Haptic Volume Rendering with an Intermediate Local Representation

Olaf Körner[1], Markus Schill[1], Clemens Wagner[1], H.J. Bender[2], and Reiner Männer[1]

[1] Department for Computer Science V, University of Mannheim, B6,26, D-68131 Mannheim, Germany, http://www-mp.informatik.uni-mannheim.de
[2] Institut für Anästhesiologie und Operative Intensivmedizin, Fakultät für klinische Medizin Mannheim der Universität Heidelberg, D-68135 Mannheim, Germany
koerner@ti.uni-mannheim.de

Abstract. In medical applications volumetric data sets are widely used. Therefore a strong desire for haptic volume rendering exists in this community. Established force-feedback approaches are limited to surface-based data and cannot satisfy the needs of volume rendering. Particularly in the medical field, where physicians are used to examine objects with their sense of touch, a combination of volume visualization with haptics would be desireable. We present an approach where volumetric data is locally transformed into a surface representation. For our research, we use a PHANToM force-feedback device with the GHOST v2 library.
In our approach, the voxel data in a neighborhood of the current interaction point is used to generate isosurfaces through points with similar density values. The isosurfaces are then passed to the GHOST library and used as an intermediate representation between the volumetric data and a surface-based haptic rendering. As the surfaces are generated on the fly and no preprocessing is needed, it is possible to change transfer functions or the volume data itself during the haptic interaction.

Introduction

Haptic rendering is a powerful means to enhance the comprehensibility of digitally represented data. Especially in virtual reality applications the immersion effect is significantly enhanced by adding a haptic display. Currently, various applications are using haptic rendering, including 3D modeling [6], medicine [5] and entertainment [7]. The authors' research group ViPA[1] is investigating the use of virtual reality environments for training and rehearsal of medical surgeries [11], where haptic feedback can be essential. Mostly, haptic rendering is used to complete a system that also presents its data graphically; by touching an object without seeing it one can't but very roughly estimate its absolute size and position. In order to add haptic rendering this one has to consider the different nature of haptic and visual interaction:

[1] Virtual Patient Analysis

Visually, a scene can be perceived in its whole. Thus, the graphical rendering has to process the whole data set at a time, which results in huge computational requirements. However, compared to haptic rendering, a low update rate of 20 Hz is enough to give the eye a realistic impression.

The temporal resolution of the sense of touch is much higher. To provide a sufficient sensation of touch, a stiff object needs an update rate of at least 1000 Hz. On the other hand a haptic scenery is in general only perceived at certain interaction points, which allows a reduction of the haptic rendering to these few points.

With increasing computational power and advances in volume visualization techniques, volumetric data is becoming more and more manageable at the required interactive rate[2]. Especially in the medical field, where imaging modalities like CT and MRI produce large volume data sets, volume visualization is the method of choice. One of the main advantages over surface rendering is that in volume visualization we don't have to pinpoint a surface for the object-light interaction in advance. Instead the transfer function that defines how light and material interacts, can be changed during visualization. This is a highly desirable feature as it allows the investigation of a 3D-data set in an easy and comfortable way.

Unfortunately the situation is different in haptic rendering. Available haptic devices use the "haptic stylus" metaphor: the only point of haptic interaction is the tip of a pen that is used to scan the object. Objects are represented as surfaces whereby the location of interaction is defined. If the use of volume visualization is desired this generally results in the need to store two representations of the same data: surfaces for the haptics and volumes for the visualization. The interactive change of the transfer function and the herewith connected change of the interaction location is no longer possible.

Previous work

There are some approaches and also test implementations trying to face the need of a combination of volume graphics and haptics. An obvious approach for haptic volume rendering is to precalculate a surface model of the whole volume data and use this data set for the haptic rendering. Common shading algorithms known from computer graphics are used to achieve smooth objects (force shading)[3]. It is obvious that changing the data set must result in the regeneration of the surface model. In most cases this can not be done in real time.

An approach presented by Gibson [4] uses a second binary data set for haptics, that was extracted from the original MRI images by segmentation of the relevant structures. The binary data was then smoothed in a second pre-processing step. Such an approach is valid in all cases where changing the haptic interaction location is not required. Gibson simulates an arthroscopic knee surgery where the haptic interaction point is on the bone surface which does not change during simulation.

[2] E.g. `http://www.volumegraphics.com`

Volume data can also be generated synthetically in a way that surface normals and edge magnitudes can be calculated analytically. This speeds up the process of finding the correct interaction point enormously. Such synthetic data was used by Avila and Sobierajski in [2] to explore and sculpture voxel-based objects.

The Intermediate Local Surface Representation

In order to maintain the necessary update rates for haptics and visualization their loops have to run asynchroniously. *Intermediate representations* are commonly used to decouple the two loops [1]. The basic idea is that a suitable representation of the virtual environment is passed once (or at a feasible frequency) to the haptic loop. The haptic loop operates on this representation until it is changed by the main application loop. Intermediate representations depend on the problem [9] and should be chosen accordingly[3].

In our approach we take advantage of the local character of haptic perception. A local intermediate representation of the voxel data surrounding the tip of the stylus is generated on the fly, while moving the stylus in the virtual environment. We use the *marching cubes* algorithm to extract a surface from the voxel data [8]. It is not necessary that the surface generation itself runs with the high haptic rendering frequency as it is updated asynchroniously. Changing the transfer function of the visualization, results in a change of the intermediate representation since the surface is extracted from the voxel data after the transfer function has been applied.

The extracted surface is then passed to the haptics loop as an intermediate representation of the local volume data. Established surface rendering routines are applied for the haptic rendering of this surface (see Fig. 1).

Outline of the algorithm:

1. Determine the current stylus position.
2. Generate iso-surface from adjacent voxel data. Note that the same voxels are used as for the visualization.
3. Pass the iso-surface to the haptics loop:
   (a) Force-shading of the iso-surface.
   (b) Generate appropriate force.
4. go to top.

For most applications the generation of the isosurfaces could be performed fast enough. The *marching cubes* algorithm is running on a $7 \times 7 \times 7$ voxel environment of the stylus position. This size could be processed in time and supplied a large enough environment for the stylus movement. When the stylus is going to leave the local interaction area, a new surface has to be calculated.

[3] The GHOST SDK provides haptic primitives that can be used as intermediate representations. E.g. one can model a virtual environment with polygones (surfaces) and pass them to the PHANToM as a *gstTriPolyMesh* – without bothering about the 1000Hz calculation of the force vectors.

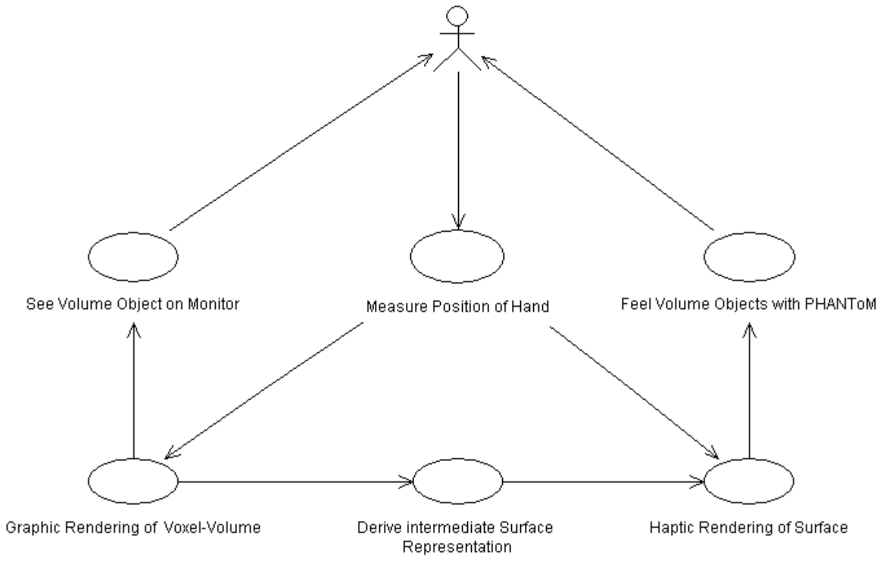
See Volume Object on Monitor
Measure Position of Hand
Feel Volume Objects with PHANToM
Graphic Rendering of Voxel-Volume
Derive intermediate Surface Representation
Haptic Rendering of Surface

Fig. 1. Flow of information between haptic and graphic

Handling fast stylus movements:

In cases of fast movements the stylus reaches the edge of the processed voxel volume before a new surface was generated. Then, no intermediate representation is present and the stylus can move freely. To avoid the resulting unrealistic behavior we move the object itself along with the stylus: When the rim of the known environment is reached, the whole object is fixed at the tip of the stylus and moved along, so that the relative positions of stylus and object does not change. The intermediate representation stays valid until a new one is generated. Two circumstances help us "cheating" the user's perception: (1) The operation is done on the intermediate representation. Thus it is not displayed by the visualization. (2) Without additional information our sense of touch estimates size and position no more accurate than ± 1 cm [12]. Together, these facts tempt our brain to trust the visual information and to ignore small displacements of the object.

Implementation

The implementation of the intermediate surface representation is solved straightforward with the GHOST library, that provides a class *gstTriPolyMesh* for the definition of an arbitrary surface. This class acts as an attribute for the class *gstTriPolyMeshHaptic*, in which parameters like friction, damping, smoothing and the spring constant of the surface can be defined for the haptic rendering.

Remarks on the implementation of the algorithm:

1. **Determine the current stylus position.** The ideal haptic interface point[4] is used for the programs' calculations and for the visualization.
2. **Generate the isosurface from adjacent voxel data.** Decoupled from the graphics loop and the haptics loop, the generation of the isosurfaces runs in its own thread. The adjacent voxel data of a size of $7 \times 7 \times 7$ is investigated by the *marching cubes* algorithm for an iso-surface of a specific grey-value. The resulting surface is made up of triangles.
3. **Pass the isosurface as *gstTriPolyMesh* to the GHOST.** The derived iso-surface is used to build a new *gstPolyMesh*-object that is added to the haptic scene. The haptic rendering is provided by the GHOST.
4. **go to top.**

Shifting the ideal haptic interface point for fast stylus movements:

In order to prevent the *ideal haptic interface point* from leaving the area of the already generated surfaces, we have to add the discussed object position adjustment to the haptic loop itself. GHOST provides a class *gstEffect* that is commonly used for additional force vectors. A virtual method *calcEffectForce()* is provided, that has to be implemented in a derived class and will be called in every pass of the haptic loop. Instead of adding forces, we exploit this method for shifting the scenery in the haptic loop.

The actual position of the *ideal haptic interface point* is compared with its position at the last update of the iso-surface. When the *ideal haptic interface point* is about to leave the known area, its position is shifted along the difference vector between the actual and the old position.

As far as we have experienced, these shiftings are not perceivable, as long as they do not exceed a distance of 1 cm.

Results

We developed a new algorithm for the haptic exploration of volumetric datasets. It provided a realistic tactile sensation of our datasets. Nonetheless strong discontinuities were still noticeable. We aim to use grey-value information to adjust the voxels' position with subvoxel accuracy.

For overcoming time delays we suggested to shift the *ideal haptic interface point*. In standard conditions, this was not sensed by the user. However, very fast movements or high resolution datasets may lead to the perception that the whole haptic scene has been moved.

Unfortunately, the implementation of the destructor of *gstTriPolyMesh* in the GHOST v2 seems to fail deallocating the used memory. Due to the extensive use of this class, our system goes out of memory after a few minutes.

[4] If rendering a surface object, the *haptic interface point* can penetrate a stiff object while the *ideal haptic interface point* remains on its surface. With the difference between the two points, a backdriving force is calculated [13].

Hard- and Software

Used Hardware:

- Pentium PC with two CPUs running at 200 Mhz.
- 128 MB memory.
- standard graphics adapter.
- PHANToM model 1.0 (see `http://www.sensable.com`)

Used Software:

- Windows NT 4.0
- Visual C++ 5.0
- GHOST v2 (see `http://www.sensable.com`)
- VGL 2.0 (see `http://www.volumegraphics.com`)

References

1. Adachi, Y.: Touch and Trace on the Free Form Surface of Virtual Object. IEEE Virtual Reality Annual International Symposium. VRAIS. (1993) 162–168
2. Avila, R.S, Sobierajski, L.M.: A Haptic Interaction Method for Volume Visualization. Proc. Visualization'96. (1996) 197–204
3. Basdogan, C., Ho, C.-H., Srinivasan, M.A.: A ray-based haptic rendering technique for displaying shape and texture of 3D objects in virtual environments. Proc. of the ASME Winter Annual Meeting. (1997)
4. Gibson, S.F.F.: Beyond Volume Rendering: Visualization, Haptic Exploration and Physical Modeling of Voxel-based Objects. Mitsubishi Electric Research Laboratories (MERL). Tech. Rep. 95-04. (1995)
5. Gibson, S.F.F, et al: Volumetric Object Modeling for Surgical Simulation. Medical Image Analysis. **1** (1998) 121–132
6. Gutiérrez, T., et al: Assembly simulation through haptic virtual prototypes. PUG98. MIT. (1998)
7. Immerse Inc.: `http://www.immerse.com`
8. Lorensen, W.E., Cline, H.E.: Marching Cubes: A high resolution 3d surface construction algorithm. Proc. of SIGGRAPH, **21** 4 (1987) 163–169
9. Mark, W.R., et al: Adding Force Feedback to Graphics Systems: Issues and Solutions. Proc. of SIGGRAPH 96 Annual Conf. Series (1996) 447–452
10. Massie, T., Salisbury, J.K.: The PHANToM Haptic Interface: A Device for Probing Virtual Objects. Proc. of the ASME Winter Annual Meeting, Symp. on Haptic Interfaces for Virtual Env. and Teleoperator Systems. **1** (1994) 295–302
11. Schill, M., Gibson, S.F.F., Bender, H.-J., Männer, R.: Biomechanical Simulation of the Vitreous Humor in the Eye Using an Enhanced ChainMail Algorithm. Proc. MICCAI98. (1998)
12. Srinivasan, M.A., Basdogan, C.: Haptics in virtual environments: Taxonomy, research status and challenges. Computer Graphics, **21** 4 (1997)
13. Zilles, C.B., Salisbury, J.K.: A constraint-based god-object method for haptic display. Proc. of the ASME Winter Annual Meeting. Symp. on Haptic Interfaces for Virtual Environments and Teleoperator Systems. **1** (1994) 146–150