Chapter 7

Incision: A hands-on system for manipulation of volumetric data sets

7.1 Introduction

Volumetric data present unique challenges in terms of visualization and interaction. Correa in his paper, "Visualizing what lies inside", illistrates the importance of visaulization for internal features [Cor09]. As opposed to tackling the challenges of internal visualization simply through visual representations, a system was derived that supports rapid pressure sensitive multipoint interaction. Data is manipulated and visualized directly on the graphics processing unit (GPU), enabling highly interactive rates that allow users to manipulate the model on a per voxel basis, akin to work seen in the field of volume sculpting.

Early digital volumetric sculpting was performed by Gaylen and Hughes in 1991 [GH91] who used a custom made 3D positioning system to cut, add, and smooth a virtual block of clay. After each manipulation, an iso-surfacing pass was applied to create a polygonal mesh, used for subsequent visualization.

Wang, et al., used ray casting to create cuts through voxel grids based on standard mouse and keyboard input [WK95]. Wang visualized these results using a progressive ray marching approach based on a single isovalue. For refined imaging, the system used out-of-core ray tracing. In 2007, Perng used a modified marching cubes approach for volumetric carving with GPU support [PWFO01].

Several researchers have looked into haptic input devices as better interface tools for the analysis of volumetric data sets, and much of the work on volume sculpting relies on six degree of freedom (6-DOF) input devices [CS02] [CHS04]. Chen et al. used a 6-DOF input device to mimic the metaphors of melting, burning, painting, construction, stamping and peeling [CS02].

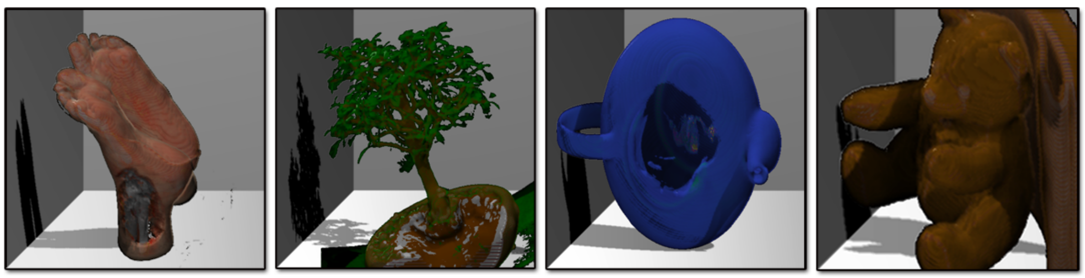

Figure 7.1: Examples of various reference data sets used for system tests. Courtesy of http://www.volvis.org/ and http://www.vis.uni-stuttgart.de/∼engel/pre-integrated/data.html

One of the major challenges with volumetric sculpting is the need to visualize results at interactive rates. In Chen's 2002 paper, the significance of this problem is apparent from Table 1 [CS02]. As voxel manipulation regions increase, the system's interactivity decreases exponentially. While bus and processor speeds have increased significantly since Chen's paper, they are still a limiting factor when performing large volumetric updates as shown in the results section below.

Bruckner et al. demonstrate a 3D illustration environment in their project VolumeShop [BVG05]. VolumeShop allows users to manipulate selected sub-volumes. These subregions can be easily manipulated, but, as shown in the paper, interaction rates are slowed dramatically as subregion sizes increase. Correa et al. demonstrated an engaging method for applying virtual peelers, retractors, pliers and dilators to volumetric data [CSC06]. While this method allowed for interactive modifications, it required data to be preprocessed. Additionally, multiple modifications caused problems when modification regions overlapped. McGuffin et al.

used deformation strategies on semantic layers which compose a volume data set [MTB03]. While this approach is incredibly powerful, it requires users to segment data into discrete layers such as skin, bone, muscle, etc.

Once data has been loaded into graphics memory, several techniques are commonly applied to extract meaningful information [Elv92] such as constructing tessellated models based on a single isovalue, using for example the marching cubes [LC87] algorithm. The advantage of this or similar techniques is that a single processing pass can be applied to extract the needed 3D information and standard 3D rendering and acceleration techniques can be subsequently used for the creation of interactive visuals. Johansson showed how this process could be moved to the GPU to further increase performance [JC06]. For the purposes of his paper, it was important to show multiple depth values concurrently and a GPU-accelerated ray marching visualization approach was selected.

Akeley proposed the possibility of using dedicated 3D hardware to accelerate volume rendering [Ake93] as part of the RealityEngine graphics pipeline and in 1994, Cabral et al. used an accelerated volumetric reconstruction via graphics hardware [CCF94]. Subsequently, Westermann [WE98] efficiently used commodity graphics hardware in volume rendering applications and Kniss et al., used graphics hardware to apply volumetric lighting and shadowing to volume models [KKH01].

Along with visualization, intuitive interaction is critically important. Yet, the aforementioned sculpting techniques only work on a single interaction point. Additionally, 6-DOF devices are generally decoupled from the display, forcing users to virtually feel the area in which they are manipulating.

Multi-touch surfaces support a natural coupling of visualization and user interface, allowing users to intuitively understand where and how they are changing the volume. Multi-touch surfaces also allow for multiple points to be inspected simultaneously.

The Virtual Autopsy Table [Ryd] demonstrated the ability to use multi-touch for visualizing medical data. This work utilized well known multi-touch pan and zoom metaphors to move volume data around. The manipulation of the volumetric data consisted of standard cutting planes and volume rendering.

As shown in Chapter 6 and Bonanni et al. [BXH+09], multi-touch devices can be used as an intuitive way to explore multi-layered data. Transitioning these types of interface modalities from layered imagery to volumetric data requires a new and novel approach.

This chapter presents Incision, a method for interactive, hands-on analysis of volumetric data, using intuitive metaphors. Hands-on volume analysis is enabled by a custom-built multi-touch system with a touch vocabulary and grammar, allowing users to freely transform the target volume, drill, scrape, and restore volumetric data types on localized regions and provides users with the ability to annotate regions of interest. Position, gesture and pressure information are used for voxel density and depth specific operations. Three-dimensional framebuffer-objects provide the support structure needed for GPU-centric processing and allow data to be modified at fixed cost. Realistic visualizations are produced through a multi-pass, multi-targeted rendering pipeline, supporting effects such as shading and shadowing, adding visual depth cues and realism. Figure 7.1 shows example data sets being manipulated, while Figure 7.2 illustrates user interaction.

With volume rendering, volume manipulation and multi-touch interaction closely interconnected, each is described to capture the research contribution. First the hardware assisted rendering algorithms are described, equations, associated data structures, equations, and rendering techniques. Since the volume manipulation uses the depth map generated in the volume rendering pass, the methodology for data manipulation is presented next. Finally, the hardware, metaphors, and vocabulary used for multi-touch centric modeling are described.

7.1.1 FrameBuffer Objects

Framebuffer objects (FBOs) provide a powerful mechanism for manipulating data on of the GPU. Standard two-dimensional FBOs enable the graphics card to render viewpoints to a texture rather than to the screen. Three-dimensional framebuffer objects enable rendering to a volume and work much like a stack of two-dimensional framebuffer objects. The framebuffer object attaches to a z-slice, and any of the x and y pixels of this attached slice can be modified on the GPU.

To change pixels with different z values, their z-slice must be attached accordingly.

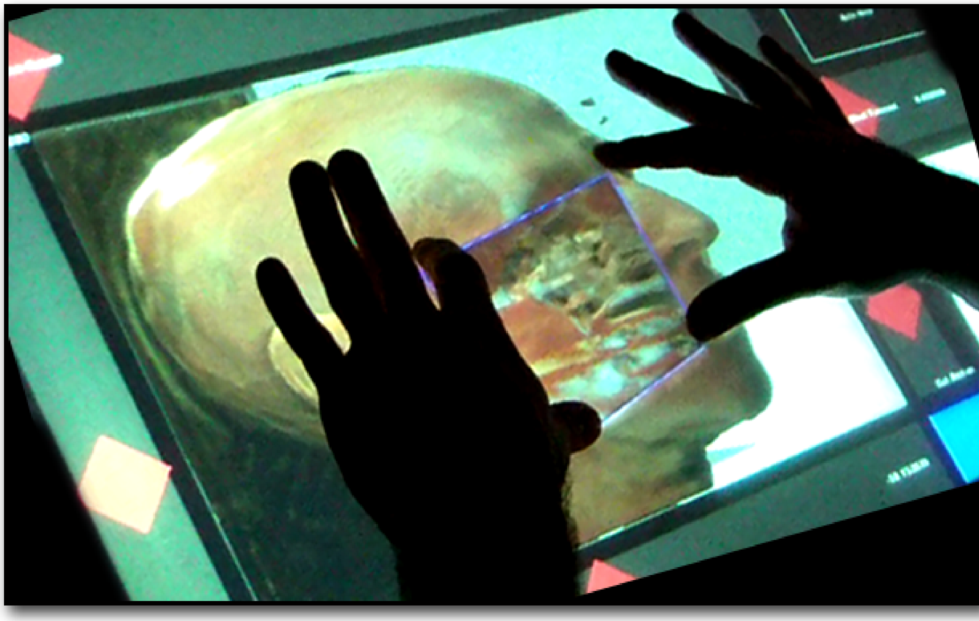

Figure 7.2: A user exploring volumetric data using the polygon mode.

For the volume to be modified with feedback, two three-dimensional FBOs are needed, one which functions as a read buffer and the second which functions as a write buffer. The buffer to be read from is named the front buffer and the buffer to be written to is named the back buffer, as seen in the OpenGL pipeline. Once the render-to-texture update finishes, the pointers to the front and back texture buffers are swapped. This step is necessary to avoid unwanted feedback loops between write and read operations. For Incision, five additional two-dimensional FBOs are used to store raster information for the rendered viewpoint. These FBOs serve as buffers for the rendering equation, shadow map, depth map, normal map, and shadow mask, as shown in Figure 7.3(A)-(E).

7.2 Volume Rendering

The initial volume rendering pass is similar to the approach presented by Cabral [CCF94]. Data slices are rendered parallel to the viewing direction in back-to-front order. Each of these data slices contains a transfer function. The volume rendering equation as shown in [EKE01] can be represented as the integral

$$I = \int_0^d color(x(\lambda))exp(-\int_0^\lambda extinction(x(\lambda'))d\lambda')d\lambda \tag{7.1}$$

where the viewing ray $x(\lambda)$ is parameterized by the distance from the viewpoint and that color density and extinction values may be calculated anywhere along this ray. Also, represents the maximum distance from the camera that can be stored.

The approximation for this integral is given by

$$C_i = \alpha_j C_i + (1 - \alpha_i)C_{i+1} \tag{7.2}$$

This representation can be approximated on the GPU by using a series of rectangles that are drawn in back-to-front order and bound to the volume texture. The number of slices should be equal to the depth of the volume at a minimum, but can also be increased to achieve ray super-sampling. Assuming a z-in orientation, the texture coordinates for each rectangle consist of the bounds of the volume in the x and y direction, and the z value equal to the slice, which is currently being rendered. By binding a fragment program to the render loop, the rendering equation is evaluated as shown in Figure 7.3(A).

It is important to note that the process to render every frame is the exact same regardless of content. This allows the entire drawing process to be handled through static display lists for additional performance.

7.2.1 Multiple Rendering Targets

With the volume rendering step being a costly operation, it is preferable to render it only once and produce additional visual effects based on the two-dimensional outputs. Since it is possible to render to multiple outputs in a single fragment program, multiple viewpoints and renderings can be produced in a single pass as shown in Figure 7.3. In the first rendering pass through the volume, three targets are specified to render to, and the volume is rendered in back-to-front order to ensure proper blending as shown in Figure 7.3(A). The output of the rendering equation is stored in the first render target. The second render target stores the shadow map and the third render target the depth map (Figure 7.3(B)(C)). The second pass uses the output of the depth buffer to generate the shadow mask and the normal map as shown in Figure 7.3(D)(E). Finally, the rendering output, normal map, and shadow mask are all used to generate the final composite (Figure 7.3(F)).

7.2.2 Matrix Manipulation

In order to change the viewpoint from which the volume is being studied, the transformation matrix is multiplied by the texture coordinates in the vertex program before being passed to the fragment program. If the modelview transformation matrix for the users viewpoint is stored in the matrix M, and the original unit cube texture location is T_{IN}, then the resulting texture lookup for the fragment program will be:

$$T_{Eye} = M^{-1} T_{IN} \tag{7.3}$$

similarly, the texture lookups for the light can be computed by

$$T_{Light} = M^{-1}(L^{-1} T_{IN}) \tag{7.4}$$

where L is the transformation matrix for the light. The light is rotated with the eye to keep it in the same relative position with the object, emulating the effect of moving the volume rather then the viewing direction. If the desired effect is that of moving the viewing direction, the T_{Light} would not need to be multiplied by M^{-1}.

7.2.3 Lighting and Shading

One common shortcoming of the approach proposed by Cabral et al. [CCF94] is that it does not include shading or lighting. Lum et al. documented the improve-

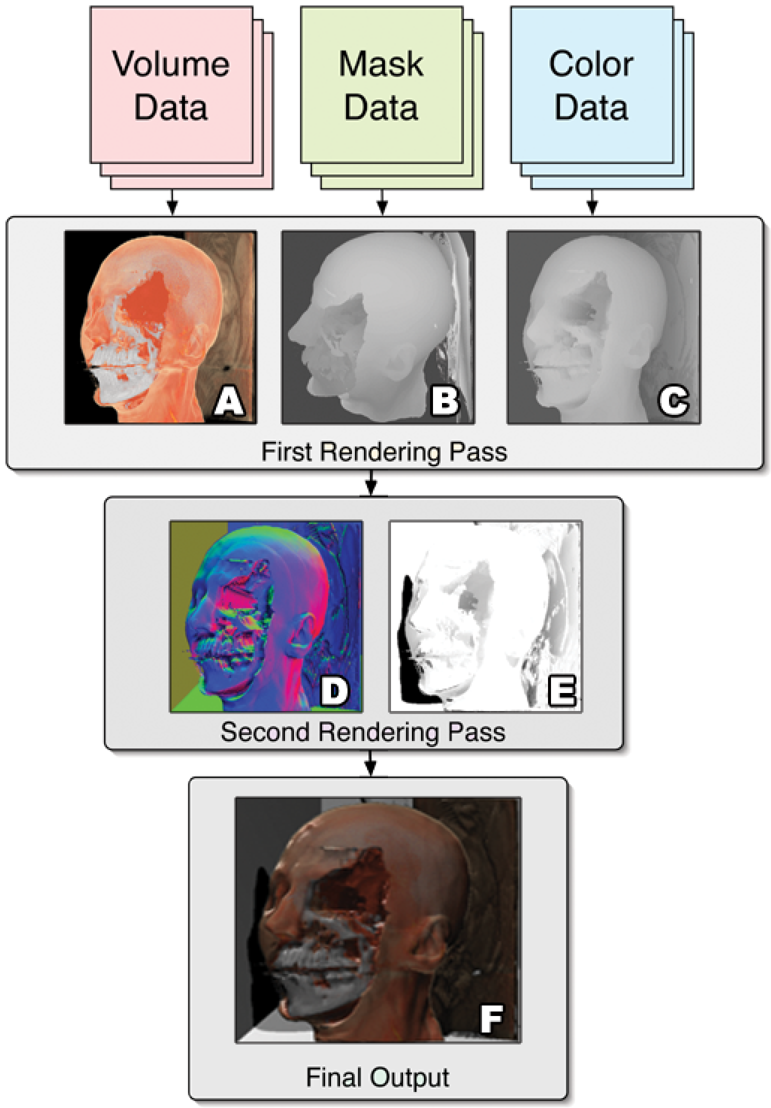

Figure 7.3: Illustration of the multi-pass rendering pipeline, showing (A) result of the front to back rendering, (B) shadow map, and (C) depth map created during the first rendering pass, the (D) normal map, and (E) shadow mask created during the second rendering pass and (F) the resulting composited image.

ments lighting adds to volume rendering [LM04] by providing helpful topological and depth cues.

For the Phong shading model, the lighting equation can be represented as

$$C = C_{voxel} * (k_a + *k_d * MAX(N * L, 0)) + k_s * MAX((N * R)^n, 0) \tag{7.5}$$

where C_{voxel} is the incoming color, N is the normal vector, L is the normalized light vector, R is the reflection vector, k_d, k_s, k_a are the diffuse, specular, and ambient representation respectively, and n is the specular shininess of the material [LM04].

The difficulty of applying this type of shading is in computing the surface normal, as there is inherently no predefined normal as which comes from triangulated meshes.

7.2.4 Normals

In order to apply the Phong shading model, surface normals need to first be determined. While Sobel gradients, as described in [HLSR08] produce high quality results, they require 26 additional volumetric texture lookups and only determine normals over a 3x3x3 voxel grid.

Lum [LM04] and Hadwiger [HLSR08] propose using forward gradients which only check three surrounding pixels as an optimized way of producing these gradient values. As shown in Hadwiger's paper, this produces faster but lower quality results [HLSR08].

As a compromise of speed and quality, pseudo normals are created based on the camera's depth map. Vectors can then be created using the texture values for x and y using the camera depth map to infer z values. By creating vectors for the surrounding depth map pixels, the surface normal can be approximated for localized regions as shown in Figure 7.3(D). While this approximation produced smoother looking normals, it also increases the frame rate by 200% compared to using sobel gradients. Pseudo code is shown in the Appendix A.3.

7.2.5 Shadow Maps

Shadows are a powerful cognitive tool for relating depth and space to a user. Shadows also add an extra sense of realism for rendered scenes. In their 1998 paper, Behrens and Ratering found that shadows significantly improved the spatial understanding of volume data if rendered at interactive frame rates [BR98].

Behren's presented an algorithm for storing shadows inside of the volumetric data structure. This approach unfortunately was not fully parallelizable as each subsequently rendered slice needed to be fully completed before the next slice could be rendered. As shown in this paper, this method causes performance to degrade upto of 50% [BR98].

Methods such as Deep Shadow Maps can be computed on the GPU, but require much of the information to pre-computed as well as pre-compressed [HKSB06]. Since the volume manipulation for the hands-on volume analysis technique has to be dynamic, this was not an option. Zhang implemented a GPU based approach without pre-computation, but as shown in the paper frame rates declined to 10 fps when dealing with volume data [Zha09]

When generating shadow maps with polygonal rendering, the scene must be rendered twice, first from the point of view of the user and subsequently from the point of view of the light. As shown above, this can be done in parallel using multiple render targets.

The depth map is computed first for the eye using the lookup equation from above. The equation for transforming a vector from the eye's point of view (V_{Eye}) to the light's viewpoint can be expressed as

$$V_{Light} = M^{-1}(L^{-1}(MV_{Eye})) \tag{7.6}$$

where M is the eyes transformation matrix and L is the light's transformation matrix. By comparing the depth value stored in the shadow map to that of the projected value stored in the depth map for the same raster point, the shadow mask can be determined (Figure 7.3(E)). Utilizing the benefits of multiple render targets, producing the shadow map comes at a minimal cost. Pseudo code for the lighting and shading used is shown in the Appendix A.4.

7.2.6 Compositing

The final result is generated by compositing the end products of the previous rendering passes (Figure 7.3(F)). The resulting pixel color is defined as

$$\begin{aligned} C_{result} = ShadowMask_{xy} * ColorMap_{xy} * \\ (k_a + k_d * MAX(NormalMap_{xy} * L, 0)) \\ +k_s * MAX((NormalMap_{xy} * R)^n, 0) \end{aligned}$$

where the k_d, k_a, k_s and n are all input parameters, L and R are calculated in the shader, and ShadowMask, ColorMap, and the NormalMap are sampled from previous rendered outputs based on the current raster position xy.

7.3 Interaction

As a pre-requisite for an intuitive and interactive environment, a highly responsive system with well known interface widgets such as buttons and sliders was targeted as the baseline.

7.3.1 Widgets

Surface widgets allow users to customize the visualization, mode of operation, and interaction preferences. Since the interface is designed for multi-touch from the bottom up, sliders, buttons, and volume manipulations can all be manipulated simultaneously. For example, this allows users to modify the volume with one hand while spinning the volume with the other. The ability to detect the pressure applied by the user while interacting with that multi-touch surface, supports pressure and pressure range specific operations.

7.3.2 Transformations

At any point in time the user may rotate the model via sliders on the top and bottom of the model's view aera. Additionally, the user can switch into the transformation mode. In this mode, user's simply touch the virtual model and use a trackball metaphor to spin it freely via hand movements on the table surface.

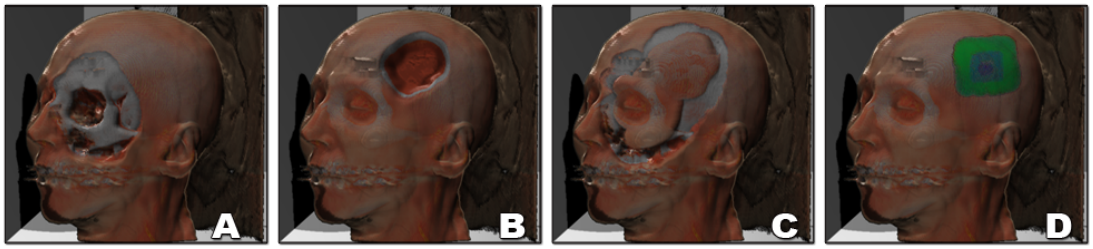

Figure 7.4: Illustration of the different interrogation modes (A) showing the scraping modality, (B) the drilling modality, (C) the healing modality, (D) the annotation modality.

7.3.3 Volume Modification

Multiple modes are used to change the volume, relying on a similar methodology. First, up to sixteen points with radius are packed and uploaded to the GPU. The number sixteen was arbitrarily chosen and the code could easily accommodate more. Currently if more than 16 touches are on the screen, the shader will be run multiple times until all of the touches are accounted for.

To modify the volume, it is first rendered in slices along the z direction. Each of these slices are attached to, and modified directly on the GPU. Each pixel fragment will be transformed from the original unit coordinate system into the eye space coordinate system. The surface nearest to the eye can also be modified using the depth map. As the depth map was already computed for the rendering of the volume, detecting the nearest surface incurs no additional cost.

7.3.4 Scraping Mode

In the scraping mode, users can wipe away surfaces mimicking the mode of operation seen in [BXH$^+$09] and Chapter 6. The pressure mapping can be easily controlled via a pressure widget on the side. This produces straight forward means to peel off volume layers such as flesh, muscle and bone for medical data sets as shown in Figure 7.4(A).

Pseudo code is as follows:

$T_{EyeSpace} = M * T_{IN}$

$T_{EyeSpace_z} = texLookup(depthMap, T_{EyeSpace_{xy}})$

$if(abs(T_{EyeSpace} - T_{IN}) < blobSize) AND (Pressure > T_{IN_{data}})$

$\quad Volume_{OUT_{mask}} = 0$

7.3.5 Drilling Mode

In the drilling mode, each of the user's touches will cut through the volume based on the user's viewpoint and the pressure of each touch. This is useful for drilling into the volume in order to see data on the inside as shown in Figure 7.4(B).

Pseudo code is as follows:

$T_{EyeSpace} = M * T_{IN}$

$if(Pressure > T_{EyeSpace_z})$

$\quad Volume_{OUT_{mask}} = 0$

7.3.6 Healing Mode

Figure 7.5: Automatic volume healing image sequence. Frame numbers are shown in the lower right-hand corner. For this example, the autohealing was activated at frame 0 and took 20 frames to complete.

Healing mode works to restore previously removed sections. Conceptually the inverse of scraping, voxels which are visibly nearest to the camera are re-added when the isovalue is less than or equal to the pressure. This allows for lesser isovalues to easily be restored as shown in Figure 7.4(C).

Pseudo code is as follows:

$T_{EyeSpace} = M * T_{IN}$

$T_{EyeSpace_z} = texLookup(depthMap, T_{EyeSpace_{xy}})$

$if(abs(T_{EyeSpace} - T_{IN}) < blobSize)\ AND\ (Pressure > T_{IN_{data}})\quad AND\ (T_{IN_{mask}} < 1)$

$\quad Volume_{OUT_{mask}} += healAmout$

7.3.7 Annotation Mode

The annotation mode is used to visually annotate the volume itself (Figure 7.4(D)). The voxel closest to the viewing direction along the path of the eye ray has its color channel set to the current annotation look up color, similar to the procedure used in [BVG05]. As this lookup is only 8 bits, no more than 256 colors can be used on the scene simultaneously. In practice, users felt 256 colors were a sufficient for volume annotation. Since these colorizations are volumetric, they can subsequently be manipulated with any of the other analysis metaphors.

Pseudo code is as follows:

$T_{EyeSpace} = M * T_{IN}$

$T_{EyeSpace_z} = texLookup(depthMap, T_{EyeSpace_{xy}})$

$if(abs(T_{EyeSpace} - T_{IN}) < blobSize)$

$\quad Volume_{OUT_{color}} = colorLookupValue$

7.3.8 Polygon Mode

Quick regional inspection of volumetric data is possible with a polygonal mode. For this mode, all user touches are run through a convex hull algorithm [Gra72] in order to determine the encompassing polygon. The color for each vertex of the encompassing polygon is matched with pressure of the corresponding touch. The encompassing polygon is then rendered into a FBO. This FBO is used as a lookup for volume deformation based on the current viewpoint.

Pseudo code is as follows:

$T_{EyeSpace} = M * T_{IN}$

$p = texLookup(pressureMap, T_{EyeSpace_{xy}})$

$if(p > T_{EyeSpace_z})$

$\quad Volume_{OUT_{mask}} = 0$

7.3.9 Automatic Volume Healing

Since the entire volume is being analyzed on the GPU, it is also possible to create"self-healing" technique that can dynamically revert from the current state to the visual representation of the original, unmodified volume. This mimics the effect shown in the Khronos projector [CI05] on a three-dimensional volume as opposed to a two-dimensional clipping plane. This effect would be difficult to implement in realtime via the CPU since the entire volume has to be parsed and large sections need to be changed.

This approach functions as a three-dimensional dilation filter with the addition that dilation can only occur between voxels with similar isovalues, allowing skin to heal surrounding skin while bone would not be able to heal skin. The speed at which this healing takes place is fully adjustable. When set to rapid healing rates, the algorithm acts almost as a virtual "flashlight", providing updated information for the section that is currently being analyzed. As soon as the user stops interacting with the target region or moves away from it, localized changes are immediately reversed. An example frame series of volume healing is shown in Figure 7.5.

Pseudo code is as follows:

$if(T_{IN_{mask}} == 0)$

$\quad commonNeighbor = 0;$

$\quad for\ voxels\ V\ between\ (-1,-1,-1)\ to\ (1,1,1)$

$\quad\quad if(T_{IN} - V_{xyz})_{data}\ is\ common\ to\ T_{IN}$

$\quad\quad AND\ ((T_{IN} - V_{xyz})_{mask} == 1)$

$\quad\quad\quad commonNeighbor = 1;$

$\quad if(commonNeighbor)$

$\quad\quad Volume_{OUT_{mask}} = 1$

7.4 Design Decisions and Limitations

The primary design decisions for the algorithms and interfaces at the software and hardware level are based on the desire to create highly-interactive and realistic visuals of volumetric data. This approach is not without limitations though.

The most notable is that it requires a graphics card which supports the glFramebufferTexture3DEXT extension and texture memory which is large enough to hold the entire volume. In addition, for the ability to scrape, drill, heal and annotate, three color-channels must be stored per voxel increasing the required memory footprint. Furthermore, to prevent feedback loop issues, front and back volumes must be stored. While this large memory space is limiting at present, it is important to note that this method removes unneeded data transfers and is an effective way for modifying and visualizing mass amounts of data as long as this condition is met, as the results section demonstrates.

Given the feature set of current graphics cards this constraint appears to be tolerable when working with most mainstream volume data set sizes. There also is evidence that with the current growth rate of graphics card texture memory and alternate memory architectures, this issue may soon be mute. As shown in the results section below, as long as the data does fit into the graphics card memory, the method described in this paper scales in linear fashion.

One of the major limitations of this approach comes from the implementation of the three-dimensional framebuffer objects. Three-dimensional framebuffer objects can be randomly written to in the x and y plane, but only for a given attached z slice. This is limiting for the implementation of how data can be modified and proves to be a major bottleneck in terms of performance. To render the entire volume, every single z-slice must be attached and drawn into.

One common optimization of volumetric data visualization is to run a preprocessing step to pre-compute needed data structures. Attributes such as normals, geometry, and volume hierarchies can all be computed a priori and lead to quicker processing for drawing and the ability to visualize data sets which are too large to fit into texture memory. Since the objective was to allow the user to swiftly and continuously manipulate the data, these pre-optimizations are not helpful. Additionally, volume hierarchies such as oct-trees and kd-trees do not fit well into the render-to-texture model.

Commonly, when render times cannot keep up with the desired frame rate, a preview example is rendered and shown until the full rendering can be completed.

This can be done by either using less samples in the raster x and y directions, or by sampling the ray less along the z-in direction. This optimization unfortunately would greatly reduce the effectiveness and intuitiveness of the system described in this paper. The primary reason for this is that as soon as users attempt to manipulate the volume, the render engine may be forced to switch to a lower resolution preview, with visual artifacts of this preview including incorrect object transparency and physical position. These effects are incredibly unnatural and jarring, which is why this optimization was not chosen.

7.5 Applications

The approach described in this paper has many applications in the fields of volume analysis and sculpting with applications in medicine and non-destructive evaluation, to name a few. Volume sculpting is usually done with only a single point of interaction. By allowing multiple points of interaction of various sizes, volume sculpting can be done in accelerated and more intuitive fashion. The presented approach also allows for data analysis to be done in localized regions. Generally, volumetric data interrogation is only done on a global scale. While iso-surface values can be modified, they are changed over the entire volume. The presented system allows for localized areas to be modified without effecting surrounding regions.

Medical professionals have shown a great deal of interest and enthusiasm in this approach. By taking CT scans of a patient, a surgeon for example, can flexibly reveal the anatomy, practice cuts, label areas of interest and conceptualize three-dimensional relationships of topological features and different tissue types. This interactive and realistic representation could also be used to demonstrate the procedure in front of the patient in ways the patient could easily understand. Plastic surgeons could also use this visualization to model modifications on a realistic representation of the patient based on previous medical scans.

7.6 Results

Incision's hands-on techniques for the analysis of arbitrary sized regions in volumetric data sets were evaluated through an initial user survey and a set of performance tests of small, medium and larger sized baseline samples, including the CT volume of a human head, an MRI scan of the upper body, and a computer generated volume, respectively. The derived performance metric distinguishes between volume manipulation and volume rendering.

7.6.1 Initial User Testing

Users were asked to identify how internal volume anomalies are correlated to surface features for a set of baseline cases. Anomalies consisted of regions which were filled with abnormally high density values and regions which which were filled with abnormally low density values. Users were instructed to locate these internal abnormalities underneath a grid, etched on the boundary of the volume model. For each trial the size of the abnormality decreased, from an initial size of 10^3 voxels, to 5^3 voxels to finally a size of 3^3 voxels.

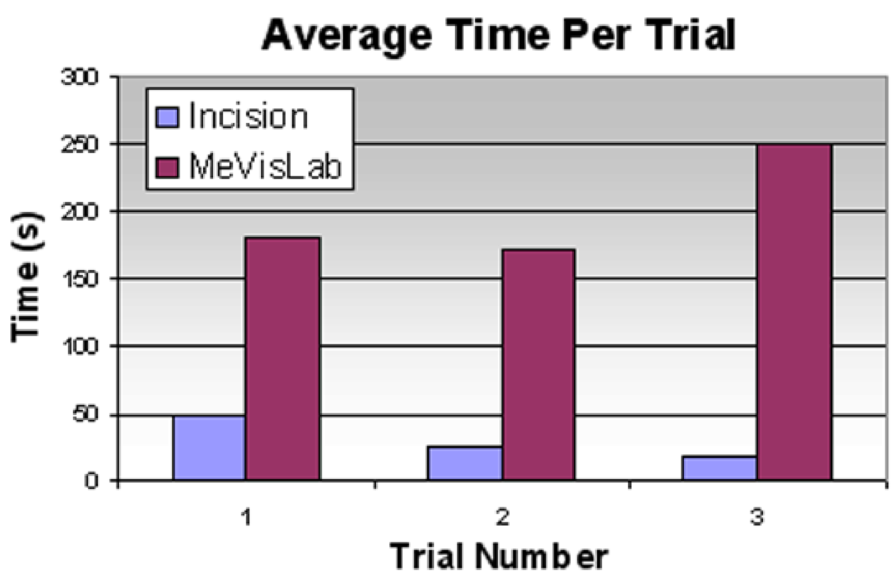

Figure 7.6: Average time taken to complete each trial for the initial user study.

As a comparison to the Incision system, MeVisLab was selected, which implements both VTK and ITK in a graphic user environment [KSR+06]. Its volume renderer with enabled cutting planes was selected as comparison basis, as it most closely matches the capabilities of the Incision system.

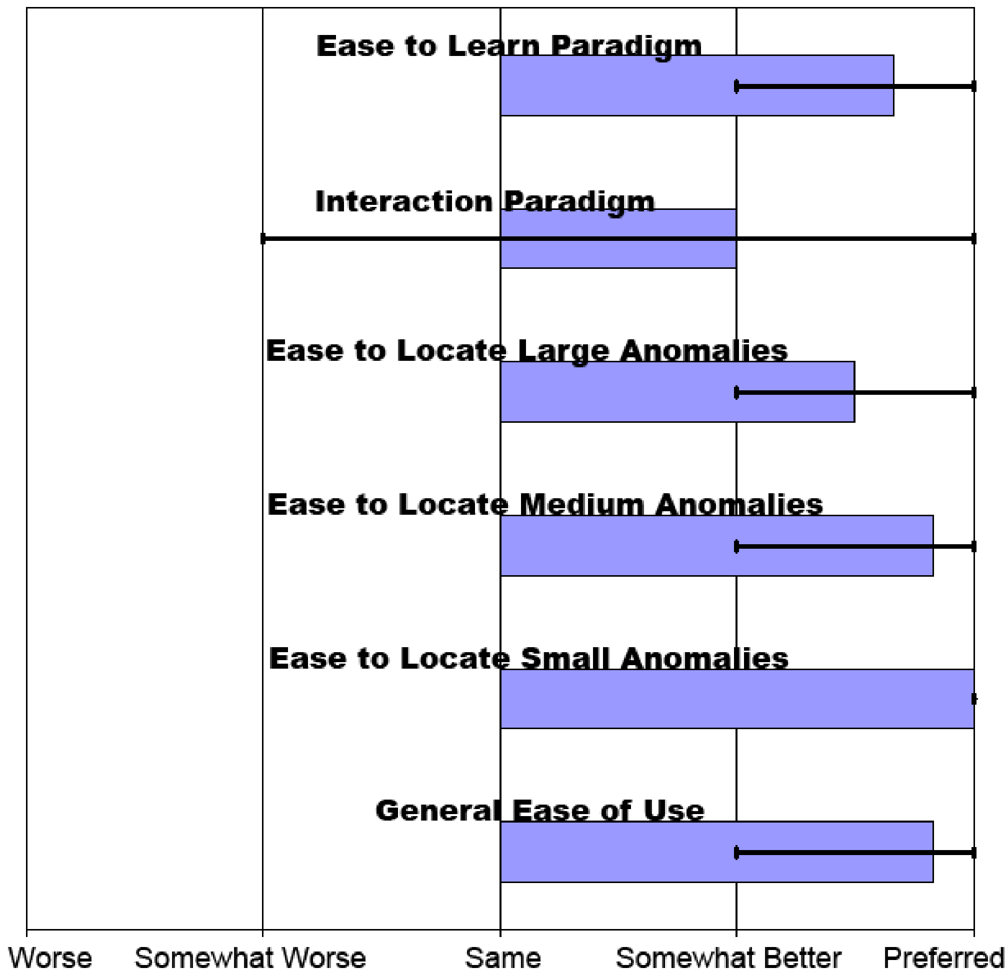

Figure 7.7: User ratings comparing Incision to volume rendering via MeVisLab. Black bars indicate the maximum and minimum score for the given question.

Six users were selected which had little experience on either system. This group consisted of two females and four males all ages twenty to forty. As show in Figure 7.6 users were able to accomplish the given task using the Incision system in less then a third of the time required for MeVisLab. Furthermore as users became more accustomed to the Incision system, the time required to finish the required task decreased further, even with increasing task difficulty. In comparison, the time required for Trial 3 of MeVisLab was approximately 70% greater than the time required for Trial 1.

Furthermore in qualitative ratings, the Incision system scored better in every category compared to MeVisLab as shown in Figure 7.7. While one participant ranked Incision worse for user interaction, in all other categories Incision was considered to be better than MeVisLab. Furthermore, in every category Incision received at least one vote that it was a better than MeVisLab and when locating small anomalies Incision received a unanimous affirmation that it was the better system to use.

7.6.2 Volume Modification

The common approach of modifying volumetric data is to modify the data on the CPU and then upload the changes to the GPU in order to visualize the results. This approach works well for input devices that set a 3D position and modify small amounts of data. This setup is comprised of three steps, changing the volume data in the CPU, filling in a data structure for upload, and finally uploading the data to the GPU. Figure 7.8 shows the time required to change and upload data using this traditional CPU-centric technique in contrast to the GPU-centric approach described in this paper.

As shown, the GPU approach has a fixed cost when changing the entire volume, which is at the same complexity of changing an individual voxel. This is due to the fact that the entire volume is rendered in the render-to-texture pass. While optimizations could be made to only re-render the desired z-slices to be modified, the time it would take to calculate which z-slices needed to be rendered would take longer than the time to render the entire volume itself.

The major bottleneck in the GPU approach was attaching the z-slices. As shown in Figure 7.8, this enables this GPU approach to scale linearly with the number of depth-planes for the volume.

Conversely, the CPU approach scales proportionally to the number of voxels in the data set. While this approach is actually faster than the GPU in cases where only a small amount of data is changed, for larger amounts of data this approach

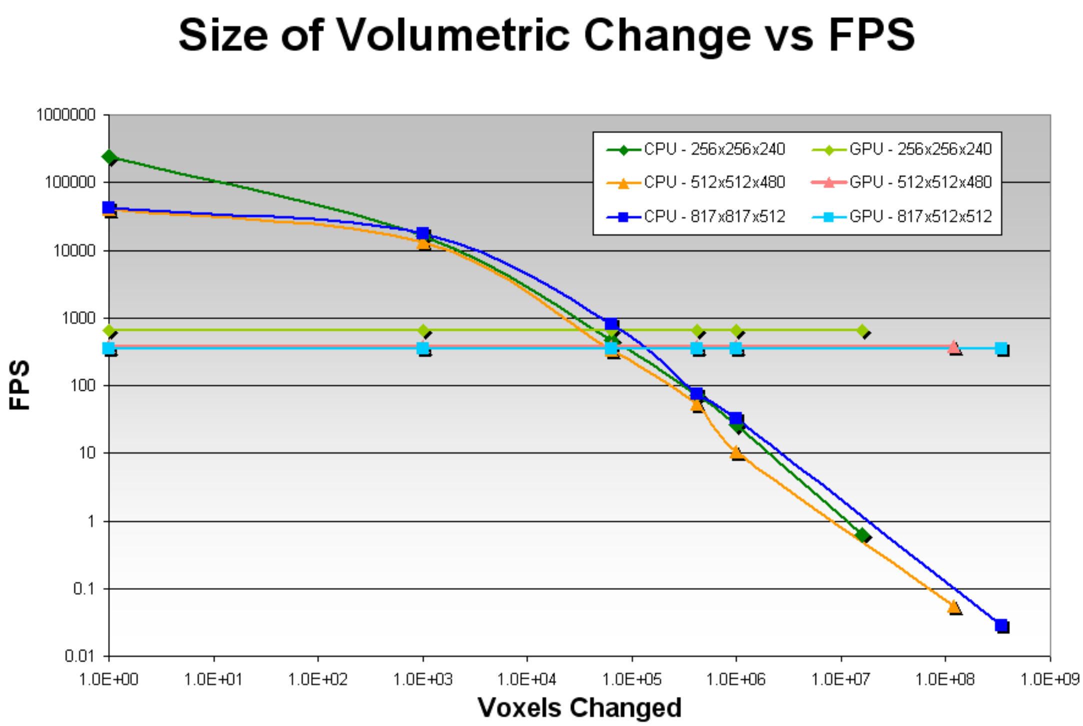

Figure 7.8: Render performance as a function of volume complexity expressed by the number of modified voxels, using a CPU and GPU centric approach.

quickly becomes incredibly time consuming as shown in Figure 7.8. It should also be noted that the changed data must fit inside of the axis aligned bounding box, which encompasses the changes. This means that even if only a moderate amount of data needs to be updated, if the data is spread out, a bounding volume as large as the distribution would need be uploaded, causing detrimental performance.

7.6.3 Visualization

The visualization pipeline scales as a function of the number of rays and the sampling rate of the rays. Table 7.1 shows the frame rates for three differently sized volumetric data sets with various sample rates. As expected, the higher the sampling rate, the higher the quality of the result, and the lower the frame rate. For general use, 256^3 sized volume sets could easily perform at highly interactive rates (greater than 60 Hz) as shown with a single voxel per sample.

Table 7.1: Performance of various sampling rates

Volume Size: 256x256x240

Samples per Voxel	1	4	8	64
FPS	63	42	32	12

Volume Size: 512x512x460

Samples per Voxel	1	4	8	64
FPS	12	7	5	1

Volume Size: 817x817x512

Samples per Voxel	1	4	8	64
FPS	7	5	2.5	0.7

7.7 Conclusion

This chapter presents a method for intuitive, hands-on analysis of volumetric data, using simple metaphors. Interactive rendering rates are achieved by utilizing the massive parallelism of the GPU for multi-pass rendering, supporting dynamically changing visuals. The write-to-volume capability provided by three dimensional framebuffer objects is used to create the support structure for in-core modifications directly on the GPU, allowing large amounts of data to be modified at fixed cost. Multi-pass, multi-targeted rendering supporting effects such as shading and shadowing enhances visual depth cues and overall realism.

Hands-on volume analysis is enabled by a custom-built multi-touch system with a touch vocabulary and grammar, allowing users to freely transform the target volume, drill, scrape, annotate and restore volumetric data types on localized regions. Position, gesture and pressure information are used for voxel density and depth specific operations. With all interaction metaphors at most one touch or hand-gesture away, users were able to begin analyzing data with no or limited self-training and were at large proficient within seconds or minutes.

7.8 Acknowledgments

This chapter is currently being prepared for submission for publication of the material. Ponto, K., Doerr, K., and Kuester, F. The dissertation author was the primary investigator and author of this paper.

Bibliography

[Ake93] Kurt Akeley. Reality engine graphics. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 109–116, New York, NY, USA, 1993. ACM.

[bbc] http://www.apple.com/quicktime/guide/hd/bbc-cfb.html.

[BJH+08] Allen Bierbaum, Christopher Just, Patrick Hartling, Kevin Meinert, Albert Baker, and Carolina Cruz-Neira. Vr juggler: a virtual platform for virtual reality application development. In *SIGGRAPH Asia '08: ACM SIGGRAPH ASIA 2008 courses*, pages 1–8, New York, NY, USA, 2008. ACM.

[BN05] Robert Ball and Chris North. Effects of tiled high-resolution display on basic visualization and navigation tasks. In *CHI '05 extended abstracts on Human factors in computing systems*, pages 1196–1199, New York, NY, USA, 2005. ACM.

[BR98] Uwe Behrens and Ralf Ratering. Adding shadows to a texture-based volume renderer. In *VVS '98: Proceedings of the 1998 IEEE symposium on Volume visualization*, pages 39–46, New York, NY, USA, 1998. ACM.

[BVG05] Stefan Bruckner, Ivan Viola, and M. Eduard Gröller. Volumeshop: interactive direct volume illustration. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Sketches*, page 60, New York, NY, USA, 2005. ACM.

[BXH+09] Leonardo Bonanni, Xiao Xiao, Matthew Hockenberry, Praveen Subramani, Hiroshi Ishii, Maurizio Seracini, and Jurgen Schulze. Wetpaint: scraping through multi-layered images. In *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems*, pages 571–574, New York, NY, USA, 2009. ACM.

[car] http://www.apple.com/trailers/disney/cars/.

[CB04] Xiang Cao and Ravin Balakrishnan. Visionwand: interaction techniques for large displays using a passive wand tracked in 3d. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 729–729, New York, NY, USA, 2004. ACM.

[CCF94] Brian Cabral, Nancy Cam, and Jim Foran. Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. In *VVS '94: Proceedings of the 1994 symposium on Volume visualization*, pages 91–98, New York, NY, USA, 1994. ACM.

[CEM01] F. Capani, M.H. Ellisman, and M.E. Martone. Filamentous actin is concentrated in specific subpopulations of neuronal and glial structures in rat central nervous system. *Brain Research*, 923(1-2):1–11, 2001.

[Che02] Han Chen. A parallel ultra-high resolution mpeg-2 video decoder for pc cluster based tiled display system. to appear. In *Proc. Int'l Parallel and Distributed Processing Symp. (IPDPS), IEEE CS*, page 30. Press, 2002.

[Che03] Han Chen. *Scalable and Ultra-High Resolution MPEG Video Delivery on Tiled Displays*. PhD thesis, Princeton University, 2003.

[CHS04] Ian Creighton and Chris Ho-Stuart. A sense of touch in online sculpting. In *GRAPHITE '04: Proceedings of the 2nd international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, pages 118–122, New York, NY, USA, 2004. ACM.

[CI05] Alvaro Cassinelli and Masatoshi Ishikawa. Khronos projector. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Emerging technologies*, page 10, New York, NY, USA, 2005. ACM.

[Cor09] Carlos D. Correa. Visualizing what lies inside. *SIGGRAPH Comput. Graph.*, 43(2):1–6, 2009.

[CS02] Hui Chen and Hanqiu Sun. Real-time haptic sculpting in virtual volume space. In *VRST '02: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 81–88, New York, NY, USA, 2002. ACM.

[CSC06] Carlos Correa, Deborah Silver, and Min Chen. Feature aligned volume manipulation for illustration and visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1069–1076, 2006.

[CSM02] E.F. Churchill, D.N. Snowdon, and A.J. Munro. Collaborative virtual environments: digital places and spaces for interaction. *Educational Technology & Society*, 5(4), 2002.

[CT09] Andrew A. Chien and Nut Taesombut. Integrated resource management for lambda-grids: The distributed virtual computer (dvc). *Future Generation Computer Systems*, 25(2):147 – 152, 2009.

[DC02] James Davis and Xing Chen. Lumipoint: multi-user laser-based interaction on large tiled displays. *Displays*, 23(5):205 – 211, 2002.

[(DC06] Digital Cinema Initiatives (DCI). Standard evaluation material (stem), 2006.

[DDS+09] Thomas A. DeFanti, Gregory Dawe, Daniel J. Sandin, Jurgen P. Schulze, Peter Otto, Javier Girado, Falko Kuester, Larry Smarr, and Ramesh Rao. The starcave, a third-generation cave and virtual reality optiportal. *Future Generation Computer Systems*, 25(2):169 – 178, 2009.

[DK10] Kai-Uwe Doerr and Falko Kuester. CGLX: A Scalable, High-performance Visualization Framework for Networked Display Environments. *IEEE Transactions on Visualization and Computer Graphics*, 99(PrePrints), 2010.

[DL01] P. Dietz and D. Leigh. Diamondtouch: a multi-user touch technology. In *Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 219–226. ACM New York, NY, USA, 2001.

[DLR+09a] Thomas A. DeFanti, Jason Leigh, Luc Renambot, Byungil Jeong, Alan Verlo, Lance Long, Maxine Brown, Daniel J. Sandin, Venkatram Vishwanath, Qian Liu, Mason J. Katz, Philip Papadopoulos, Joseph P. Keefe, Gregory R. Hidley, Gregory L. Dawe, Ian Kaufman, Bryan Glogowski, Kai-Uwe Doerr, Rajvikram Singh, Javier Girado, Jurgen P. Schulze, Falko Kuester, and Larry Smarr. The optiportal, a scalable visualization, storage, and computing interface device for the optiputer. *Future Gener. Comput. Syst.*, 25(2):114–123, 2009.

[DLR+09b] Thomas A. DeFanti, Jason Leigh, Luc Renambot, Byungil Jeong, Alan Verlo, Lance Long, Maxine Brown, Daniel J. Sandin, Venkatram Vishwanath, Qian Liu, Mason J. Katz, Philip Papadopoulos, Joseph P. Keefe, Gregory R. Hidley, Gregory L. Dawe, Ian Kaufman, Bryan Glogowski, Kai-Uwe Doerr, Rajvikram Singh, Javier Girado, Jurgen P. Schulze, Falko Kuester, and Larry Smarr. The optiportal, a scalable visualization, storage, and computing interface device for the optiputer. *Future Generation Computer Systems*, 25(2):114 – 123, 2009.

[EKCB03] Jr. Easton, R.L., K.T. Knox, and W.A. Christens-Barry. Multispectral imaging of the archimedes palimpsest. *Applied Imagery Pattern Recognition Workshop, 2003. Proceedings. 32nd*, pages 111–116, Oct. 2003.

[EKE01] Klaus Engel, Martin Kraus, and Thomas Ertl. High-quality pre-integrated volume rendering using hardware-accelerated pixel shading. In *HWWS '01: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, pages 9–16, New York, NY, USA, 2001. ACM.

[Elv92] T. Todd Elvins. A survey of algorithms for volume visualization. *SIGGRAPH Comput. Graph.*, 26(3):194–201, 1992.

[FAJ07] G. Flint, C. Aves, and MT Jones. The gigapxl project. ttp://www.gigapxl.org, 2007.

[GH91] Tinsley A. Galyean and John F. Hughes. Sculpting: an interactive volumetric modeling technique. *SIGGRAPH Comput. Graph.*, 25(4):267–274, 1991.

[Gra72] R. L. Graham. An efficient algorith for determining the convex hull of a finite planar set. *Information Processing Letters*, 1(4):132 – 133, 1972.

[GRC+07] J.F. Gantz, D. Reinsel, C. Chute, W. Schlichting, J. McArthur, S. Minton, I. Xheneti, A. Toncheva, and A. Manfrediz. The expanding digital universe: A forecast of worldwide information growth through 2010. *IDC white paper*, 2007.

[GSW01] François Guimbretière, Maureen Stone, and Terry Winograd. Fluid interaction with high-resolution wall-size displays. In *UIST '01: Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 21–30, New York, NY, USA, 2001. ACM.

[HA08] J. Heer and M. Agrawala. Design considerations for collaborative visual analytics. *Information Visualization*, 7(1):49–62, 2008.

[Han05] J.Y. Han. Low-cost multi-touch sensing through frustrated total internal reflection. In *Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 115–118. ACM New York, NY, USA, 2005.

[Har90] Stevan Harnad. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42(1-3):335 – 346, 1990.

[HEB+01] Greg Humphreys, Matthew Eldridge, Ian Buck, Gordan Stoll, Matthew Everett, and Pat Hanrahan. Wiregl: a scalable graphics system for clusters. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 129–140, New York, NY, USA, 2001. ACM.

[Her08] L. Herr. Creation and Distribution of 4 K Content. *Television Goes Digital*, page 99, 2008.

[HKSB06] M. Hadwiger, A. Kratz, C. Sigg, and K. Bühler. Gpu-accelerated deep shadow maps for direct volume rendering. In *Graphics Hardware 2006: Eurographics Symposium Proceedings, Vienna, Austria, September 3-4, 2006*, pages 49–52. Eurographics Association, 2006.

[HLSR08] Markus Hadwiger, Patric Ljung, Christof Rezk Salama, and Timo Ropinski. Advanced illumination techniques for gpu volume raycasting. In *SIGGRAPH Asia '08: ACM SIGGRAPH ASIA 2008 courses*, pages 1–166, New York, NY, USA, 2008. ACM.

[HYB02] T. Hansen, P. Yalamanchili, and H.W. Braun. Wireless measurement and analysis on HPWREN. In *Proceedings of Passive and Active Measurement Workshop, Fort Collins, Co*, pages 222–229, 2002.

[JC06] G. Johansson and H. Carr. Accelerating marching cubes with graphics hardware. In *Proceedings of the 2006 conference of the Center for Advanced Studies on Collaborative research*, page 39. ACM New York, NY, USA, 2006.

[JJR+05] B. Jeong, R. Jagodic, L. Renambot, R. Singh, A. Johnson, and J. Leigh. Scalable graphics architecture for high-resolution displays. In *IEEE Information Visualization Workshop*, 2005.

[JRJ+06a] Byungil Jeong, L. Renambot, R. Jagodic, R. Singh, J. Aguilera, A. Johnson, and J. Leigh. High-performance dynamic graphics streaming for scalable adaptive graphics environment. In *SC 2006 Conference, Proceedings of the ACM/IEEE*, pages 24 –24, 11-17 2006.

[JRJ+06b] Byungil Jeong, Luc Renambot, Ratko Jagodic, Rajvikram Singh, Julieta Aguilera, Andrew Johnson, and Jason Leigh. High-performance dynamic graphics streaming for scalable adaptive graphics environment. In *SC '06: Proceedings of the 2006*

ACM/IEEE conference on Supercomputing, page 108, New York, NY, USA, 2006. ACM.

[KKH01] J. Kniss, G. Kindlmann, and C. Hansen. Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets. In *Proceedings of the conference on Visualization'01*, pages 255–262. IEEE Computer Society Washington, DC, USA, 2001.

[KSR+06] Matthias Koenig, Wolf Spindler, Jan Rexilius, Julien Jomier, Florian Link, and Heinz-Otto Peitgen. Embedding vtk and itk into a visual programming and rapid prototyping platform. *Medical Imaging 2006: Visualization, Image-Guided Procedures, and Display*, 6141(1):61412O, 2006.

[KUDC07] Johannes Kopf, Matt Uyttendaele, Oliver Deussen, and Michael F. Cohen. Capturing and viewing gigapixel images. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, page 93, New York, NY, USA, 2007. ACM.

[KVV+04a] NK Krishnaprasad, V. Vishwanath, S. Venkataraman, AG Rao, L. Renambot, J. Leigh, AE Johnson, and B. Davis. Juxtaview-a tool for interactive visualization of large imagery on scalable tiled displays. In *Cluster Computing, 2004 IEEE International Conference on*, pages 411–420, 2004.

[KVV+04b] NK Krishnaprasad, V. Vishwanath, S. Venkataraman, AG Rao, L. Renambot, J. Leigh, AE Johnson, and B. Davis. JuxtaView-a tool for interactive visualization of large imagery on scalable tiled displays. In *Cluster Computing, 2004 IEEE International Conference on*, pages 411–420, 2004.

[LBS85] SK Lee, W. Buxton, and KC Smith. A multi-touch three dimensional touch-sensitive tablet. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 21–25. ACM New York, NY, USA, 1985.

[LC87] W.E. Lorensen and H.E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 163–169. ACM New York, NY, USA, 1987.

[Lee84] S. Lee. A fast multiple-touch-sensitive input device. *Master's thesis, University of Toronto*, 1984.

[Leh97] Roy S. Lehrle. Forensics, fakes, and failures: Pyrolysis is one part in the overall armoury. *Journal of Analytical and Applied Pyrolysis*, 40-41:3 – 19, 1997. PYROLYSIS '96.

[LM04] Eric B. Lum and Kwan-Liu Ma. Lighting transfer functions using gradient aligned sampling. In *VIS '04: Proceedings of the conference on Visualization '04*, pages 289–296, Washington, DC, USA, 2004. IEEE Computer Society.

[Mar91] K. Martinez. High resolution digital imaging of paintings: The vasari project. *Microcomputers for Information Management*, 8(4):277–83, 1991.

[MC05] Kirk Martinez and John Cupitt. Vips - a highly tuned image processing software architecture. In *ICIP (2)*, pages 574–577, 2005.

[MCSP02] K. Martinez, J. Cupitt, D. Saunders, and R. Pillay. Ten years of art imaging research. *Proceedings of the IEEE*, 90(1):28–41, 2002.

[MDH+03] A. MacEachren, X. Dai, F. Hardisty, D. Guo, and G. Lengerich. Exploring high-D spaces with multiform matrices and small multiples. In *IEEE Symposium on Information Visualization, 2003 (INFOVIS 2003); 19–21 Oct. 2003; Seattle, Washington*, pages 31–38. Citeseer, 2003.

[Mit97] J.L. Mitchell. *MPEG video compression standard*. Kluwer Academic Publishers, 1997.

[Mor98] H. Moravec. When will computer hardware match the human brain. *Journal of Evolution and Technology*, 1:1–14, 1998.

[MRB05] Shahzad Malik, Abhishek Ranjan, and Ravin Balakrishnan. Interacting with large displays from a distance with vision-tracked multi-finger gestural input. In *UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 43–52, New York, NY, USA, 2005. ACM.

[MTB03] Michael J. McGuffin, Liviu Tancau, and Ravin Balakrishnan. Using deformations for browsing volumetric data. In *VIS '03: Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, page 53, Washington, DC, USA, 2003. IEEE Computer Society.

[NR02] S. Navrud and R.C. Ready. *Valuing cultural heritage*. Elgar, 2002.

[PDMDRP08] A. Pelagotti, A. Del Mastio, A. De Rosa, and A. Piva. Multispectral imaging of paintings. *Signal Processing Magazine, IEEE*, 25(4):27–36, July 2008.

[PKS+08] Peter Peltonen, Esko Kurvinen, Antti Salovaara, Giulio Jacucci, Tommi Ilmonen, John Evans, Antti Oulasvirta, and Petri Saarikko. It's mine, don't touch!: interactions at a large multi-touch display in a city centre. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 1285–1294, New York, NY, USA, 2008. ACM.

[Ple08] L. Plesea. The design, implementation and operation of the JPL OnEarth WMS server. In *Geospatial Services and Applications for the Internet*, pages 93–109. Springer US, 2008.

[PSH97] Vladimir I. Pavlovic, Rajeev Sharma, and Thomas S. Huang. Visual interpretation of hand gestures for human-computer interaction: A review. 1997.

[PWFO01] KL PERNG, WT WANG, M. FLANAGAN, and M. OUHYOUNG. A Real-time 3D Virtual Sculpting Tool Based on Modified Marching Cubes. In *Int Conf Artif Real Telexistence*, volume 11, pages 64–72, 2001.

[RBJW01] Meredith Ringel, Henry Berg, Yuhui Jin, and Terry Winograd. Barehands: implement-free interaction with a wall-mounted display. In *CHI '01: CHI '01 extended abstracts on Human factors in computing systems*, pages 367–368, New York, NY, USA, 2001. ACM.

[Rek98] Jun Rekimoto. A multiple device approach for supporting whiteboard-based interactions. In *CHI '98: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 344–351, New York, NY, USA, 1998. ACM Press/Addison-Wesley Publishing Co.

[RJJ+06] L. Renambot, B. Jeong, R. Jagodic, A. Johnson, J. Leigh, and J. Aguilera. Collaborative visualization using high-resolution tiled displays. In *ACM CHI Workshop on Information Visualization Interaction Techniques for Collaboration Across Multiple Displays*, 2006.

[RJL05] L. Renambot, A. Johnson, and J. Leigh. Lambdavision: Building a 100 megapixel display. In *NSF CISE/CNS Infrastructure Experience Workshop, Champaign, IL*, 2005.

[RP00] M. Riesenhuber and T. Poggio. Models of object recognition. *Nature Neuroscience*, 3:1199–1204, 2000.

[Ryd] Thomas Rydell. Virtual autopsy table. https://www.tii.se/projects/autopsy.

[SBdL09a] Larry Smarr, Maxine Brown, and Cees de Laat. Editorial: Special section: Optiplanet - the optiputer global collaboratory. *Future Gener. Comput. Syst.*, 25(2):109–113, 2009.

[SBdL09b] Larry Smarr, Maxine Brown, and Cees de Laat. Special section: Optiplanet – the optiputer global collaboratory. *Future Generation Computer Systems*, 25(2):109 – 113, 2009.

[SC93] D. Saunders and J. Cupitt. Image processing at the national gallery: The vasari project. 1993.

[SGHB07] J.D. Smith, TC Graham, D. Holman, and J. Borchers. Low-cost malleable surfaces with multi-touch pressure sensitivity. In *Horizontal Interactive Human-Computer Systems, 2007. TABLETOP'07. Second Annual IEEE International Workshop on*, pages 205–208, 2007.

[SGM03] Stacey D. Scott, Karen D. Grant, and Regan L. Mandryk. System guidelines for co-located, collaborative work on a tabletop display. In *ECSCW'03: Proceedings of the eighth conference on European Conference on Computer Supported Cooperative Work*, pages 159–178, Norwell, MA, USA, 2003. Kluwer Academic Publishers.

[SHP+96] Rajeev Sharma, Thomas S. Huang, Vladimir I. Pavlovi'c, Yunxin Zhao, Zion Lo, Stephen Chu, Klaus Schulten, Andrew Dalke, Jim Phillips, Michael Zeller, and William Humphrey. Speech/gesture interface to a visual computing environment for molecular biologists. In *IEEE Computer Graphics and Applications*, pages 30–35, 1996.

[SLJM08] D. Svistula, J. Leigh, A. Johnson, and P. Morin. MagicCarpet: a high-resolution image viewer for tiled displays, 2008.

[SPS48] C. Shannon, N. Petigara, and S. Seshasai. The Mathematical Theory of Communication . *Communication, Bell System Technical Journal*, 1948.

[Sre08] M. Sreenivasan. Microsoft silverlight. 2008.

[STA] C. STANDARD. THE MPEG VIDEO COMPRESSION STANDARD.

[SVFR04] Chia Shen, Frédéric D. Vernier, Clifton Forlines, and Meredith Ringel. Diamondspin: an extensible toolkit for around-the-table interaction. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 167–174, New York, NY, USA, 2004. ACM.

[SVS+05] R. Stockli, E. Vermote, N. Saleous, R. Simmon, and D. Herring. he blue marble next generation – a true color earth dataset including seasonal dynamics from modis. *Published by the NASA Earth Observatory*, 2005.

[SW06] Bram Stolk and Paul Wielinga. Building a 100 mpixel graphics device for the optiputer. *Future Generation Computer Systems*, 22(8):972 – 975, 2006.

[SYK+05] H. Shimamoto, T. Yamashita, N. Koga, K. Mitani, M. Sugawara, F. Okano, M. Matsuoka, J. Shimura, I. Yamamoto, T. Tsukamoto, et al. An Ultrahigh-Definition Color Video Camera With 1.25-inch Optics and 8k x 4k Pixels. *SMPTE Motion Imaging Journal*, pages 3–11, 2005.

[SYS+06] D. Shirai, T. Yamaguchi, T. Shimizu, T. Murooka, and T. Fujii. 4k shd real-time video streaming system with jpeg 2000 parallel codec. In *Circuits and Systems, 2006. APCCAS 2006. IEEE Asia Pacific Conference on*, pages 1855–1858, Dec. 2006.

[TC05] James J. Thomas and Kristin A. Cook. *Illuminating the Path: The Research and Development Agenda for Visual Analytics.* National Visualization and Analytics Ctr, 2005.

[TFM96] S. Thorpe, D. Fize, and C. Marlot. Speed of processing in the human visual system. *Nature*, 381(6582):520–522, 1996.

[Tuf91] E.R. Tufte. Envisioning information. *Optometry and Vision Science*, 68(4):322, 1991.

[TWC+06] Nut Taesombut, Xinran (Ryan) Wu, Andrew A. Chien, Atul Nayak, Bridget Smith, Debi Kilb, Thomas Im, Dane Samilo, Graham Kent, and John Orcutt. Collaborative data visualization for earth sciences with the optiputer. *Future Generation Computer Systems*, 22(8):955 – 963, 2006.

[VBRR02] G. Voß, J. Behr, D. Reiners, and M. Roth. A multi-thread safe foundation for scene graphs and its extension to clusters. In *EGPGV '02: Proceedings of the Fourth Eurographics Workshop on Parallel Graphics and Visualization*, pages 33–37, Aire-la-Ville, Switzerland, Switzerland, 2002. Eurographics Association.

[VL03] H.R. Varian and P. Lyman. How much information. *University of California at Berkeley, School of Information Management & Systems (SIMS)*, 2003.

[VOT]

[WAB+05] G. Wallace, O.J. Anshus, P. Bi, H. Chen, Y. Chen, D. Clark, P. Cook, A. Finkelstein, T. Funkhouser, Anoop Gupta, M. Hibbs, K. Li, Z. Liu, Rudrajit Samanta, Rahul Sukthankar, and O. Troyanskaya. Tools and applications for large-scale display walls. *Computer Graphics and Applications, IEEE*, 25(4):24–33, 2005.

[WE98] R. Westermann and T. Ertl. Efficiently using graphics hardware in volume rendering applications. In *Proceedings of SIGGRAPH*, volume 98, pages 169–178, 1998.

[WEH01] W. Westerman, J. Elias, and A. Hedge. Multi-touch: A new tactile 2-d gesture interface for human-computer interaction. In *Proceedings of the Human Factors and Ergonomics Society 45th Annual Meeting*, volume 1, pages 632–636, Minneapolis/St. Paul, MN, 2001.

[Wil83a] Lance Williams. Pyramidal parametrics. In *SIGGRAPH '83: Proceedings of the 10th annual conference on Computer graphics and interactive techniques*, pages 1–11, New York, NY, USA, 1983. ACM.

[Wil83b] Lance Williams. Pyramidal parametrics. In *SIGGRAPH '83: Proceedings of the 10th annual conference on Computer graphics and interactive techniques*, pages 1–11, New York, NY, USA, 1983. ACM.

[Wil04] A.D. Wilson. Touchlight: an imaging touch screen and display for gesture-based interaction. In *Proceedings of the 6th international conference on Multimodal interfaces*, pages 69–76. ACM New York, NY, USA, 2004.

[WK95] Sidney W. Wang and Arie E. Kaufman. Volume sculpting. In *I3D '95: Proceedings of the 1995 symposium on Interactive 3D graphics*, pages 151–ff., New York, NY, USA, 1995. ACM.

[wms]

[Zha09] Jian-Feng Zhang. Gpu-based direct volume rendering with advanced illumination and deep attenuation shadows. *Computer-Aided Design and Computer Graphics, 2009. CAD/Graphics '09. 11th IEEE International Conference on*, pages 536 –539, 2009.