Interactive Animations for Medical Data Visualization

David Gering*, GE Healthcare

1 Introduction

For many applications of 3-D medical data visualization, common graphics hardware is capable of performing surface rendering of a few segmented structures in true real-time. Volume rendering, which can generate photo-realistic images, can be performed in near real-time by making a trade-off in image quality. An example of this trade-off is the feature where medical images are rendered with reduced resolution while a user is rotating the viewpoint using the computer mouse or other input device. Then, when the user pauses his/her movement, the computer spends more time, such as a couple seconds, to render the scene at a higher resolution.

Naturally, the clinically desired solution would be a means of providing both true real-time interaction, and full photo-realistic image quality. An example of such a solution is the OpenGL preview mode that BMRT had [Apodaca and Gritz 2000]. A method is proposed that progresses toward the goal of providing both real-time interaction to an attending physician, and high image quality for archival storage.

2 Method

As a clinician interacts with a 3-D rendering, the navigation steps taken by the user are recorded as a script, so that an ultra high quality animation can be generated offline and stored. Then, when a reviewing physician views the stored animation, the user interface allows the animation to be interrupted for novel navigation of the 3-D scene in real-time (rendered with typical quality). Whenever the physician resumes playing the high quality stored animation, it begins not from the point of interruption, but from the point in the script that best matches the current view. This allows the most seamless transitions between a recorded animation and novel navigation. The workflow is:

1. Perform medical image analysis such as segmenting the image into key structures, and then forming surface models of each structure, as shown in Figure 1 [Slicer].
2. Navigate the OpenGL scene with real-time interaction.
3. Record a script that best describes the particular medical case under study.
4. Store the script, MRI, and surface models in a database.
5. Let a render farm generate a high-quality animation, and store it in the database.
6. View the stored animation, interrupting it at will to interact at a lower quality.
7. Resume playing the stored animation from the point in the script that best matches the interactive view.

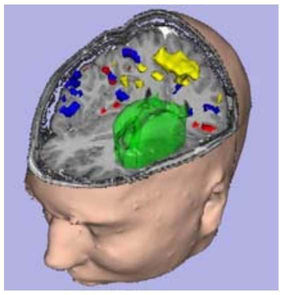

Figure 1: A rendering of the scene showing 3 orthogonal slices of the MRI volume, and 5 colored surface models of segmented structures, each with an independently variable opacity.

The format of the script is a list of records, one per frame of animation. Each record contains information describing the *world* and the *view*. The *world* information describes the aspects of the data that can be adjusted by the user in real-time via the user interface. This includes the opacity of each surface model, and the position of each reformatted slice plane. The *view* information describes the position and orientation of the virtual camera.

Once the script is recorded from the user's navigation, the script's data together with the static scene data (such as the surface models' triangle strips that do not change throughout the animation) are written as Renderman Interface Bytestream (RIB) files for offline rendering by a RenderMan-compatible renderer.

To replay the animation, its frame data, in addition to the MRI, surface models, and script, are loaded into memory. When the user interrupts the animation, the world and view information are retrieved from the script for the given frame number, and applied to the OpenGL scene, which is now rendered in place of the animation. The user then navigates (rotate, dolly, etc.) this scene.

To resume the animation, (indicated by a double-click of the mouse in our prototype), the current view and world information are matched with the script to find the frame with the best correspondence. A similarity score is computed for each parameter, and then a single score for the frame is computed by calculating a weighted average of these scores.

3 Conclusion

A method was presented that makes it convenient for a physician to record a script that can be used for offline rendering, and an interface for combining real-time interaction with a stored animation. Users can have both real-time interaction (to determine which navigation motions best explore the data for the case at hand), and the highest possible image quality for the stored animation (the level that can only be produced by a render farm). This combination offers a reviewing physician the choice of simply watching the high-quality animation unfold, or interrupting it to temporarily navigate the 3-D scene before continuing with the animation in the most seamless manner.

References

APODACA, A., GRITZ, L. 2000. *Advanced RenderMan*. Morgan Kaufman.

SLICER. http://www.slicer.org

* email: david.gering@med.ge.com