no topology. Finally, the following tasks are proposed as future work:

- Integrating stereoscopic visualization systems.
- The presented application can be applied in preparing various types of surgical procedures.
- Measuring the usability of the application.
- Obtaining topologically correct meshes from triangle soups.
- Incorporating a method to remove outliers automatically that avoids the user having to delete them manually.
- Improving the area selector by using a spherical selector.

Acknowledgements

This work has been partially supported by the Ministerio de Economía y Competitividad and the European Union (via ERDF funds) through the research project TIN2011-25259 and by the University of Jaén through the research project UJA2010/13/08 sponsored by Caja Rural de Jaén.

References

[AVT05] ABELLAN P., VERGÉS E., TOST D.: Design of graphical interfaces for biomedical applications. In *Actas del VI Congreso de Interacción Persona-ordenador* (2005), pp. 145–148. 1

[BD02] BHANIRANTKA P., DEMANGE Y.: Opengl volumizer: a toolkit for high quality volume rendering of large data sets. In *IEEE/ACM SIGGRAPH Symposium on Volume Visualization and Graphics* (2002), pp. 45–53. 1

[BHGS06] BOUBEKEUR T., HEIDRICH W., GRANIER X., SCHLICK C.: Volume-surface trees. *Computer Graphics Forum (Proc. of EUROGRAPHICS 2006) 25*, 3 (2006), 399–406. 2

[BnK96] BRO-NIELSEN M., KREIBORG S.: Virtual teeth: A 3d method for editing and visualizing small structures in ct scans. In *Computer Assisted Radiology. Proc. of the Int Symp. on Comp. and Comm. Systems for Image Guided Diagnosis and Therapy* (1996), pp. 921–924. 1

[BS06] BLANCHETTE J., SUMMERFIELD M.: *C++ GUI Programming with Qt 4*. Prentice Hall Open Source Software Development Series. Prentice Hall PTR, 2006. 2, 3

[CEI11] CEIT: Viewit spine, 2011. http://www.ceit.es/en/press-room/you-may-be-interested/585-neurosurgery-in-3d. 1

[dB03] DER BERGEN G. V.: *Collision Detection in Interactive 3D Environments*. Elsevier, 2003. 1

[ENS*12] ETIENE T., NONATO L., SCHEIDEGGER C., TIENRY J., PETERS T., PASCUCCI V., KIRBY R., SILVA C.: Topology verification for isosurface extraction. *Visualization and Computer Graphics, IEEE Trans. on 18*, 6 (2012), 952 –965. 1

[LC87] LORENSEN W. E., CLINE H. E.: Marching cubes: A high resolution 3d surface construction algorithm. *COMPUTER GRAPHICS 21*, 4 (1987), 163–169. 2

[LGLM99] LARSEN E., GOTTSCHALK S., LIN M., MANOCHA D.: *Fast Proximity Queries with Swept Sphere Volumes*. Tech. rep., Dep. of Computer Science, UNC Chapel Hill, 1999. 2, 3

[LYG*09] LV S., YANG X., GU L., XING X., PAN L., FANG M.: Delaunay mesh reconstruction from 3d medical images based on centroidal voronoi tessellations. In *Int. Conference on Computational Intelligence and Software Engineering* (2009). 2

[NLM86] NLM: The visible human project, 1986. http://www.nlm.nih.gov/research/visible/visible_ human.html. 2

[PJP12] PULIDO R., JIMÉNEZ J. J., PAULANO F.: Surface reconstruction from 3d medical images based on tri-tree contouring. In *Proc. of the International Conference on Computer Graphics Theory and Applications* (2012), pp. 175–181. 2

[PJPO12] PAULANO F., JIMÉNEZ J. J., PULIDO R., OGAYAR C. J.: A comparative study of implemented collision detection strategies. In *Proc. of the International Conference on Computer Graphics Theory and Applications* (2012), pp. 485–490. 3

[RBG07] RAUTEK P., BRUCKNER S., GROLLER M.: Semantic layers for illustrative volume rendering. *IEEE Trans on Visualization and Computer Graphics 13*, 6 (2007), 1336–1343. 1

[Sac04] SACHSE F.: 5. digital image processing. In *Computational Cardiology*, vol. 2966 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2004, pp. 91–118. 2

[Shr10] SHREINER D.: *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Versions 3.0 and 3.1*. OpenGL Series. Addison-Wesley, 2010. 3

[SML06] SCHROEDER W., MARTIN K. M., LORENSEN W. E.: *The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics*. Kitware, Inc., 2006. 2