Erasing, Digging and Clipping in Volumetric Datasets with One or Two Hands

Rafael Huff\textsuperscript{1}, Carlos A. Dietrich\textsuperscript{1}, Luciana P. Nedel\textsuperscript{1}, Carla M.D.S. Freitas\textsuperscript{1}, João L.D. Comba\textsuperscript{1}, Silvia D. Olabarriaga\textsuperscript{2}

\textsuperscript{1}Instituto de Informática - Universidade Federal do Rio Grande do Sul\textsuperscript{*}
\textsuperscript{2}Informatics Institute - University of Amsterdam\textsuperscript{†}

\textbf{Abstract}

Visualization of volumetric datasets is common in many fields and has been an active area of research in the past two decades. In spite of developments in volume visualization techniques, interacting with large datasets still demands research efforts due to perceptual and performance issues. The support of graphics hardware for texture-based visualization allows efficient implementation of rendering techniques that can be combined with interactive sculpting tools to enable interactive inspection of 3D datasets. In this paper we report the development of three 3D interactive tools, eraser, digger and clipper, which specify regions within the volume to be discarded from rendering. Sculpting is accomplished by running special fragment programs that discard fragments based on geometric predicates. The interaction techniques we proposed were implemented using the virtual hand metaphor. The tools were evaluated by comparing the use of a 3D mouse against a conventional wheel-mouse for guiding volume and tools manipulation. Two-handed input was tested with both types of mouse and the results obtained indicate a preference for a combination of 2D and 3D mouse.

\textbf{CR Categories:} I.3.6 [Methodology and Techniques]: Interaction techniques; I.3.7 [Three-Dimensional Graphics and Realism]: Virtual reality

\textbf{Keywords:} 3D interaction, interactive volume sculpting, real-time volume rendering

1 Introduction

The visualization of 3D datasets, or volumes, is a common task in several fields, such as medical imaging, weather analysis and oil prospecting. For example, MRI, CT and ultrasound images are widely used to support diagnosis, surgery planning, and treatment follow-up. Despite the evolution of volume rendering techniques, 2D slice-by-slice inspection is still widely used in practice due to several reasons. First, datasets can be very large and rendering algorithms might not be fast enough for real-time response. Also, inspection of inner structures in the volume requires a segmentation step, which is a non-trivial task for medical images, for instance. And finally, nearly everybody is familiar with 2D interaction, which can be easily accomplished with a standard mouse and keyboard available in any workstation.

Interactive systems that enable visualization and interaction with data volumes in fully immersive 3D experiences have recently become a reality. In such systems, the visualization is provided in projection-based VR displays and the interaction is almost always natural and intuitive, making use of special devices such as data glove and 3D mouse. The argument that holds for this kind of systems is simple: if the problem is 3D, it seems obvious to implement a complete 3D solution, assuring a good mapping between reality and virtuality. On the down side, such systems are still expensive and far from the user’s reality.

The goal of our work is to develop 3D interactive techniques for real-time volume sculpting and visualization in commodity PC workstations. Although the techniques are generic, they are motivated by medical applications, where volume sculpting often plays an important role in the selection of a volume of interest (VOI) for...
the visualization. Real-time response has been achieved by running the sculpting tools and volume rendering on the GPU.

An earlier work [Dietrich et al. 2004], presents the implementation of GPU-based sculpting tools based on a 2D interaction paradigm, using a conventional mouse. The present paper describes three sculpting tools that fully operate in 3D: eraser, digger and clipper (Figure 1.b-d). The tools work in image space (on the GPU) and enable the volume to be clipped by a volumetric region defined by a cylinder, a sphere or a plane. The tools and the VOI are manipulated with the virtual hand metaphor [Bowman and Hodges 1999] to enlarge the continuity between visualization and interaction, even if non-immersively. The virtual hand is controlled with the 3D mouse, using one- or two-handed interaction. An initial evaluation study was performed to compare the usage of a 3D mouse against a conventional 2D mouse, as well as single against two-handed interaction.

The remainder of the paper is structured as follows. In Section 2 we review relevant work in the areas of GPU-based volume rendering, volume sculpting and 3D interaction. An overview of the system is presented in Section 3, while the tools and the interaction are detailed in Sections 4 and 5 respectively. Section 6 describes the experimental design, presenting the scenario, evaluation criteria and parameters, as well as the procedure followed for the tests. The experiment results and discussion are presented Section 7. Section 8 presents conclusions and discusses future work.

2 Related Work

Although volumetric visualization techniques have been proposed since the 80’s, only recently the advances in commodity graphics hardware allowed texture-based volume rendering to achieve satisfying image quality with better performance than software-based methods [Rezk-Salama et al. 2000; Kniss et al. 2001].

Interacting with volumes to expose inner structures through clipping tools has been practically restricted to the use of clipping planes. Only recently clipping approaches beyond standard clipping planes have been considered in scientific visualization [Weiskopf et al. 2003; Manssour et al. 2005].

Van Gelder and Kim [Gelder and Kim 1996] use clipping planes to specify the boundaries of the clipping region in texture-based volume rendering. Pfister [Pfister et al. 1999] proposes the Volume-Pro hardware architecture with cropping and cutting planes features for clipping the volume. Westermann and Ertl [Westermann and Ertl 1998] introduce the concept of clipping geometries by means of stencil buffer operations. In their approach, the clipping region has to be rendered for each slice to set the stencil buffer correctly. Regions where the clipping region intersects the current slice are marked in the stencil buffer. Stencil test is used to discard marked fragments.

Weiskopf et al. [Weiskopf et al. 2002] proposes clipping tests exploiting per-fragment operations on the graphics hardware to maintain high rendering speed. They propose a depth-based clipping technique that analyzes the depth structure of the clipping geometry to decide which parts of the volume have to be clipped (a similar approach is used by Diepstraten et al. [Diepstraten et al. 2002]). They also propose a clipping tool that uses a voxelized clipping object to identify clipped regions. In this tool a texture buffer stores visibility information, and it is used as a mask that indicates if each voxel will contribute to the final image. This clipping texture is a binary volume obtained by the voxelized geometry of clipping tools on the CPU. Scale, rotation and translation operations of the clip object are allowed by adapting the texture coordinates of the clipping texture. The “voxelized clipping object” paradigm is also exploited in our work but the geometry of our tools are voxelized on the GPU.

In [Bruckner and Gröller 2005] a volume illustration system called VolumeShop that uses sculpting tools to produce volume illustrations was proposed. The similarity of their work to ours lies in their cutaway and ghost tools, which use opacity assignments to remove occluding regions and allow inspection of inner features. It differs from our work in the selection of the regions within the volume to be eliminated. We use the virtual hand metaphor and they cast a ray from a point clicked on a screen into the volume. At the first voxel crossed by this ray, a volumetric brush centered on the voxel can either add or erase details into the selection volume for every voxel within the brush bounding box.

Different input devices have been used in sculpting applications, such as conventional 2D mouse used by Wang et al. [Wang and Kaufman 1995], 3D mouse used by Galyan [Galyean and Hughes 1991], and even both at the same time used by Parviainen et al. [Parviainen et al. 2004].

Two-handed input has often been viewed as a technique to improve the efficiency of human-computer interaction, by enabling the user to perform two sub-tasks in parallel [Buxton and Meyers 1986], and transferring everyday skills for manipulating tools with two hands to the operation of a computer, with little or no training [Kabbash et al. 1994]. Two-handed interaction can also help users to better perceive a 3D environment because it takes advantage of people’s innate ability of knowing precisely where their hands are relative to each other [Hinckley et al.; Sachs et al. 1991]. Leganchuk et al. [Leganchuk et al. 1998] also noticed that using both hands in an application brings cognitive and manual advantages and their work provided a complimentary perspective, exploring the potential benefits into everyday applications.

3 System Overview

The sculpting tools available in our system are integrated into a texture-based volume rendering pipeline. Following the notation described in [Ikits et al. 2004], we divided this pipeline into three stages: initialization, update and drawing (Figure 2).

In the initialization stage the dataset is processed and stored as a 3D texture (called VolumeTex) on the GPU. During this process, if required, a pre-classification according to a transfer function [Engel et al. ] or calculations of scalar values and gradients for volume shading [Weiskopf et al. 2003] can be used. These, however, are not discussed in this work.

Updates are requested when the user interacts with the system, either changing viewing parameters or sculpting the volume. Viewing changes are issued at camera manipulation, but sculpting can be produced by any tool. Tools are implemented by fragment shaders that run on the GPU and are used to discard fragments from contributing to the final image.

Every tool follows the paradigm proposed by Weiskopf et al. [Weiskopf et al. 2002]. Like in their work, we also store a binary visibility mask in a 3D texture (SculptMap, in Figure 2), but the geometry of our tools are voxelized on the GPU. In order to achieve that, our tools are associated to certain primitive geometries that are generated by fragment programs on the GPU. Among all possible clip objects that can be created by few parameters sent to fragment programs, we chose those that are better suited for medical volume sculpting and could simulate complex geometries through several interactions. They correspond to a plane for the clipper tool,
Figure 2: System Architecture. First, an updating phase is used to define a selection volume (sculpt map) based on the current sculpting tool. The final phase corresponds to a drawing step that generates the final visualization while discarding regions within the sculpt map.

a sphere for the digger tool, and a cylinder for the eraser tool, all controlled by a virtual hand.

By voxelizing the clipping tools on the GPU, it is possible to change the active clipping tool interactively. When this is done on the CPU, the tools can hardly be voxelized in real time. Therefore, every tool should be voxelized during the initialization and should be stored in different texture buffers. This is unfeasible when the application works with large volume datasets, as in medical applications.

When a sculpting tool is applied to the volume, the voxels that lie within the geometry associated to each tool are marked to be removed during the drawing stage (see Updating phase in Figure 2). To accomplish this, the alpha values of the equivalent SculptMap voxels are used as visibility flags. The voxels to be eliminated have alpha value set to 0, while voxels to keep have value set to 1. This is done by the fragment shaders FSEraser, FSDigger and FSClipper, which are detailed in Section 4. Following Weiskopf, we have also modified the approach shown here to avoid jaggy artifacts similar to those in aliased line drawings. This can be accomplished by storing Euclidean distance values to the closest point on the clip object and performing a trilinear interpolation during rendering.

The visualization of the dataset is achieved by sampling the 3D texture by a set of planes (proxy geometry) aligned with the viewing direction (see Drawing phase, in Figure 2). During the rendering stage, these planes are supposed to be drawn in sorted order. However, before these polygons are rendered, the rendering parameters should be set up. In our system, the combining step is performed in a fragment shader called FSCombiner (Figure 2). The shader input parameters are two 3D textures - the original volume represented in Figure 2 diagram as VolumeTex, and the visibility information saved in SculptMap.

The final image in the framebuffer is produced by combining the VolumeTex with the SculptMap. As the SculptMap dimension is frequently smaller than the VolumeTex, trilinear interpolation of the SculptMap should be considered during the combining process.

4 Distance-based Volume Sculpting

Our sculpting system is composed of three different tools. The eraser and digger are special tools that allow the elimination of voxels selected with a “painting” paradigm. Selection is achieved by only considering voxels that are inside an imaginary volume. This volume is calculated by a certain distance of the voxels to a line or a point, for the eraser and digger tools respectively. This distance is the radius of the tools specified by the user. For the clipper tool, the volume to be eliminated is one of the sides of the original volume divided by a plane. Tool location in the scene corresponds to the location of the virtual hand.

4.1 Eraser Tool

The eraser is a tool that eliminates the voxels inside a virtual cylinder crossing the entire volume. This cylinder can be generated by an infinite line and a circular region around this line. In our work, we calculate this region by using a projection plane, a 2D point P on this plane and the radius r of the tool (Figure 3). The selection of a voxel for elimination is done by first projecting its center into the projection plane. For each projected center, a distance to P is computed (called projected distance d of a voxel). Note that d is calculated on the projection plane, and corresponds to a 2D Euclidean distance. Every voxel with d < r is removed.

Figure 3: Eraser tool based on projection of voxels onto a plane

The fragment shader that implements the eraser tool is called FSEraser, which receives as input the SculptMap, the projection plane, the 2D point P, and the tool radius r (see Figure 2). For each fragment in this proxy rectangle, the steps shown in Figure 4 are performed.
4.2 Digger Tool

The digger is a tool that eliminates the voxels inside a virtual sphere. This region is generated by a 3D point $P$ and radius $r$ specified by the user (Figure 5). The selection of a voxel for elimination is done by first calculating the 3D Euclidean distance $d$ of its center to $P$. If this distance is smaller than $r$, this voxel is removed.

The fragment shader that implements the digger tool is called FS-Digger, and receives as input the SculptMap, the point $P$ and the radius $r$ (see Figure 2). Cg code for FS-Digger is shown in Figure 6.

4.3 Clipper Tool

The clipper tool defines a convex region that lies inside the 3D original volume. The specification of this convex region by the user follows a carving approach. Starting with a basic convex shape (a cube, for example), the user can define a cutting plane that slices the current convex volume, leading to a convex shape with one additional face. The process is repeated until necessary. The selection of a voxel for elimination is done by calculating the smaller distance of its center to the plane (Figure 7). This distance (signed) is calculated by a dot product between voxel coordinates and the plane coefficients. The voxel is removed if the distance is negative.

FSClipper is the fragment shader that implements the clipper tool. It receives as input the SculptMap and the cutting plane (see Figure 2). Cg code for FSClipper is shown in Figure 8.

5 Volume and Tools Manipulation

Our main goal was to enlarge the continuity between visualization and interaction by adopting a 3D interaction technique to control the tools. This technique used the virtual hand metaphor and a 3D input device which does not force the user to decompose a 3D task into a series of 2D or 1D tasks [Liang and Green 1993].

We used a Magellan Space mouse (Figure 9.a), because it has 6 degrees of freedom (DOFs) and it is relatively simple and easy to learn. However, for evaluation purposes, we also implemented the virtual hand metaphor controlled by a conventional mouse (Figure 9.b), since it is the most commonly used input device.

In our application two different objects can be manipulated by the input devices: the tools and the VOI. The tools can be rotated around themselves and translated in $x$, $y$, and $z$ axis. The VOI can also be rotated around itself but cannot be translated.
The 3D mouse can easily control 3D rotations and translations on a virtual hand interaction, because it has 6 DOFs, but these transformations are not trivially executed by a conventional 2D mouse. In order to compare both devices, we had to compensate the lack of DOFs of the 2D mouse. Translations along the $x$ and $y$ axis were normally performed by moving the 2D mouse, but translations along the missing $z$ dimension were performed by rolling the wheel button of the mouse. Rotations were compensated by holding down a button and moving the mouse around (dragging movement).

The user could select in the interface of the application which object (tools or VOI) he/she wanted to manipulate with each mouse (2D or 3D). The manipulation of both objects at the same time was not allowed, e.g. rotating the volume and erasing it at the same time. In this case, the user could perform the rotation first and erase afterwards, or vice-versa, switching the object in the interface controlled by the device. In order to manipulate both objects at the same time, two-handed interaction was enabled by using the 2D and 3D mouse at the same time. In this case, the user could choose which device would be associated with each object (tools and VOI). Due to implementation difficulties, two 3D mice or two 2D mice cannot be used at the same time.

6 Evaluation

In order to evaluate the performance and preference of users while interacting with volumetric datasets using our sculpting tools, we designed an experiment that allowed manipulating and carving volumes with the developed tools (eraser, digger and clipper) employing different combination of input devices. This section describes the experiment design and results.

6.1 Hypotheses

Our evaluation is based on three hypotheses, concerning to the combination of input devices used:

- **H1.** The manipulation of the tools is faster using a conventional mouse.
- **H2.** The error rate decreases when the subjects adopt the 3D device to manipulate the tool.
- **H3.** The subjects will prefer two-handed input, using both device simultaneously to manipulate volume and tools.

The first hypothesis arises from the thought that normal users are used to the conventional mouse and a training stage with the 3D mouse would not provide them the same skills. The second hypothesis, on the other hand, is based on the statement that 3D devices are better suited for 3D interaction techniques [Liang and Green 1993]. Finally, the last hypothesis relies on the cognitive benefits that reduce the load of mentally composing and visualizing the tasks at an unnaturally low level which is imposed by traditional unimanual techniques [Buxton and Meyers 1986].

6.2 Scenario

The experiment consisted of the elimination of red voxels in a red-and-white volume.

The testbed application takes as input different datasets, each one being a set of images with red and white pixels that describe a volume. We designed 3 different datasets, one for each tool, because of the geometry associated with each tool. A fourth volume was used to train the users before they start the evaluation process (Figure 10).

The user has to remove the maximum number as possible of red voxels from the volume, while avoiding the elimination of the white ones as well.

To test the three hypotheses we defined 12 user tasks consisting of the basic task applying a different combination of the input devices. The combination of 3 sculpting tools (digger, eraser and clipper), 2 different objects (tools and volume) and 2 input devices (2D and 3D mouse) results in a set of 12 different tasks.

6.3 Independent and Dependent Variables

Independent variables are the experiment variables that generate different conditions to be compared. As independent variables we used: age; gender; user occupation; previous use of 2D and 3D devices; previous use of 2D and 3D applications, like games, CAD, 3D Studio, Blender, etc., and finally the 12 tasks.

Dependent variables are measured taken during the execution of tasks. They can be objective, as the time spent to accomplish a task, or subjective, like the level of satisfaction, collected from post-test questionnaires answered by the subjects. The dependent variables used in our experiment were: task completion time; error rate and device preference.
Figure 10: Experiment datasets: (a) eraser evaluation volume, (b) digger evaluation volume, (c) clipper evaluation volume and (d) training evaluation volume.

The completion time was measured from the beginning of each task until the user finished it by pressing a button. The error rate (Equation 1) took into account the error after the sculpting task is completed and time spent to perform the task, as given by Equation 1:

\[ \text{error rate} = \frac{\varepsilon}{T} \]  

(1)

where \( T \) is completion time and \( \varepsilon \) is given by Equation 2:

\[ \varepsilon = R_e + W_e \]  

(2)

with \( R_e \) being the number of remaining red voxels and \( W_e \) the number of deleted white voxels.

Finally, ease of use and device preference were verified through the analysis of questionnaires.

6.4 Subjects

We have performed tests with 15 subjects (1 professor and 14 students), most of them with a Computer Science major. Even though only two of the students were not working in computer graphics, most of them had no or few experience with 3D mouse. Our population was heterogeneous, consisting of 3 women and 12 men with ages between 20 and 40, average 25 years old. Each of them tested the 3 tools (Section 4) and the 2 devices (Figure 9), performing the 12 tasks.

6.5 Methodology and Procedure

We performed our tests in a Windows XP platform on a standard PC (AGP 4) with a single 3.2 GHz Intel Pentium IV with 512 Mb of RAM, and an ATI Mobility Radeon X600 graphic card with 256 MB.

The following steps were applied in the same order to all subjects:

- Instructions about the application and devices were provided;
- A pre-test questionnaire was completed by the subject;
- Unlimited time for training was provided;
- The 12 tasks were performed randomly;
- The log file was recorded;
- A post-test questionnaire was completed by the subject.

Before being asked to perform the tasks, each user had unlimited time to learn and become familiarized with the application and devices. For this the users had a training task (Figure 10-d) to practice

and only during this stage they could freely switch devices. Once the user felt comfortable, the 12 tasks were performed.

Tasks were selected in random order; after the display of a task specification, the user had to press a start button to actually begin the task. The task did not finish automatically, but when the user felt that he had accomplished it, he/she pressed a button. Then, the new task was sorted by the application and displayed, until all the 12 tasks were accomplished.

The log file generated for each user contained the following data items associated with each task: description of the task, total number of white and red voxels removed from the volume, remaining number of white and red voxels and time spent during the task.

7 Results and Discussion

For the statistical analysis of the collected data, several variables were considered. These variables were obtained through the pre- and post-test questionnaires and the log files. The results are discussed below.

7.1 Comparing Previous Experience with Performance

The correlation between the data of the pre-test questionnaire and the log files (independent and dependent variables) was calculated.

<table>
<thead>
<tr>
<th>Subject Previous Experience</th>
<th>Excellent</th>
<th>Good</th>
<th>Poor</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>3D Applications</td>
<td>40%</td>
<td>20%</td>
<td>26.66%</td>
<td>13.33%</td>
</tr>
<tr>
<td>2D Games</td>
<td>73.33%</td>
<td>20%</td>
<td>6.66%</td>
<td>0%</td>
</tr>
<tr>
<td>3D Games</td>
<td>60%</td>
<td>20%</td>
<td>20%</td>
<td>0%</td>
</tr>
<tr>
<td>Mouse 2D</td>
<td>100%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>Mouse 3D</td>
<td>0%</td>
<td>6.66%</td>
<td>66.66%</td>
<td>26.66%</td>
</tr>
<tr>
<td>Keyboard</td>
<td>100%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>HMD</td>
<td>6.66%</td>
<td>53.33%</td>
<td>20%</td>
<td>20%</td>
</tr>
<tr>
<td>Data Glove</td>
<td>13.33%</td>
<td>40%</td>
<td>26.66%</td>
<td>20%</td>
</tr>
<tr>
<td>FOB</td>
<td>13.33%</td>
<td>26.66%</td>
<td>20%</td>
<td>40%</td>
</tr>
</tbody>
</table>

Table 1: Subject previous experiences with 2D and 3D applications and devices.
It was possible to verify that performance with the digger and clipper tasks was correlated to the previous experiences of the subjects with 3D devices, especially when the user had to use the 2D mouse to manipulate the tools and the 3D mouse to control the volume ($r > 0.514, p < 0.05$).

Other variables, like age and gender, did not show any correlation.

### 7.2 Testing the Hypothesis

We used ANOVA (Analysis of Variance) to verify the significance of the time and error results obtained from the log files.

To evaluate the first hypothesis (The manipulation of the tools is faster using a conventional mouse) the time spent by the users to complete each task was analyzed. The ANOVA test showed that the difference of the time spent in each task was significant only for the digger ($F = 5.308, p < 0.02$) and eraser ($F = 4.663, p < 0.04$) tools. When setting the 2D mouse for controlling volume manipulation, the mean time using the 3D mouse for guiding the tool is higher than the time spent using the 2D mouse for that, both for the digger and the eraser (Figures 11 and 12).

![Figure 11: Digger mean times (in seconds) when controlling the volume with the 2D mouse.](image)

![Figure 12: Eraser mean times (in seconds) when controlling the volume with the 2D mouse.](image)

In order to verify the second hypothesis (The error rate decreases when the users adopt the 3D device to manipulate the tool), the error rate was calculated dividing user errors by time spent in each task. Error is the number of remaining red voxels and deleted white voxels (recalling that the task was only to remove red voxels). The difference of performance was not significant according to the ANOVA test except when comparing the different devices to manipulate the digger tool and using the 2D mouse to control the volume ($F = 8.239, p < 0.008$). Only in this case, the second hypothesis is confirmed because the mean error rate when using the 3D mouse to control the tool is much smaller than when using the 2D mouse (Figure 13). This can be due to the two handed input in this case, which avoids the need to switch the 2D mouse from controlling the volume to control the tool.

![Figure 13: Digger error rate when controlling the volume with the 2D mouse.](image)

The analysis of the post-test questionnaire confirmed the third hypothesis (The users will prefer two-handed input, using both device simultaneously to manipulate volume and tools). Table 2 shows that the great majority of the users preferred the 2D mouse to manipulate the tools and the 3D mouse to control the volume separately. Table 3 shows that the users also preferred this combination of devices for two-handed input.

<table>
<thead>
<tr>
<th></th>
<th>2D Mouse</th>
<th>3D Mouse</th>
</tr>
</thead>
<tbody>
<tr>
<td>Volume</td>
<td>26.67%</td>
<td>73.33%</td>
</tr>
<tr>
<td>Digger</td>
<td>66.67%</td>
<td>33.33%</td>
</tr>
<tr>
<td>Eraser</td>
<td>73.33%</td>
<td>26.67%</td>
</tr>
<tr>
<td>Clipper</td>
<td>53.33%</td>
<td>46.67%</td>
</tr>
</tbody>
</table>

Table 2: Preferred device to control the volume or tools

<table>
<thead>
<tr>
<th></th>
<th>Preference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Volume (2D Mouse) and Tools (2D Mouse)</td>
<td>0%</td>
</tr>
<tr>
<td>Volume (2D Mouse) and Tools (3D Mouse)</td>
<td>26.66%</td>
</tr>
<tr>
<td>Volume (3D Mouse) and Tools (2D Mouse)</td>
<td>66.66%</td>
</tr>
<tr>
<td>Volume (3D Mouse) and Tools (3D Mouse)</td>
<td>6.66%</td>
</tr>
</tbody>
</table>

Table 3: Preferred combination of devices

### 8 Conclusions

We presented three fully 3D interactive tools to sculpt volumetric datasets. The tools are based on the multi-texturing facilities provided by GPUs. The eraser, digger and clipper are implemented in special fragment programs that discard fragments based on visibility information.

An experiment with users allowed us to evaluate the use of different devices to interact with the application for manipulating the sculpting tools and controlling the dataset position and orientation. Analysis of the results showed that in the few cases where the difference of performance was significant, the simultaneous use of two different devices (2D and 3D mouse) presented a better performance. The preference of the users, obtained from questionnaires, was also the use of two-handed input. The collected data shows that better results are obtained when the interaction in the real and virtual worlds is direct. More images and
some videos illustrating the use of the tools are available on the web (http://www.inf.ufrgs.br/cg/liver3d).

We are currently designing new experiments using medical images as datasets and physicians as subjects. The set of tasks will be directed to hepatectomy surgery planning, which has been our motivation problem all along [Zanchet et al. 2005].

Acknowledgments

The authors would like to thank CNPq (Brazilian Council for Research and Development) projects number 478721/2003-0, 306099/2004-7 and 540414/01-8, and the Brazilian Education Ministry (MEC-CAPES) for financial support. They are also grateful to the colleagues that participated as subjects in the experiments.

References


Engel, K., Kraus, M., and Ertl, T. High-quality pre-integrated volume rendering using hardware-accelerated pixel shading.


Hinckley, K., Pausch, R. F., Goble, J. C., and Kassel, N. F. A survey of design issues in spatial input.


