An Application to Interact with 3D Models Reconstructed from Medical Images

Félix Paulano, Juan J. Jiménez and Rubén Pulido
Computer Science Department, University of Jaén, Jaén, Spain

Keywords: 3D Interaction, Stereoscopy, CT Scans, Model Reconstruction.

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. To interact with these models, the methods 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 scans. Then, the user can define models by selecting sets 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 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. Finally, the Leonar3Do system have been incorporated to improve the interaction and to provide stereo visualization. The presented application can be used in the field of education since users can manipulate individual body parts to examine them. Moreover, the application can be utilized in the preparation of an intervention or even as a guide for it, since it enables the utilization of models reconstructed from real medical scans.

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 geometry is not topologically correct or even topological information is not available. This geometry is huge and has a lot of outliers in most cases. In addition, the different models represented by the reconstructed geometry are not isolated or labelled. For all these reasons, it is necessary to develop tools that allow to interact with the reconstructed geometry. These tools should include, among others, visualization, picking, area selection and collision detection methods.(der Bergen, 2003)(Jiménez et al., 2006).

Due to the fact that the different models represented by the reconstructed geometry are neither isolated nor labelled, a tool that allows to select pieces of geometry and define models from them is necessary. This allows the user to interact with the models independently. In addition, this type of tools helps to clean from the scene the not interesting geometry. Once the models are defined, picking methods are needed. Since this type of methods allows to select the model to interact with, they enable the interaction with a model independently from the others. The integration of a detailed collision detection provides visual aid during the interaction and allows to detect which models are colliding or even the colliding zones. This information is useful to avoid that two models collide and then generate a response to the collision. Moreover, it also can be helpful to inform the user that the models collide, hence the user can act accordingly during the interaction. If the collision detection is not calculated, the user can find difficulties when placing the models is the scene. Since time is a very important factor in medicine, stereo display and 3D interaction systems can be integrated to improve the usability of the application and reduce the usage time. Because of the complexity of the reconstructed models, all these techniques and systems must be fast and robust. 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 (Lv et al., 2009). Other techniques are based on Hierarchical Space Subdivision Schemes (Boubekeur et al., 2006). In (Sharf et al., 2007) presented a method to interactively reconstruct surfaces by using a prior distribution. On the other hand, the extraction of contours, using a hierarchical spatial decomposition, can be complemented with a table of patterns to triangulate these contours (Pulido et al., 2012). Applications that are usually used to work with medical images are limited to 3D visualization (Bhanirantka and Demange, 2002) and they usually only allow to rotate the camera and to remove outliers if it is necessary. In (Rautek et al., 2007), authors, instead of using the traditional transfer function specification, use semantic layers to define the mapping of volumetric attributes to a specific visual style. This mapping is specified by rules which are evaluated with fuzzy logic arithmetic. There are several toolkits which enable the manipulation of models reconstructed from medical scans. ITK-SNAP (Yushkevich et al., 2006) is a application that allows to interactively segment structures from medical scans. MITK (Wolf et al., 2004) combines VTK and ITK to provide interaction with models reconstructed from medical scans. Most of these methods and tools work with volumetric models and do not allow to separate and classify parts of the model and interact with them. Thus, it could be useful to develop an application that lets the user interact with a model individually. In addition, 3D motion tracking devices could allow to improve the user experience and could make the application easy to use.

In this paper, we present an application that enables the examination of 3D models, which are generated from medical images, in detail. This can be used in the field of education since users can translate and rotate individual body parts to examine them. Moreover, proximity queries and collision detection methods can help the user to place body parts in their correct position. On the other hand, the application can be used in the preparation of an intervention or even as a guide for it, since it allows to work with models reconstructed from real medical images. For all the above reasons, the application can be used to improve the visualization and thus the diagnosis.

2 OVERVIEW OF THE APPLICATION

The graphic user interface is mainly composed of four canvas and a hierarchy tree. The main canvas (Figure1, A) allows to move the camera and the other three show static views: side, front, and top view (Figure1, B). All canvas can exchange their position

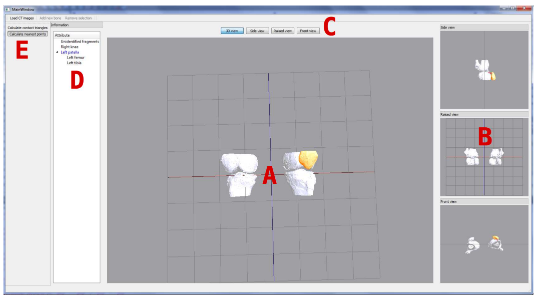

Figure 1: Screen capture of the application. A - Main canvas. B - Secondary canvas. C - Buttons to exchange canvas. D - Hierarchical tree of the scene. E - Toolbars.

dynamically by clicking on any of the auxiliary canvas or using the buttons on the top of the window (Figure1, C). This interface is similar to that used in current medical visualization applications, hence it is easy to use for professionals. On the left side of the application window, an hierarchy tree shows the relation between the models and the model which is currently selected (Figure1, 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 (Figure1, E). The procedure for using the application can be summarized as follows. First of all, the user must specify the medical images to be loaded. Once the images are loaded, the application generates a triangle soup from them. Then, models must be identified and defined using an area selector. Each time a model is defined, the user can set the models to which it is related and the application updates the hierarchy of models. Moreover, the unnecessary parts and the outliers can be removed by using the area selector. Defined models can be selected, translated and rotated to place it correctly in the scene. With the aim of providing assistance during the interaction, the application calculates the nearest points and the overlapping triangles. The definition of the hierarchy of models allow to only calculate collisions and distances between the models that have some type of relationship. To improve the interaction, the Leonar3Do input device can be used to manipulate the models.

The generation of 3D geometric models from medical images takes a few seconds. For instance, it takes approximately 20 seconds to perform a reconstruction from 200 medical images and 30 seconds from 400. The resolution of the CT images utilized in the test is 512x512 and the application is able to load up to 400 medical images. Once the geometric models have been generated, the interaction is performed in real time. To measure this, a Intel i7 2.80GHz processor, 4GB RAM and a NVidia GeForce GTS

240 have been used. 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 library (Blanchette and Summerfield, 2006). 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 (Schroeder et al., 2006) to reconstruct the 3D models and PQP (Larsen et al., 1999) to calculate a detailed collision detection. To program the Leonar3Do, the 1.0 version of the LeoAPI has been utilized.

3 MODEL RECONSTRUCTION

Model reconstruction from medical data is a difficult task due to the complexity of the human body structures. Once the medical images are obtained, the first step consists in segmenting the images in order to isolate the areas of interest. After that, the result obtained is utilized to reconstruct a 3D geometric model. The results obtained depend on the techniques used in each of the parts of the process.

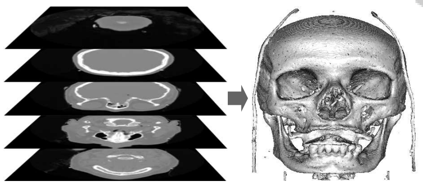
Figure 2: Iso-surface rendering from 3D medical images by using the Marching Cubes algorithm.

There are different sources such as Computed Tomography (CT), Magnetic Resonance Imaging (MRI) or Ultrasonic (US) techniques (Sachse, 2004) that provides 3D datasets and values that represent physycal quantities, e.g. proton densities in MRI, and attenuation coefficients in CT. In the context of modelling, these values can be used to extract features of a patient, such as bone, muscle or fat. These medical image sources can store the information in multiple formats. The Digital Imaging and Communications in Medicine (DICOM) standard is the most common format. Our application allows to visualize 2D images and reconstruct models from 3D medical image series and it has been tested with DICOM datasets and images from The Visible Human Project repository (NLM, 1986). To generate 3D data, the presented application makes use of the Marching Cubes algorithm [LC87] (see Figure 2). Marching Cubes is a fast and simple method that allows to automatically generate large polygonal datasets from volumetric data with high resolution. The outcome of the algorithm is a large soup of triangles without topology. However, the presented application is able to deal with surfaces that present holes by using the interaction techniques described in the next section. Before the reconstruction process, a threshold based method is utilized to segment the medical images in order to isolate the desired tissue. The threshold depends on the type of tissue and it is manually selected by the user. The segmentation result is a volume which is used as a input of the Marching Cubes algorithm.

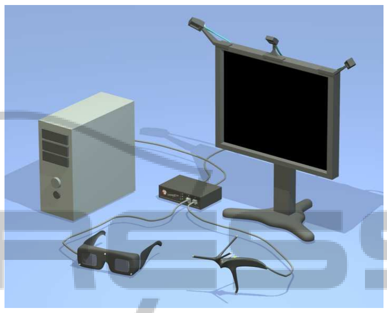
Figure 3: Virtual representation of the Leonar3Do (Leonar3Do, 2012).

4 INTERACTION

Once the 3D geometric model has been reconstructed, our application enables the interaction with it. With this aim, it has been integrated registration, collision detection, picking and multi-view methods. These methods allow the user to define 3D models from the reconstructed geometry and interact with each of the models independently. In addition, the detailed collision detection and the multi-view implementation provide visual aid to the user during the interaction. In order to improve the interaction, the Leonar3Do has been utilized. This is a virtual reality system that enables a stereo interaction. The Leonar3Do system mainly consist of a spatial input device, 3D glasses and monitor-mounted sensors (figure 3). Both the bird and the 3D glasses operate in six degrees of freedom and sensors can track both the bird and the glasses. To this end, the Leonar3Do system uses a technology based on infrared sensors. The bird has two buttons, one big and one small, which can be programmed. In addition, the big button is sensitive to pressure. On the other hand, 3D glasses enable stereoscopic vision. For that, Leonar3Do can use either an active and a passive system. While the passive system uses commonly polarizing lenses, the active system uses liquid crystal shutters which are powered through an USB port. In our case, the active system have been utilized.

4.1 Registration of the Models

Due to the fact that the reconstructed geometry is a triangle soup, 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 (Figure 4, bottom). When a model is defined by the area selector, the application allows to relate it to another previously defined model. Thus, it is possible to establish an hierarchy between the defined models. Furthermore, the area selector can also be used to remove triangles from the scene (Figure 4, top). 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.

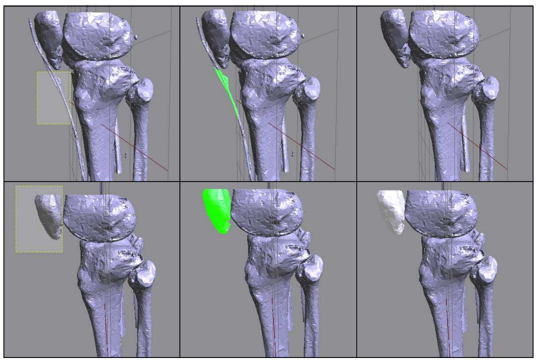
Figure 4: The area selector is used to remove unnecessary parts and to define models. Top, from left to right: selecting the part to remove; the selected part is displayed in green; Unnecessary parts have been removed from the scene. Bottom, from left to right: selecting the triangles that represent the patella; selected triangles are displayed in green; finally, the defined model is shown in white.

4.2 Collision Detection

In order to implement the interaction between the different models, we have not only to calculate the collision but also to provide visual aid to the user. Since the reconstructed models are triangle soups, it is necessary to use algorithms that can work with that type of models. These calculations are necessary to avoid that two models overlap or even to inform the user that the models are colliding (Figure 5, right). Furthermore, the distances and the nearest points between two models can help the user to place models that are not correctly positioned (Figure 5, left). The leonar3Do system has been used to improve the collision response. When a collision occurs, the spatial input device emits a small vibration. Some algorithms have been tested (Paulano et al., 2012) to choose the best suited to this problem. In that work, algorithms were tested with models with a complexity of up to one million triangles. The results shown that the PQP library (Larsen et al., 1999) 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.

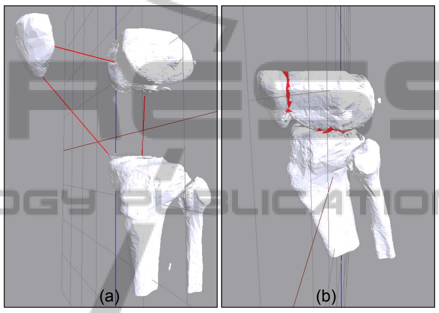

Figure 5: a) Calculation of the nearest points between some models. b) Overlapping triangles are displayed in red.

Table 1: Picking time using the ray picking method based on the PQP library. Run in a computer with Intel i7 2,80GHz, 4GB RAM, NVidia GeForce GTS 240.

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

4.3 3D Picking

After the models are defined, the application allows to select them by picking.To this end, the spatial input device is used. By moving this device, a 3D cursor in the scene is manipulated. This cursor enables the selection of previously defined models by using the small programmable button. If a model is selected, the spatial input device manipulates the selected model instead of the 3D cursor. Apart from this, each selected model can be rotated and translated by using the mouse. These transformations are performed independently from the rest of the models in the scene. In order to enable the selection of models, we decided to implement a method based on ray picking. The method consist of throwing a ray from the observer that pass through the cursor position and calculating the collision between the ray and all the objects in the scene. In order to check the collisions, we have used the PQP library. In this way, the data structures previously constructed are reused. To measure the efficiency of the method, some tests have been performed. As shown in table 1, 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.

4.4 Multi-canvas

Although the 3D view provides extra information to the user, doctors and radiologists usually work with a 2D view of the area of the patient. 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 top, side, and front view respectively. However, the four canvas can exchange their positions dynamically by clicking them or by pushing the buttons located at the top of the window. Instead of implementing multi-canvas using several glViewport (Shreiner, 2010), we decided to use 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.

4.5 Stereo Visualization and Immersion

The Leonar3Do system have been used to provide stereo visualization in the main canvas by using active glasses (see figure 6). This visualization system allows the user to better understand the scene and makes easy to select objects by using the spatial input device. In addition, the stereo visualization increases the feeling of immersion when using the application.

The Leonar3Do API (LeoAPI) is divided into two main parts: the tracking API and the stereoscopic API. The first one allows to know where both the spatial input device and the glasses are situated. As said in previous sections, the spatial input device has been used to manage a 3D cursor that enables the selection of models in the scene and to manipulate selected models. On the other hand, the position of the glasses is used to manage the camera, hence when the user move the head, the camera moves accordingly. To do this, both the translation and the rotation of the user head are considered. This increases the feeling of immersion and makes the application easy to use. It is important to consider that the camera position is modified by the LeoAPI, hence that it is placed at the focal point. The stereoscopic API enables the implementation of the stereo view and it is responsible for generating the final frame of the application, whether or not the stereo mode is activated. In our case, the stereo API is responsible for drawing the main canvas of the application. Because of this responsibility, it is necessary to disable the auto-buffer swapping because the LeoAPI already does it. Moreover, the LeoAPI fill a projection matrix which have to be provided to OpenGL before rendering the main canvas. However, this projection matrix can be modified before passing it to the OpenGL pipeline.

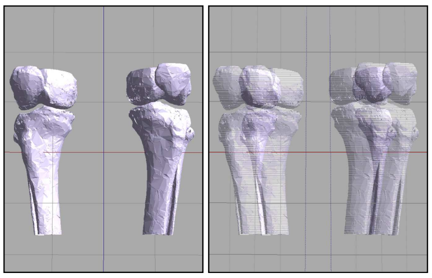
Figure 6: Screenshot of the main canvas of the application. Left, non-stereo mode. Right, stereo mode.

5 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:

- The presented application can be applied to prepare various types of surgical procedures. The surgeon could generate a 3D model from real patient data and simulate the interaction with the damaged area before the intervention.
- Measure the usability of the application. Experts will test the application and propose improvements to make the interaction more realistic.
- Obtain topologically correct meshes from triangle soups. These meshes would enable the use of more efficient algorithms to perform the interaction. In addition, different body parts could be labelled during the reconstruction step, avoiding the selection and registration of the models.

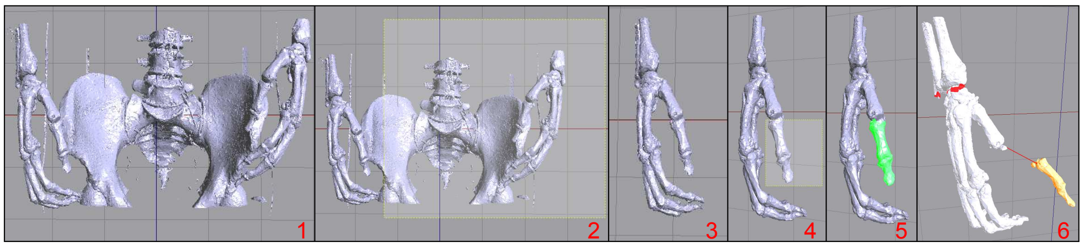

Figure 7: Example of using the application. 1 - Triangle soup generated from CT scans. 2 - Selection of some unnecessary triangles. 3 - Cleaned scene. 4 - Selection of a model. 5 - Selected model. 6 - Interaction with defined models.

- Incorporate a method to remove outliers automatically that avoids the user having to delete them manually. This would allow to obtain less complex models and improve the interaction.

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.

REFERENCES

Bhanirantka, P. and Demange, Y. (2002). OpenGL volumizer: a toolkit for high quality volume rendering of large data sets. In *Symposium on Volume Visualization and Graphics, 2002. Proceedings. IEEE / ACM SIGGRAPH*, pages 45–53. IEEE.

Blanchette, J. and Summerfield, M. (2006). *C++ GUI Programming with Qt 4*. Prentice Hall Open Source Software Development Series. Prentice Hall PTR.

Boubekeur, T., Heidrich, W., Granier, X., and Schlick, C. (2006). Volume-Surface Trees. *Computer Graphics Forum*, 25(3):399–406.

der Bergen, G. V. (2003). *Collision Detection in Interactive 3D Environments*. Elsevier.

Jiménez, J. J., Ogáyar, C. J., Segura, R. J., and Feito, F. R. (2006). Collision Detection between a Complex Solid and a Particle Cloud assisted by Programmable GPU. In *Vriphys: 3rd Workshop in Virtual Realitiy, Interactions, and Physical Simulation*, pages 43–52.

Larsen, E., Gottschalk, S., Lin, M., and Manocha, D. (1999). Fast proximity queries with swept sphere volumes. Technical report, Department of Computer Science, UNC Chapel Hill.

Leonar3Do (2012). Leonar3do. http://www.leonar3do.com. [Online; accessed 2-October-2013].

Lv, S., Yang, X., Gu, L., Xing, X., and Pan, L. (2009). Delaunay mesh reconstruction from 3D medical images based on centroidal voronoi tessellations. In *Computational Intelligence and Software Engineering, 2009. CiSE 2009. International Conference on*, pages 1–4.

NLM (1986). The visible human project. http://www.nlm.nih.gov/research/visible/visible_ human.html.

Paulano, F., Jiménez, J. J., Pulido, R., and Ogayar, C. J. (2012). A comparative study of implemented collision detection strategies. In *Proceedings of the International Conference on Computer Graphics Theory and Applications (GRAPP 2012)*, pages 485–490.

Pulido, R., Jiménez, J. J., and Paulano, F. (2012). Surface reconstruction from 3d medical images based on tritree contouring. In *Proceedings of the International Conference on Computer Graphics Theory and Applications (GRAPP 2012)*, pages 175–181.

Rautek, P., Bruckner, S., and Gröller, E. (2007). Semantic layers for illustrative volume rendering. *IEEE transactions on visualization and computer graphics*, 13(6):1336–1343.

Sachse, F. (2004). 5. digital image processing. In *Computational Cardiology*, volume 2966 of *Lecture Notes in Computer Science*, pages 91–118. Springer Berlin.

Schroeder, W., Martin, K. M., and Lorensen, W. E. (2006). *The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics*. Kitware, Inc.

Sharf, A., Lewiner, T., Shklarski, G., Toledo, S., and Cohen-Or, D. (2007). Interactive topology-aware surface reconstruction. *ACM Trans. on Graphics*, 26(3):43.

Shreiner, D. (2010). *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Versions 3.0 and 3.1*. OpenGL Series. Addison-Wesley.

Wolf, I., Vetter, M., Wegner, I., Nolden, M., Bottger, T., Hastenteufel, M., Schobinger, M., Kunert, T., and Meinzer, H.-P. (2004). The Medical Imaging Interaction Toolkit (MITK) a toolkit facilitating the creation of interactive software by extending VTK and ITK. In *Medical Imaging 2004*, pages 16–27. International Society for Optics and Photonics.

Yushkevich, P. A., Piven, J., Hazlett, H. C., Smith, R. G., Ho, S., Gee, J. C., and Gerig, G. (2006). User-guided 3D active contour segmentation of anatomical structures: Significantly improved efficiency and reliability. *NeuroImage*, 31(3):1116–1128.