Model Guided Rendering for Medical Images

Derek Merck

A dissertation submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science.

Chapel Hill
2009

Approved by:

Stephen M. Pizer

Julian Rosenman

Russell Taylor

Edward Chaney

Marc Niethammer

ABSTRACT

Derek Merck: Model Guided Rendering for Medical Images
(Under the direction of Stephen Pizer & Julian Rosenman)

High quality 3D medical image visualization has traditionally been restricted to either particular clinical tasks that focus on easily identified or high contrast structures, such as virtual colonoscopy, or to atlas patients such as the Visible Human, which can be painstakingly micro-segmented and rendered offline. Model Guided Rendering (MGR) uses partial image segmentations as a framework for combining information from multiple data sources into a single view, which leads to a variety of methods for synthesizing high quality visualizations that require only a short setup time. Interactively presenting such scenes for particular target patients enables a variety of new clinical applications.

MGR draws information about a scene not only from the target medical image but also from segmentations and object models, from medical illustrations and solid textures, from patient photographs, from registration fields, and from other patient images or atlases with information about structures that are hidden in the base modality. These data sources are combined on a region-by-region basis to estimate context-appropriate shading models and to compose a globally useful composition (clipping) for the entire scene. Local mappings are based on segmenting a sparse set of important structures from the scene by deformable shape models with well defined volumetric coordinates, such as the discrete medial representation (m-reps). This partial segmentation provides object coordinates that can be used to guide a variety of fast techniques for oriented solid texturing, color transfer from 2D or 3D sources, volume animation, and dynamic hierarchical importance clipping.

The mgrView library computes medial-to-world and world-to-medial mappings and implements many of MGR's methods within a fast rasterize-and-blend rendering core that can render complex scenes in real time on modest hardware. Several vignette views demonstrate how MGR's unique capabilities can lead to important new comprehensions in clinical applications. These views include an interactive anatomic atlas of the head and neck, animated display of the effects of setup error or anatomic shape change on fractionated external beam radiotherapy treatment, and a pharyngoscopic augmentation that overlays planning image guidance information onto the camera view.

ACKNOWLEDGEMENTS

I would foremost like to thank my loving and patient wife, **Lisa**, and my children, **Alex**, **Vanessa**, and **Benjamin**, for their invaluable support during this long trip. My extended family, **Sherry** and **Rich Bader**, **Veronica Fagan**, **Lloyd Merck** and **Judy Foreman**, and **Art** and **Violet Lane** have all contributed hugely to this project through their constant acts of kindness.

The idea of "Netterly Rendering", which originally engendered the entire Model Guided Rendering enterprise, was born out of a series of passing conversations with **Julian Rosenman**, who has been a visionary leader and a generous patron of this project.

UNC has a wealth of faculty and staff who have been incredibly helpful to me over the last several years. **Steve Pizer** has nurtured me academically since the moment that I joined the department. My other committee members, **Ed Chaney**, **Marc Niethammer**, and **Russell Taylor** have been equally constructive and critical in forging the ideas behind this dissertation. Other UNC faculty members, notably **Mark Foskey**, **Eric Schreiber**, and **Anselmo Lastra** have been more than generous with their time and insight. The entire MIDAG group of the last six years and especially my research partner, **Ilknur Kabul**, has been both supportive and charitable.

Finally, I would particularly like to single out **Gregg Tracton**, who has mentored me and served as a constant sounding board for these ideas for the last six years. I have told Gregg many times that when I get my piece of paper, I will give half of it to him.

CONTENTS

FIGURES

EQUATIONS

PROGRAMS

1 Model Guided Rendering for Medical Images

Model Guided Rendering (MGR) is a set of methods for creating high quality, patient-specific medical visualizations for clinical procedure planning.

High quality volume rendering has so far been limited to atlas images because of the large time investment required in annotating the data. The view from VoxelMan (Pommert, et al. 2001) shown in Fig. 1 is quite impressive, but the views are not portable outside their source data, which was carefully micro-segmented voxel-by-voxel (hence the name) at a cost of over 10,000 graduate student hours. Furthermore, the renderings from this data may take from minutes to hours, which restricts the user to pre-rendered views and animations.

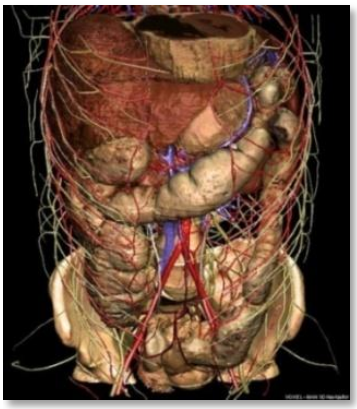

Fig. 1 Rendering from the VoxelMan project

Classical Direct Volume Rendering (DVR) for patient images, as in Fig. 2 (M. Levoy 1990), has been limited since its inception by the fact that there is insufficient information in a single image for truly high quality rendering. While medical imaging devices have become steadily more sophisticated and patient data collections have grown to encompass dozens of interrelated structural and functional images, segmentations, photographs, and intervention plans, medical visualization remains in its infancy. Indeed, patient images are still routinely viewed using the same cumbersome slice-by-slice views (Fig. 3) introduced in the 1970's when CT scanners first became widely available.

Fig. 2 Classical DVR from Levoy's original papers

Model Guided Rendering is a novel framework for medical image visualization that integrates as much patient data as possible into a high quality, patient-specific view tailored to a particular clinical task and rendered at interactive rates. MGR is based on the idea of merging information from multiple image sources on a region-by-region basis, using volumetric coordinates from a sparse set of

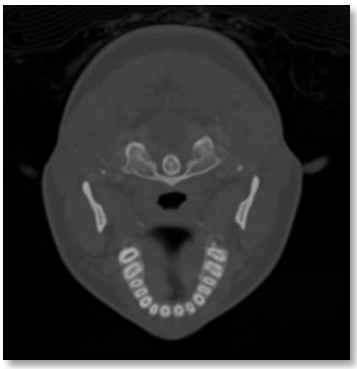

Fig. 3 Slice-wise view are still common in clinical applications.

segmentations for important anatomic structures to guide the combination.

Model Guided Rendering is being developed through the Medical Image Display and Analysis Group (MIDAG) at UNC Chapel Hill as a joint project between the Departments of Radiation Oncology and Computer Science. UNC Hospital's Radiation Oncology clinic has provided many of the driving problems that MGR immediately addresses.

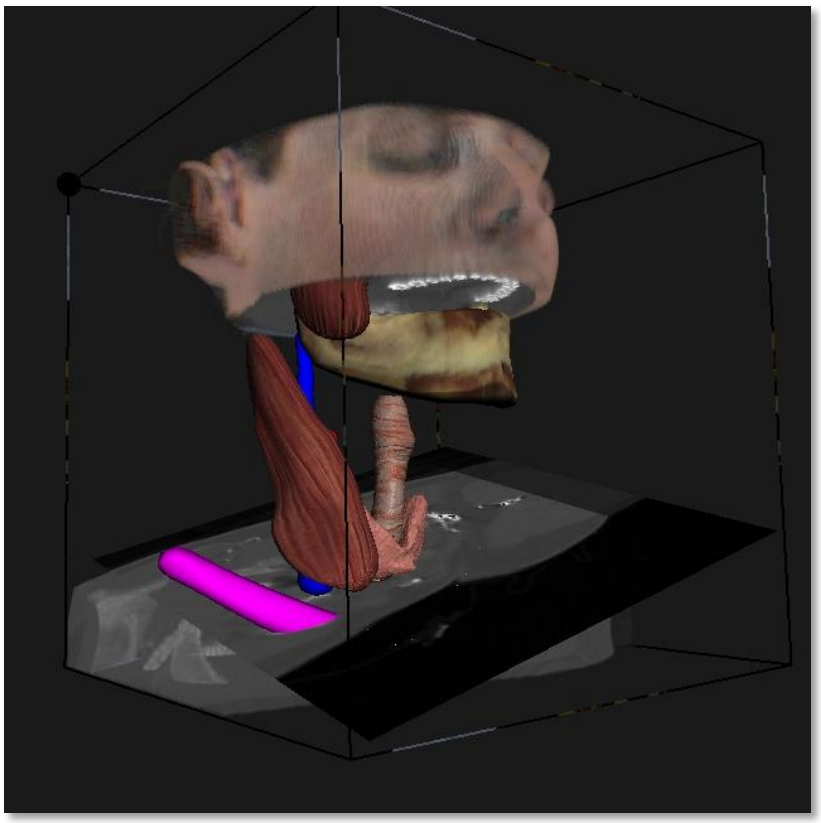

Fig. 4 Model Guided Rendering of a target patient using information taken from the CT volume, a 3D color atlas, a patient photo, and several synthetic textures mapped onto various regions

1.1.1 Clinical Applications of MGR

Detailed Anatomic Understanding

Artists and anatomists have worked for centuries on improving 2D anatomical renderings. For example, Frank Netter's anatomic textbooks are full of compelling examples of how data can be 'processed' into useful 3D scenes by the human mind, such as the head and neck illustration shown in Fig. 5 (Netter 2006). Netter's illustrations are deemed to be so useful that they are commonly taken in the operating room for reference during surgery despite two huge

drawbacks: first, they are not of this particular surgical patient's anatomy and second, they are not interactive. A major goal of Model Guided Rendering is to lift the atlas-only restriction on high-quality medical rendering and generate similarly high quality 3D "Netterly Renderings" for individual target patients.

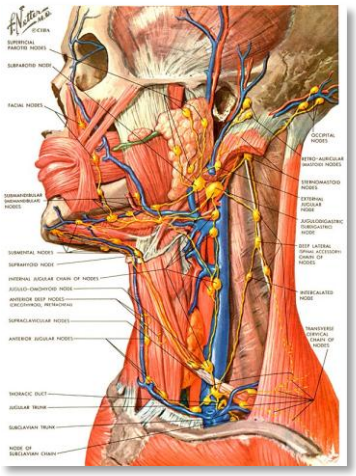

Fig. 5 Anatomic illustration from Netter with intuitive shading and composition.

Multi-Source Data Visualization for Clinical Treatment Planning

Combining data in smart ways is vital to modern medicine. Clinical planning, diagnosis, and evaluation can be improved by looking at data from multiple sources such as anatomic and functional fusion images for diagnosis, or longitudinal imaging taken for adaptive radiotherapy (ART), or by comparing a patient's traits to various normal and abnormal distributions. MGR provides a framework for bringing the same kinds of multi-source improvements to a broad set of potential clinical tasks, such as external beam radiotherapy planning or virtual endoscopy.

In general, image guided clinical treatment planning revolves around understanding relationships between anatomic features in space and across time. MGR provides a number of tools for displaying longitudinal shape change as for ART, modality relationships as for image guided biopsy, and relationships hidden or deep features in relation to observed features as for patient setup or forensic analysis.

For surgical planning, for example, major blood vessels and nerves can be highlighted in ways that make them avoidable, whereas for radiation treatment planning, those structures are radioresistant and are usually ignored, but radiosensitive structures must be easily identified.

Fig. 6 Virtual colonoscopy is successful because it reduces 3D data to a few important surfaces.

1.1.2 Model Guided Rendering

Medical visualization technology has been relatively dormant since the brief explosion of interest in volume rendering in the early 90s. Since then, 3D rendering for medical images has found niche applications where it can be quite effective, such as virtual colonoscopy (Fig. 6). But rendering for general applications has never been able to adequately portray complex anatomy because it is

computed on a *voxel-by-voxel* basis. That is, classic DVR relies entirely on surface estimates from the local gray-scale value alone. This is equivalent to relying on grayscale value alone to segment an image, a process that is known to require additional or "*a priori*" information to be generally successful. Extending volume rendering by adding additional information through a "transfer function" tends to be an *ad hoc* and non-generalizable method to solve this problem.

On the other hand, high-quality medical rendering systems such as VoxelMan require a huge investment in "scene priors", that is, information about features and relationships in the scene that allow the renderer to create a focused, understandable image. Typically there is not enough time to carefully hand-edit an expansive set of scene priors for individual patient images. However, modern patient data collections can provide us with an abundant source of scene priors. Table 1 shows a few examples.

PATIENT AND ATLAS SOURCES	PROVIDE SCENE PRIORS FOR
Partial Segmentations	Object type, orientation
Multi-modal scans	Likely structures that are invisible in a particular modality
Longitudinal and related scans	Shape relationships over time or over populations
Shape and intensity statistics	Normal features
Patient photography, thermography, endoscopy	Color, texture on and near the surface
2D and 3D color atlases	Likely color, texture of normal anatomy
Registration fields	Motion
Dose distributions	Dose to at risk anatomy

Table 1 Some example uses of various image sources in a patient data collection.

Model Guided Rendering is a framework for quickly bringing a large number of scene priors to bear on a patient-specific rendering by combining information from multiple sources. From the combined scene information, MGR derives a notion of what the scene should look like, what is shown, and what is important about it for a particular application. Then it presents and highlights the important structures where the information is clear, but where the information is insufficient or uncertain, MGR turns to non-patient-specific sources to make educated guesses about how that region is likely to appear.

The sources may all be in different spaces, so methods from image analysis are leveraged to assign explicit relationships between 3D regions in various sources. MGR relates images to each other in two ways: by volume filling registration fields or according to regional non-linear transforms derived from medial object coordinates.

Externally computed registration fields such as are used in (Davis, et al. 2004) are a common tool for studying longitudinal shape changes in ART. Such deformation fields provide a key input to MGR's scene composition methods, both for ART-relevant animations and by tailoring deformations to tasks such as describing how anatomic structures might move under various forces so that they can be "retracted" rather than simply clipped away when they occlude other important features.

M-reps (Pizer, et al. 2008) provide a natural basis for the region-by-region mappings required by MGR's appearance methods. They provide both intuitive volumetric coordinates (i.e., (*u,v,t*) = (along, around, through)) and volumetric correspondence across a population. Additionally, semi-automatic m-rep based segmentation is commonly applied in our clinical pipeline (Fig. 7). M-reps are used to identify and parameterize a few important anatomic structures in the scene with a minimum of manual editing, which defines region-by-region mappings across the multiple disparate image sources. **Medial object-coordinates allow the renderer to work on an object-by-object basis rather than a voxel-by-voxel basis.** This is a key insight for approaching the goal of high-quality rendering without exhaustive segmentation.

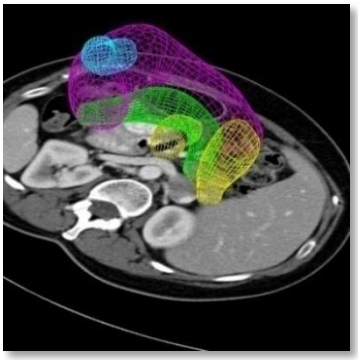

Fig. 7 Implied surfaces of m-reps fit to anatomic structures in the abdomen.

1.1.3 Scene Design Technologies

Because deep anatomy is very complex, a 3D visualization will only be useful if structures less significant to the current task can be suppressed while those structures that aid understanding are emphasized and clearly labeled. In the context of a 3D patient image, this problem can be considered as two ways of relating apparent anatomy to occult anatomy, i.e., those features that are either hidden by modality or occluded from view in space or time. Addressing such hidden features are two domains where artistic illustration particularly excels relative to computed visualization.

1. Anatomy is visually labeled – structures are colored and textured intuitively so that the viewer can understand what they are seeing in the scene. Monochrome 3D imaging in particular inherently loses this visual information.
2. Shapes and relationships between anatomic structures and regions that are relevant to the task at hand are exposed so that the viewer can see what they need to see in the scene. Classical DVR has very few tools to simulate this.

In the classic paper on computer assisted technical illustration (Seligmann and Feiner 1989), these two concepts are called respectively **design of appearance** and **design of composition**. For the purposes of MGR, intuitive appearance and clear composition is the very definition of the term "high quality rendering" used earlier. MGR's key technologies can be broadly divided into methods for assigning scene appearance and methods for scene composition.

MGR addresses appearance with algorithms that leverage multiple data sources to shade regions for understandability. **MGR's key appearance technologies are scene catalogs, solid texture mapping, 3D color transfer and texture synthesis, and 2D color transfer.**

The scene must be composed to focus attention on important structures for this patient, for this problem (Fig. 8). **MGR's key composition technologies work either by clipping the volume to eliminate objects that obscure the view or by deforming the volume to move occluding structures out of the way.** Netter uses a combination of the two approaches, sometimes removing occluders completely, sometimes showing a cut and peeling a surface away to keep context. Volumetric deformation further extends to the general problem of volumetric animation or volume morphing, which can be used to indicate shape changes over time or shape variance relative to population statistics.

Fig. 8 Photoshopped example of how scene composition addresses the issue of seeing internal structures while preserving context.

1.1.4 Key Appearance Technologies

Scene Maps

MGR is based on the idea of object-coordinate driven shading, but the rendering engine still works in world or voxel coordinates. Medial models are not particularly amenable to doing such transformations

directly, but MGR Uses m-reps only implicitly – to generate world-to-object (x2u) and object-to-world (u2x) maps from corresponding surface and medial positions. These maps contain at each sample the combined information from any image analysis preprocessing. Each voxel is assigned a variety of feature channels such as object label, model coordinates, local directions, and local statistical variation from atlas shapes and intensities. Simple maps representing a single object related to a single image can be combined into a comprehensive map for all the objects in a scene, called a "scene catalog". Even a relatively simple scene catalog can provide the rendering engine with much more information than the raw data alone.

These scene catalogs are organized for immediate reference during rendering and provide a fast method for moving back and forth between whatever coordinate systems – object, world, or screen – are appropriate for a particular part of the rendering pipeline. Computing and using such maps dynamically is a key component of many parts of MGR's shading algorithms.

High quality Volume and Surface Rendering

Previously, volume rendering has only taken information from a single source, the patient image. It then attempts to visually imply anatomic structures by assigning colors based on increasingly complex local transfer functions. (See (Kindlmann, et al. 2003) for example) The simplest and most intuitive means of displaying anatomic features, by applying intuitive anatomically based textures, is incredibly difficult to implement in the classical framework.

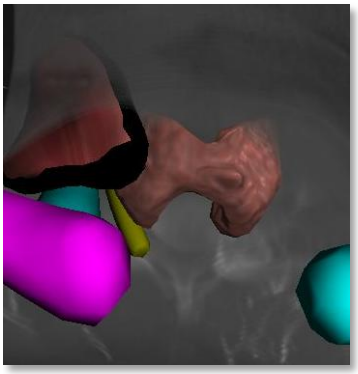

Fig. 9 Synthetic 2D and 3D textures mapped onto target regions in a patient image.

MGR is a suite of methods that can integrate data from many sources, such as patient images from different times or devices, segmentations or registration fields, color atlas and anatomic textures, to synthesize a single high quality view. The scene catalog provides a mapping between the underlying anatomic structures in a scene and the various possible sources for assigning regional textures. Given the parameterization discussed, simple oriented solid texture mapping from 3D or 2D textures into 3D regions or 2D surfaces follows naturally: library textures can be assigned according to object label, and then oriented according to local object-coordinate derivatives. These same orientations can be used to enable sophisticated lighting from normal maps for solid textures.

Color Mapping from 3D Images

It is unlikely that MGR will ever get a patient-specific empirical color volume as input because a clinician is unlikely to actually section their patient. However, normal anatomy can be approximately visually labeled in a target patient by collecting information from a color atlas such as the Visible Human and then mapping into the patient space using corresponding regional coordinates in both the source and target image (Fig. 10). This amounts to rendering certain regions not from the grayscale data directly, but from an altered version of some empirical or pre-rendered atlas color volumes that has been deformed to fit this patient.

3D color mapping is the primary candidate for fast change of coordinates using the scene catalog. Given a scene catalog with forward and backwards transforms, this becomes an extremely local and parallelizable problem. The basic algorithm for shading a pixel at target($\underline{X}_{tgt}$) is as follows.

1. Convert $\underline{X}_{tgt}$ to model coordinate $\underline{U}$
2. Convert $\underline{U}$ to $\underline{X}_{src}$ in the source image
3. Apply the color from source($\underline{X}_{src}$)

This concept extends naturally to the idea of mapping any values from one spatial volume to another, for example, pulling information from a 3D functional image or statistical distribution and then making shading decisions based on both the base modality and any additional information. This supports visualizations such as highlighting voxels corresponding to regions with a high likelihood of pathology in fMRI.

The concept further extends to the idea of including visual estimates for not for just color but for anatomic structures that cannot be seen in the underlying digital image. For example, although nerves, smaller blood vessels, or lymph levels are invisible in CT, having a visual estimate of their position can be useful to a clinician who otherwise has to make position estimates based only on collections of 2D slices. Models of these occult structures, based on population statistics, can be included in model-based rendering, albeit in such a way that they are clearly identified as structures that are only likely to be present but not guaranteed to be so.

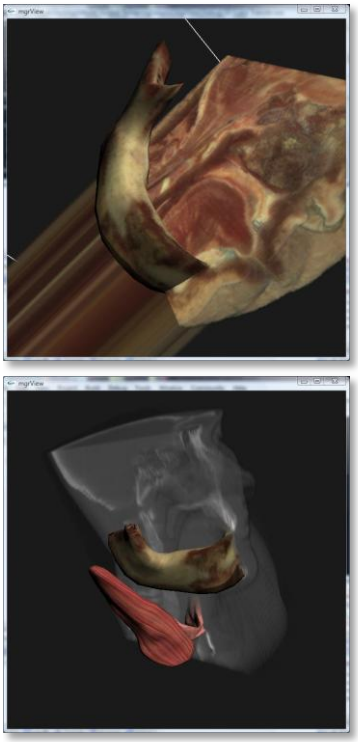

Fig. 10 Top, color atlas mandible. Bottom, atlas colors transferred to a target patient according to object-coordinates.

Supporting Guided Texture Synthesis

Atlas-based rendering as described above can be thought of as a very simple example of a probability-based rendering: atlas textures show a "most likely" appearance. Beyond color mapping from normal atlas images, guided texture synthesis offers a considerably more sophisticated method for probability-based rendering. Guided texture synthesis enables normal as well as abnormal and non-atlas structures to be rendered with textures appropriate to a target patient's condition as determined from prior clinical knowledge or statistical estimates. That is, lesions obvious in the gray image might be rendered to look pathological, and regions with clinically identified cancers might look cancerous. In general, unexpected structures or structures with abnormal data values can rendered to reflect that. Relevant image interpretation, such as object label, local directions (i.e., $\nabla U = (du,dv,dt)$ at each voxel), and variation from normality can be exported as feature channels directly to solid texture synthesis modules and the resulting space filling oriented texture appropriate for this patient can be incorporated seamlessly into the view. MGR currently works with the exemplar based algorithm described in (Kabul, et al. 2010) but could also work with procedural methods.

Color Mapping from 2D Images

Mapping patient photos into the rendering, as shown in Fig. 11, allows the user to visually relate surface and deep features. The photos may come from visible light, thermography, or another modality.

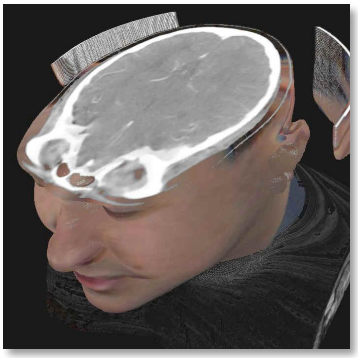

Fig. 11 An image of the author mapped onto a target patient

The photographed surface is identified in the volume image using model coordinates. Then color information from the corresponding photograph is mapped onto those voxels using a projective or cylindrical transform depending on the camera arrangement. Cylindrical maps can be collected using specialized hardware or synthesized from multiple planar camera images. Multiple planar camera images can also be selected individually by comparing the angle between the view direction and the surface normal, by time of capture, or by both criteria.

Color mapping from 2D images has applications in diagnosis, planning, and procedure setup. Thermographic images could be used for vein based ("near surface feature") patient setup. Interpolating across serial patient photos using the volume data as an alignment scaffold

can be used to track surface features such as changing lesions or skin reactions to radiotherapy. Using uncalibrated photos of the patient such as might be taken for charting would require an algorithm for aligning 2D and 3D landmarks, but the problem then reduces to the calibrated planar camera case. Using images taken from tracked cameras during endoscopic procedures could provide an additional source of interior color information and serve as the basis for novel views based on endoscopic data but relieved of the burdensomely narrow field of view.

1.1.5 Key Scene Composition Technologies

Importance Clipping

With most classical DVR, the user can rarely see what is actually important in the scene. The standard method of surface finding by examining local gradient magnitude is a poor proxy for importance. This has recently been addressed by using explicit data segmentation. "Importance rendering" is the term used by (Viola, Kanitsar and Groller 2004) to describe a kind of object-based region-of-interest (ROI) clipping, where voxel opacity is computed according to an "importance" factor determined by voxel-wise pre-segmentation. In MGR a similar function is implemented to "disocclude" importance ranked regions, so that relevant features can always be seen without giving up local context information. Fig. 12 left shows a standard pseudo-colored region of interest in the male pelvis. The objects-of-interest, the bladder, prostate, and rectum are completely hidden by the intervening tissue. On the right is an importance clipped scene, where those unimportant intervening voxels have been suppressed.

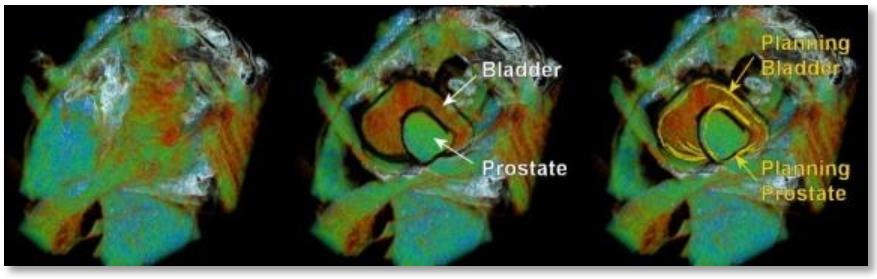

Fig. 12 Left, a standard volume rendering of a region in the pelvis, middle, the same region with the data occluding the prostate and bladder dynamically clipped away. Right, the same view with the planning position of the objects overlayed as contours.

Whereas Viola relies on exhaustively pre-segmented data, in MGR importance regions can be assigned dynamically not only to anatomic shapes, but to suspicious regions according to probability

distributions, or to dynamic and interactive shapes such as an "importance flashlight". MGR's algorithms also extend to hierarchically ranked objects.

Furthermore, while volume ray-tracing such as Viola uses can render at interactive rates only on specialized hardware, MGR's implementation of the importance clipping algorithm works at interactive rates on even modest hardware. The implementation is related to methods for computing "shadow volumes" for tiled surfaces, wherein the dark-side of a closed surface is extruded away to infinity along the contour edges and then the stencil buffer is used to track which screen fragments are shadowed according to that particular light source and that particular object. MGR's method is called "importance stenciling" and relies on a similar idea. Important objects are extruded *towards* the camera and this "importance shadow" is stenciled against every voxel as it is projected onto the screen.

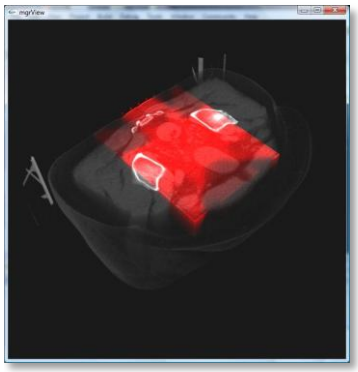

Fig. 13 Interpolating between two daily patient images according to a registration field gives a smooth volumetric animation of anatomic change relative to a fixed dose distribution (red overlay).

Volumetric Animation

Deformation fields can be used to drive surface and volumetric interpolations for animation. Visualization of longitudinal anatomic change is particularly interesting in the context of ART for showing how the daily changes in anatomy will affect the expected dose distribution (Fig. 13). Deformation fields can also serve as models of how to "retract" occluding anatomic structures as a different, more organic kind of importance rendering. Few methods have been proposed for 4D volume morphing, so this is an interesting and unexplored research area in its own right.

1.1.6 Implementation

Beyond appearance and composition there is an implicit third constraint on MGR: **speed**. While none of the key technologies discussed depend explicitly on a particular DVR framework, our C++/OpenGL library of the core MGR functions, **mgrView**, is based on rasterize-and-blend DVR (Drebin, Carpenter and Hanrahan 1988). Rasterize-and-blend DVR allows the complex, non-linear mappings such as the world-to-object transforms to be moved onto the graphics accelerator, where they can be done very quickly. Rasterize-and-blend DVR has traditionally been limited in quality by the fixed

functionality graphics pipeline, but recent improvements in volume texture representations and programmable shaders have alleviated those restrictions. mgrView can achieve interactive rates for complex volumetric scenes on laptops and inexpensive workstations.

mgrView is wrapped in an extensible GLUT/GLUI (Radamacher, Stewart and Baxter 2006) user interface, but it could be embedded in other windowing environments such as UNC's in-house clinical radiotherapy planning tool Plan-UNC (UNC Hospital Department of Radiation Oncology 2007). mgrView also includes a variety of default routines for frame grabs and file readers and writers, but the code is not tied directly to any particular data representations. mgrView has also been designed to be flexible with respect to future technology extensions such as virtual or augmented reality for online image guided procedures like image guided biopsy. mgrView also includes a comprehensive user guide detailing both its usage and its algorithmic implementations.

The next page provides a short introduction to mgrView's programming format for interested engineering-oriented readers of this dissertation. It can safely be skipped by other readers.

1.1.7 Thesis

Image segmentation via **medial shapes provides an effective basis for guiding context-appropriate shading** in 3D medial image display by supporting regional color mapping from library or synthesized **solid textures**, **cross-modal images**, and **atlas data sources**.

Precomputing a global "**scene catalog**" that collects multiple local medial-to-world and world-to-medial transforms enables these techniques in an **interactive object-order volume rendering framework**.

This framework additionally extends other perception-enhancing effects such as **importance rendering** and **volume deformation** to dynamic scenes.

INTERMEZZO: MGRVIEW

mgrView programs are characterized by establishing relationships between images and models, which directs the rendering engine's appearance and composition algorithms. In Program 1 three kinds of objects are loaded, 2D images (textures), a surface file, and a volumetric medical image. These objects are attached to one another with particular channel labels, such as "source_im" or "color_im". Layers are derived from the surface object and automatically attached as children. The surface and volume objects are automatically attached as children to the root world object, which includes them automatically when the window makes the call to world->glRender(). Derived textures such as the duodenum uvt coordinates and the volume data gradient are automatically loaded and attached as sub-objects if they are present in the correct directories, or they are computed and cached if they do not exist.

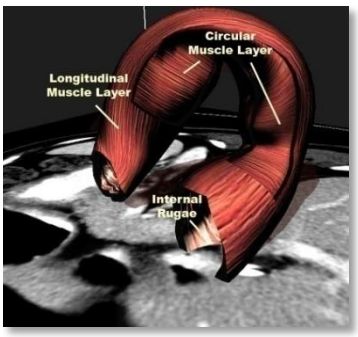

Fig. 14 Labeled rendering of the duodenum from mgrView's simple application demo program

A simple example application is shown in Program 1 and the resulting display is shown in Fig. 14. An appendix walks through implementing an entire project using and extending mgrView.

```
// Sample mgrView program
#include "../common/mgr.h"
mgrWindow* mgrw;
char* MGR_DATA_PATH = "../../data/";  char* MGR_PROJECT_DATA_SUBDIR =
"abdomen/";
int main(int argc, char* argv[]){
   mgrRenderable world;
   mgrw = new mgrWindow( argc, argv, &world );
   // -- Load textures --
   mgrImage2 muscle = mgrImage2( "muscle2.tga" );
   mgrImage2 rugae = mgrImage2( "rugae.tga" );
   // -- Setup the surface object --
   mgrSurface duodenum = mgrSurface( "duodenum.pseudotube.l2.byu" );
      duodenum.AttachImage(&muscle, MGR_COLOR_IM0);
      duodenum.SetShader(MGR_SIMPLE_TEX2_SURF_SHADER );
      duodenum.clip_end_caps = true;
   mgrSurface d1 = duodenum.AddLayer(0.9 );
      d1.FlipTransform( MGR_COLOR_IM0, Y_AXIS );  // Rotate texture
   mgrSurface d2 = duodenum.AddLayer(0.8 );
      d2.AttachImage(&rugae, MGR_COLOR_IM0 );
   // -- Setup a volume data object --
   mgrImage3 gray = mgrImage3( "3301.hires.raw", 512, 512, 64 );
   mgrVolume v = mgrVolume( &gray );
      v.images[MGR_SOURCE_IM0]->iw.set( 0.55, 0.3 ); // Intensity window
   mgrw->glStart();                                  // Start rendering
   return 0;}
```

Program 1 Sample application code invoking mgrLib to load a gray volume and surface object. The rendering and default UI is shown in Fig. 14.

1.1.8 Claims

This dissertation contributes the following novel **methodologies** for producing high-quality volume visualizations using segmentations of a sparse set of important anatomic structures to combine information from multiple image sources:

1. Method for using medial coordinates to guide context-appropriate shading in medical images by regional color mapping from several different kinds of data sources
   1.1. Method for mapping and lighting library or patient-specific synthetic solid textures
   1.2. Method for mapping from 2D data sources such as patient photographs
   1.3. Method for mapping from 3D data sources such as cross-modal images or atlas data sources

2. Method for generating such renderings at interactive rates on relatively modest hardware by precomputing a "scene catalog" data structure and manipulating it in an object-order rendering framework
   2.1. Algorithms for computing world-to-medial ("x2u") and medial-to-world ("u2x") maps from a set of segmentations by medial shapes and a data structure for collecting these mappings together
   2.2. Algorithms for using the scene catalog in various ways through programmable shader hardware to do the mappings described in (1)

3. Refactored versions of important state of the art volume rendering methods such as importance rendering and volume deformation that allow these techniques to be applied in dynamic scenes
   3.1. Object-order implementations for global and local volume deformation and for importance rendering based on ranked surfaces

1.1.9 Outline

Supporting discussion for these claims is divided according to the following chapters and sections.

Chapter 1: Overview of Model Guided Rendering

Chapter 2: The Medical Imaging Pipeline

2.1. Review of 3D medical image acquisition
2.2. Review of 3D medical image analysis, including deformable registration and medial coordinate systems
2.3. Review of classical 3D medical image visualization, including surface extraction, image-order and object-order direct volume rendering (DVR), and transfer functions

Chapter 3: Model Guided Appearance for Medical Volume Rendering

3.1. Creating scene catalogs from medial representations
3.2. Object-coordinate based solid textures and dynamic lighting
3.3. Color mapping from 2D patient photos
3.4. Color mapping from empirical or synthetic solid atlas textures

Chapter 4: Model Guided Composition for Medical Volume Rendering

4.1. Fast volumetric animation
4.2. Fast importance clipping based on importance shadows
4.3. Model-coordinate based surface windows

Chapter 5: Bringing MGR to the Clinic

5.1. MGR's potential role in medical image applications
5.2. Segmentation, planning, and patient setup in radiotherapy (and appendix describing how to implement a project with mgrView)
5.3. Augmented endoscopic guidance

Chapter 6: Conclusions & Future Work

6.1. Review of thesis and claims
6.2. Directions for future work, including advanced display and interactivity

Appendix: Implementing the Planning Under Error View Using mgrView

The mgrView software tool is frequently used to demonstrate techniques throughout this dissertation. The examples shown in chapter 2 are all demonstrations of known techniques implemented in this framework. The regional color mapping methods developed in chapter 3 are novel methodology and implementation. Chapter 4 is concerned with extending known volume rendering methods that are currently restricted to static scenes to interactive or dynamic scenes. Chapter 5 presents a series of project vignettes that demonstrate how MGR methods might be applied to clinical problems. The MGR components used in each project are clearly called out. The appendix at the end of the dissertation walks through the programming required to build one of the sample vignettes in Chapter 5 using mgrView.

2 The Medical Imaging Pipeline

Medical images begin with physical devices and reconstruction, which produce 3D data. This data can be visualized or analyzed as suited to various clinical applications. This chapter reviews the basic pipeline for applying medical imaging technology in the clinic shown in Fig. 15. It is divided up into three parts, each focused on one section of the pipeline.

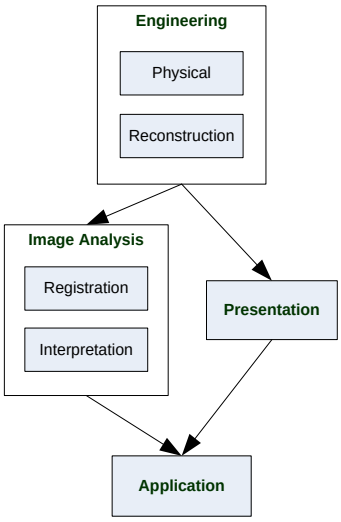

Fig. 15 This chapter reviews the medical imaging pipeline.

1. **Sources of Medical Images** reviews the principles of the **Engineering** layer, where medical images are actually produced. Energy is passed through the patient and a signal is collected. A variety of hardware devices work across the energy spectrum and can collect both structural and functional information. The signal is then processed into an image of the underlying geometry and filtered to reduce artifacts.
2. The **Image Analysis** layer, described in the section **Interpreting Medical Images**, serves to relate images to other images and to relate image regions to structures. Relating images to images is called "image registration" and relating image regions to structure is called "image interpretation" in this text. The regional model that MGR uses to relate regions across images is based on image segmentation using the discrete medial representation ("m-reps").
3. The section on **Data-driven Medical Visualization** reviews what I call the naïve or data-driven **Presentation** layer. Visualization based on independent images provides a direct and usually simple view of the data to the clinician. Classical volume rendering is based on casting rays through the image, then applying a "transfer" function to assign a false color to each voxel. Classical volume rendering forms the basis of MGR's rendering algorithms.

While automatic image analysis has become steadily more sophisticated, the results can be cryptically difficult to interpret. The goal of Model Guided Rendering is to create informed visualizations, that is, to reorganize the pipeline so that the presentation layer becomes dependent on the analysis layer.

2.1 Sources of Medical Images

This section gives a brief overview of the physical and reconstruction layers of the medical imaging pipeline. The main topics include the following.

1. **Computed Tomography** is the most important modality for radiotherapy planning, and MGR has been conceived largely to support it. This topic reviews x-ray radiation, sampling and reconstruction, and sources of artifacts.
2. **Other 3D Imaging Modalities** such as nuclear medicine imaging, magnetic resonance imaging (MRI), and ultrasound (U/S) are briefly discussed at the end of the section. MGR is designed to support rendering from multiple data sources, and while those sources are usually serial CT or CT + color textures, the same principles apply when mapping from other modalities into the rendering space. These other modalities have various clinical advantages over radiographs and CT with respect to particular future visualizations and applications.

A few examples of using mgrView to load and create simple views are presented along with the discussions of the various modalities. However, mgrView's data representation for images is detailed in the later chapter on working with mgrView.

2.1.1 Computed Tomography

Computed tomography (CT) is a widely adopted imaging modality with many clinical applications from diagnosis to procedure planning. Because it is a 3D modality, data can be presented in a variety of displays such as axial, sagittal, or coronal slices, off-axis cut planes, or even simulation of a standard x-ray projection image (called a "radiograph"). Slice-by-slice views eliminate the saturation and occlusion common in analog radiographic images and can make structures with difficult to understand 3D shapes, such as complex fractures, relatively easier to interpret. For example, in the analog chest radiograph shown in Fig. 16, the lungs are severely obscured by the ribs, making them difficult to understand. Digital images such as CT or digital radiographs can also be windowed to expose 1% density differences,

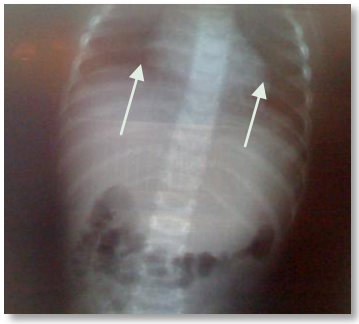

Fig. 16 Chest x-ray of the author's son at one year of age. The lungs (indicated) are extremely obscured by the ribs both in front of and behind them.

which provides relatively higher contrast resolution then analog films. Moreover, digital data can be easily post-processed at a variety of levels, from edge enhancement to data labeling (as described in the next section on image interpretation). Modern fast CT scanners can create time-varying or "4D" images, which are very valuable for studying motion in structures such as the heart or lungs.

In MGR's target domain of radiotherapy planning, the CT image provides a physically and geometrically accurate basis for therapy planning. Magnetic resonance imaging (MRI), for example, has greater contrast for pathologies, which is useful for diagnosis, but it gives no information about x-ray attenuation and suffers from innate field biases that can create geometrically incorrect images, limiting its uses for planning.

INTERMEZZO: LOADING 3D DATA INTO MGRVIEW

Many of the example images produced throughout this dissertation were generated using mgrView's built-in functionality. This intermezzo shows a short C++ mgrView program used to load a CT image generated by UNC's radiotherapy planning system "Plan UNC" (PLUNC), window it for soft tissue resolution, and display it. The code shown in Program 2 produces the results shown in Fig. 17. Note that only four lines of the program are actually scene dependent: the file is loaded, and a new volume object is instantiated and attached to the ui and the rendering root. The rest of the code simply sets up the project.

```
// Sample mgrView program to load an image
#include "../common/mgr.h"
mgrWindow* mgrw;
char* MGR_DATA_PATH = "../../data/";
char* MGR_PROJECT_DATA_SUBDIR = "pelvis/";
int main(int argc, char* argv[]) {
    mgrRenderable world;
    mgrw = new mgrWindow( argc, argv, &world );
    // -- Scene dependent code --
    mgrImage3 gray = mgrImage3( "3106.gray.pim",
        512, 512, 81, vec3( 0.098, 0.098, 0.3 ) };
    gray.iw.set( 0.1, 0.3 );
    mgrVolume v = mgrVolume( &gray );
    mgrw->glStart(); return 0;}
```

Program 2 A simple mgrView program to to load and display a raw CT image shown in Fig. 17.

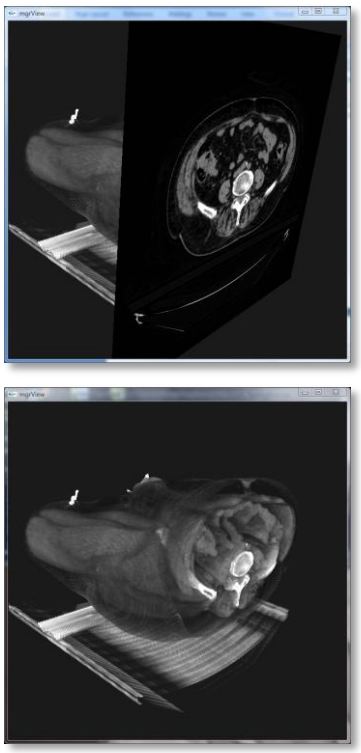

Fig. 17 A default view of the male pelvis CT image loaded using the mgrView script shown in Program 2 with and without a reference slice.

X-Ray Radiation & CT Reconstruction

X-rays are high energy photons than can penetrate solid objects. Different types of material attenuate x-ray radiation at different rates, so measuring the amount of x-ray radiation that passes through a solid object shows the radio-density shadows of structures inside of the object. Wilhelm Conrad Röntgen (Röntgen 1896) is considered to be the first person to discover x-ray radiation and demonstrate its potential medical applications (Fig. 18).

Fig. 18 Radiograph of the hand of Röntgen's wife from the late 19th century.

Projection x-ray images, called "radiographs", are a common diagnostic tool in medicine. In projection radiography, the subject is placed between an x-ray source and a film or digital sampling plate. X-rays passing through the subject are attenuated more in dense material such as bone, so the collected image is less exposed where those regions project onto the plate than it is where muscle or fat tissue regions are projected. Since most radiographs are viewed as "negative images", areas of low exposure become the brightest features in the image. As with all shadows, occlusion problems happen when more radio-dense objects such as bone obscure other structures both in front of and behind them (Fig. 16).

Computed tomography is an attempt to address obscuration and occlusion issues with projection x-rays by reconstructing the entire 3D interior of the subject. The x-ray attenuation factor of different materials is characterized by a density-weighted attenuation coefficient called μ. The calcium in bone has a very high μ, whereas air has a relatively low μ. X-ray radiation attenuates as a function of its initial energy and μ of the material that it is passing through according to Eqn. 1. Sampling the total attenuation of a particular x-ray energy at many angles and offsets about a single plane through the subject produces a large linear system that can be solved to recover the interior 2D array of attenuation factors. Repeating this process for many planes produces a 3D array of attenuation factors.

$$I_t = I_0 \mathrm{e}^{-\int \mu_s ds}$$

Eqn. 1 Formula for x-ray attenuation through tissue with local attenuation μ along a line integral parameterized by S

The total attenuation of the x-ray radiation along a single path is the exponentiation of a line integral of individual attenuation factors along the path. In the logarithmic space of this function, the total attenuation along a path is a linear combination of the attenuation factors at each sample along the path. This arrangement of the data as linear combinations producing sums at various angles and offsets is called the

Radon transform of the data. The goal of CT reconstruction is to compute the inverse Radon transform of the system. In filtered back projection, each output profile is filtered according to the amount of blur known to be introduced by the detectors (the "point spread function") and by the smearing step that follows, and then the result is "smeared" back along the original sampling line. The composite output from smearing all of the profiles is the reconstruction of the plane. Fig. 19 shows a single-slice example of forward and inverse Radon transforms applied to synthetic data[1].

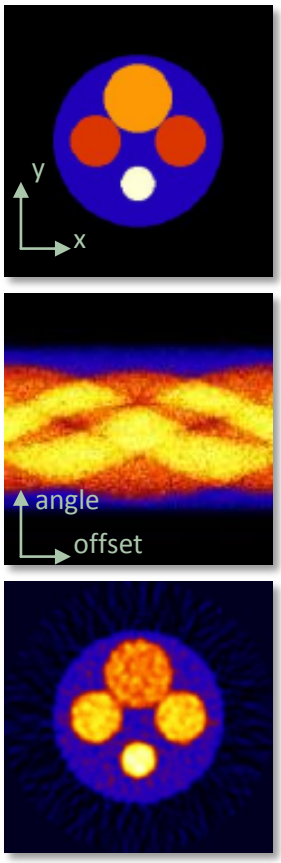

Fig. 19 Top, source image with color representing an intensity at each spatial position. Middle, Radon transformed data with color as the total intensity along a profile with a given angle and offset. Bottom, the image reconstructed by applying the inverse Radon transform to the transformed data. (Images from http://www.physics.ubc.ca/~mirg/home/tutorial/fbp_recon.html)

The original work on reconstruction from line integrals was developed in the 1960's by (Cormack 1964), based on much earlier work from (Radon 1917). In 1972 Godfrey Hounsfield created the first single-slice CT. Hounsfield's scanner (Fig. 20) took several hours to collect data and several *days* to do the reconstruction. CT machines became widely available in the 1970s. Recent advances have focused on resolution, speed, and gating for 4D motion images.

Spatial Resolution

CTs are digital systems that use analog-to-digital converters and produce sampled data, so they are subject to additional constraints on resolution and dynamic range. Modern CT scanners typically produce images on a regularly sampled grid with 512 x 512 x ~100 samples. For the abdominal, pelvic, and head and neck scans which form the basis of the later MGR case studies, the in-plane field of view (fov) is ~50cm and can span 30cm or more in length; this leads to a grid spacing of approximately 1mm x 1mm in-plane with 3mm planes (pitch). For head/brain scans, the field of view can be much smaller, so the spacing can be substantially denser.

CT is considered a potentially harmful procedure because it exposes the patient to a relatively small but non-negligible dose of the same kind of damaging radiation as is used in external beam radiotherapy. The more exposure, the higher the spatial resolution that can be achieved, so balancing exposure with medical needs is a serious issue.

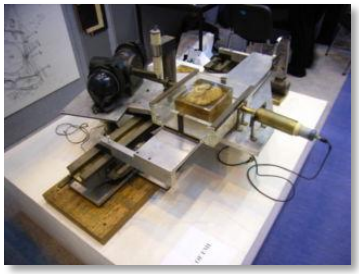

Fig. 20 Hounsfield's original prototype CT scanner (from Wikipedia).

[1] Using Matlab's "radon" and "iradon" functions. Matlab is a matrix math program developed by The Mathworks (www.themathworks.com). Because 3D images can be understood as 3D arrays, Matlab is well suited to manipulating such data structures. Simple-to-interpret Matlab functions are pointed out as examples in relationship to several topics throughout this dissertation.

The individual volume samples of any 3D image are called voxels ('volume element', as a pixel is a 'picture element'). If the sampling grid has cubic voxels, the voxels are called *isotropic*; otherwise they are *anisotropic* (Fig. 21). In the examples throughout this text, the in-plane dimensions are referenced as 'x' and 'y' and the out of plane dimension as 'z', or as **X** in general for an (xyz) world-space coordinate vector. The voxel indices are referenced as 'i', 'j', and 'k' for rows (x), columns (y), and slices (z) or **J** for an (ijk) voxel-space coordinate vector.

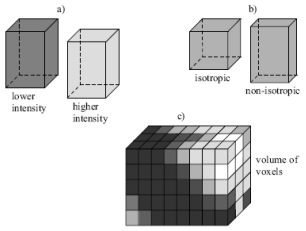

Fig. 21 Voxel representation (from (Borland07))

Converting back and forth from world-space coordinates **X** to indices **J**, called X2J and J2X functions here, requires two additional pieces of information, the size of each voxel with respect to distances in world-space, called S, the world-space position of an origin, typically O = J2X(0, 0, 0). Given this information, X2J and J2X can be written as in Eqn. 2. If world-space origin is set to (0,0,0) and the spacing is such that the largest world-space coordinate of the data is assigned to 1.0, then the data is said to be represented in the "unit cube", as our clinical modeling software encodes coordinates. The unit cube coordinate system is useful for many tasks, but it can be problematic when attempting to register data with different coordinate systems. If the coordinate system is left handed (the y-axis has flipped sign), as our clinical planning software encodes images, this can be represented by multiplying S by (1, -1, 1) and adding (0,1,0) to the unit scaled O. If the sample directions are not world-aligned, an additional rotation matrix can be attached to the formulae.

$$\mathrm{J2X}(\underline{\mathbf{J}}) = O + S * \underline{\mathbf{J}}$$

$$\mathrm{X2J}(\underline{\mathbf{X}}) = (\underline{\mathbf{X}} - O)/S$$

Eqn. 2 Basic world-to-index (X2J) and index-to-world (J2X) transforms.

Material	HU
Air	-1000
Fat	-80 to -40
Water	0
Soft Tissue	30 to 60
Bone	100 to 3000

Fig. 22 Approximate ranges for x-ray attenuation in Hounsfield units (HU) for common anatomic image values.

Value Resolution

CT intensity units, the density-related attenuation factor at each voxel, take their name from Hounsfield. Hounsfield Units measure the ratio of the local attenuation factor in a particular tissue compared to the attenuation factors of air and water (Eqn. 3). Hounsfield units range from -1000 to 3000 (12 bit range). Some important HU values are shown in Fig. 22.

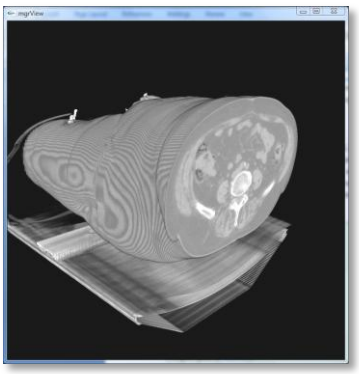

Fig. 23 The same view as in Fig. 17 intensity windowed with a lower minimum threshold.

When visualized as slices or volumes, CT visualizations typically follow the "bone is brightest" convention from radiography. It can be difficult to visually distinguish nearby values from the 4000 possible levels, particularly since there are only 256 (8 bits) of gray value difference on standard display devices. Since there are 12 bits of possible HU values, this means that 16 HU values are binned together at every intensity

level that can be shown on a standard display. However, it is straightforward to enhance the value resolution by picking out a "window" of intensities around soft tissue, for example, and expanding that range to encompass the entire 8 bits of display resolution. The tradeoff with this is that small intensity windows can substantially enhance noise. Windowing an entire volume can be quite expensive if done on the CPU. This is typically obviated by windowing only a single cut-plane at a time. In mgrView, windowing the entire volume can be done immediately using the computer's graphics hardware.

$$W(i) = \frac{\min(\max(i, top), bottom)}{top - bottom}$$

Eqn. 4. Formula for applying an intensity window to a value.

Artifacts

While CT has considerable advantages over standard analog radiography, it also has some disadvantages. Because it is a digital format that requires algorithmic reconstruction to convert the sampled signal on the machine into an intuitive spatially organized format, it can have artifacts and sampling problems that are not present in analog radiography. Artifacts develop when the underlying physical and algorithmic assumptions are broken. Broken physical assumptions such as non-narrow-beam x-ray sources or variable geometry between source and detector can lead to blurring and ringing. Broken algorithmic assumptions such as patient motion, extremely dense materials (e.g., metal fillings or prostheses) that cause "photon starvation" in their shadows, and sharp or thin objects that produce individual voxels with material of significantly different densities can all cause the characteristic starburst patterns shown in Fig. 24.

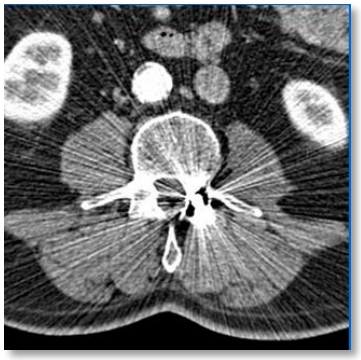

Fig. 24 A CT plane showing the characteristic "starburst" artifact from metal.

2.1.2 Other Modalities

Fig. 25 overviews a variety of other 3D medical imaging modalities covering different parts of the EM spectrum and different signaling methods. Different modalities are better or worse at detecting different phenomena. Whereas CT always collects *anatomic* information, some modalities such as nuclear medicine studies can show which regions are metabolically active. These *functional* images can be very useful for identifying pathologies. While there are currently no mgrView applications that specifically require non-CT data, combining CT images with the strengths of other modalities is an important area of future development. Ultrasound, for example, is considered non-invasive, can provide decent 3D images in a small field

Signal Type	Modality
Reflective	Ultrasound
	Visible light photography
Emmissive	MRI (RF signal)
	Nuclear Med (PET, SPECT)
	IR photography
Transmissive	X-ray (CT, radiograph)

Fig. 25 Common imaging modalities by signal type.

of view, supports limited functional scanning (Doppler), and requires very little equipment compared to a CT scanner. Visualizing online ultrasound data within the context of more detailed planning images may be an important tool in image guided biopsy.

This section briefly reviews the strengths of magnetic resonance imaging (MRI), nuclear medicine, and ultrasound in the context of potential future fusion data extensions to Model Guided Rendering. Image stacks from color anatomic sections are also included here as another kind of 3D medical imaging.

Magnetic Resonance Imaging

Imaging based on gradient field relaxation was developed in the early 1970s independently by Damadian (Damadian 1971) and Lauterbur (Lauterbur 1973). Magnetic resonance imaging (MRI) measures local proton density (hydrogen count) and local chemical compositions. As suggested in the introduction to this section, this leads to exquisite contrast resolution around soft tissue and makes pathology much easier to detect. However, MRI contains no electron density information, so it is useless for the radiation transport calculations required for treatment planning unless paired with a CT image. An area of interest for Model Guided Rendering is fusion data rendering using, for example, an MRI image to assign appearances (pathological/normal) but using the CT image for anatomic reference geometry.

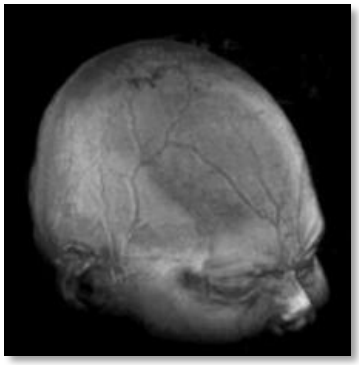

Fig. 26 MRI of the author's head.

MRI works by using radiofrequency fields (RF) to align the protons in the hydrogen atoms of water or hydrocarbons in the patient's body and then manipulates the signal to measure relaxation times, which vary by tissue type. Unlike the high energy x-rays used in CT, this RF radiation and exposure to strong magnetic fields is thought to be harmless to the patient.

Functional MRI (fMRI) can be targeted at particular systems and indicate spatial location of detected activity, which is of particular interest in brain imaging, but there is also evidence that certain non-brain pathologies can be identified by unique activation patterns. Diffusion weighted imaging (DWI) is a special type of MRI that measures the diffusion properties of tissue. Combining information from multiple DWI's can produce a diffusion tensor image (DTI) that describes directional diffusion with a tensor at each spatial sample. mgrView supports fMRI images when they are formatted as standard spatial data

distributions. mgrView currently has no support for tensor visualization, but that is a targeted future application domain.

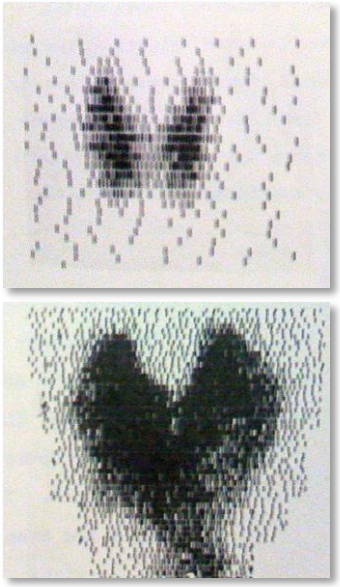

Fig. 27 Early NMI thyroid scans from (S. M. Pizer 1967). Top, a normal thyroid, bottom a thyroid with high marker uptake.

Nuclear Medicine Imaging

Nuclear medicine imaging (NMI) studies use pharmaceuticals labeled with unstable tracer elements called radioisotopes that emit radiation as they decay to a stable state. These pharmaceuticals are metabolized by targeted systems and then emit localized high energy photons as the radioisotopes decay. Nuclear medicine studies are functional by definition. For oncological studies, the radioisotopes are usually attached to glucose, which radio-labels most cancers since cancers tend to have higher glucose uptake than normal cells. NM images typically are very noisy, making them most useful when viewed in conjunction with another modality with better resolution. While MGR has no target NMI applications at present, rendering NMI fusion data, such as PET/CT is a targeted future application domain.

Ultrasound

Ultrasound (U/S) shows echoes off of tissue interfaces in the patient. The energy that it uses is sound waves above the threshold for human hearing, in the range of 20kHz to 2MHz. The primary medical application for ultrasonography is for relatively coarse imaging that does not require penetrating air or bone, such as prenatal studies (Fig. 28).

Fig. 28 Ultrasound is frequently used in prenatal screening, such as this 12-week image of the author's twins.

Ultrasound has several considerable advantages over CT. It is thought to be harmless to the patient, it has nearly instantaneous capture rates, enabling visualization of motion in real time, and the required equipment is relatively inexpensive and portable, enabling imaging during procedures without moving the patient.

In radiotherapy, ultrasound is primarily of interest for patient setup, that is, as a basis for registrations between the real-world insides of a patient and their planning CT, as in the BAT system ("B-mode acquisition and targeting") (Langen, et al. 2003). The main idea is to use a portable ultrasound device in the treatment room to find target structures and then to compute an alignment between the patient on-the-table and the planning CT. While our clinic does not currently use this method, MGR has been developed with it in mind as a potential target for multi-source data.

Anatomic Photography

Fig. 29 Color images of gross anatomic sections, such as this slice from the Visible Human can be used as a color atlas for MGR's color mapping algorithms.

Color anatomic slices can be considered as another modality. While it is unlikely that a target patient will ever be anatomically sliced up, a color atlas from anatomic sections (Fig. 29) can inform a model guided rendering by mapping the colors through a scene catalog as described in the next chapter. The Visible Human data is freely available from the National Library of Medicine's Visible Human Project (Ackerman 1998). Each subject, male and female, consists of both radiological images and color photographs of the anatomic sections.

For the female data set used in the default color maps described later, the cryosections are at 0.33mm in-plane resolution, with three slices per millimeter. For MGR's color mapping applications, structures in the Visible Human color sections were segmented manually, but similarly high resolution radiological MRI and CT images are also available for analysis, though they are not simply aligned with the color data. Given a registration between the data sets, this would provide another useful sample fusion data set.

Other types of less invasive anatomic photography, such as external patient photography or thermography, or internal endoscopic imaging are further discussed in the section on color mapping from photographs.

2.2 Interpreting Medical Images

There are two main concepts in the Model Guided Rendering taxonomy of the image analysis component of the imaging pipeline: registration and interpretation. Each layer provides important inputs for Model Guided Rendering.

1. **Image registration** is the process by which collections of images can be aligned to the same "patient space". Image registration methods may be represented either as global transforms with a small number of parameters[2] or with high dimensional local mappings, and they may be driven according to landmarks or image similarity metrics. The UNC Hospital radiotherapy clinic routinely deals with time series CTs as patients are scanned repeatedly during an ongoing treatment protocol, and aligning cross-patient or cross-modality images follows similar procedures. Model Guided Rendering uses information from image registrations primarily as a mechanism for animating anatomic shape change.
2. **Image interpretation** is about assigning anatomic labels to image regions. A simple interpretation is a label volume for an image where individual voxels are marked according to which category they belong to, prostate, not-prostate, *etc*. Such image segmentation can be done by hand or automatically by a variety of methods. One method is to use image registration tools to register an image to an already labeled "atlas space", and then to use that mapping to pull the atlas labels back to the target image. The interpretation method adopted in our clinic and used by MGR is based on segmentation by **statistical deformable shape models** (SDSMs). Here the term "interpretation" extends the basic idea of segmentation. Interpretation involves understanding not only the local label of an image region, but also the orientation and other properties of the underlying structure. Model Guided Rendering relies on image interpretation from **discrete medial representations** ("m-reps") to shade objects according to their function.

[2] Low parameter count in contrast to Fourier coefficients, which are global transformations, but provide what is here referred to as a local level of detail.

2.2.1 Image Registration

Target applications that require understanding anatomic change, such as adaptive and image-guided radiotherapy (ART, IGRT) (see (Foskey, et al. 2005) for a useful overview), require a framework for accurately mapping anatomical objects from serial images taken over several treatments into the same coordinate system as the planning or other reference image. The most common mappings are global rigid transforms, but current research activity is aimed at developing practical and reliable methods for creating space-filling non-rigid mappings. Our clinical image registration software, ImMap and its variants, can generate both global and local registrations. Visualizing registration relationships between images is one of the mgrView program vignettes presented later in section 5.2, *MGR Applications in Adaptive Radiotherapy*.

Global Transforms

Rigid image-to-image registrations can be expressed as a global matrix transformation. Positions in the space of the target image can then be passed through this matrix transform to find the corresponding positions in the source image. Rigid and more general affine transforms can be derived algorithmically for 3D point correspondences, as do both Procrustes[3] (which uses explicit correspondences) and Iterative Closest Point (ICP, see Fig. 30) (Besl and McKay 1992) (which computes both the transform and the best set of correspondences for two unlabeled sets). Rigid image transforms are most appropriate for within-patient registrations, where images change mostly in pose from day to day. Similarity (rigid plus scale) or affine (rigid plus scale and shear) transforms are more useful when studying cross-patient registrations. For dense intensity correspondences such as photo-constancy or mutual information, a global registration can be found by optimizing over the elements of the transform matrix according to an image similarity metric.

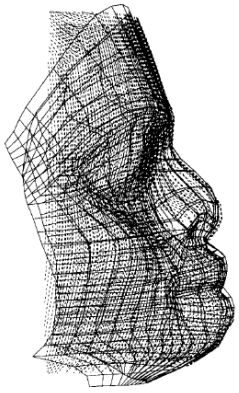

Fig. 30 Parameteric surface to point data registration from (Besl and McKay 1992)

An important application of global transforms is to align a 3D planning image with a 2D image of the patient taken at treatment time. Such visualizations can be used to verify that the patient is set up correctly on the therapy machine. This is typically done manually, using lasers to line

[3] Procrustes was a Greek bandit who forced his victims to lie in an iron bed and cut off their feet to fit his measuring device.

up the room origin with fixed surface landmarks such as tattoos. In other procedures such image-to-world alignment may involve a mechanical device such as a frame physically bolted onto the subject's head that provides a reference coordinate system (a "stereotactic head frame").

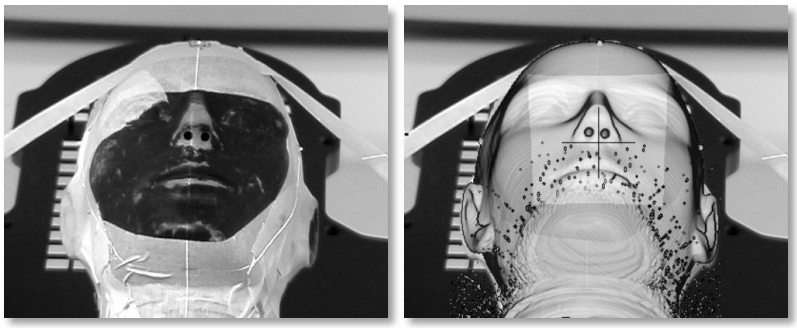

Fig. 31 Left, a photograph of our clinic's phantom, RANDO (or "Randeau" as he is sometimes known) on the treatment table. Right, a rendering of a CT of RANDO synthesized from the same camera point of view and overlaid onto the photo.

mgrView has been designed to produce images that combine real world photography with 3D images. Fig. 31 shows an example of a planning image transformed and rendered into the space of a 2D photograph. The later section on color mapping from photographs describes a method for projecting a 2D photograph into the space of a 3D planning image and includes some relevant discussion of camera models.

Local Mappings

Non-linear image-to-image registrations[4] cannot be expressed concisely as a matrix transform, so they are typically represented locally across the field, either indirectly by a set of control points for splines, or directly by a dense displacement field. Non-linear registrations are more general than affine registrations because they can capture both local and global properties (although in practice, they usually only capture residual local changes after a global registration has been applied). As with global registrations, local registrations may be feature driven, such as thin plate splines (TPS) (Bookstein 1989) or the basis paths used in (Joshi and Miller 2000). Or they may be intensity driven, such as "optical flow" in 2D (Horn and Schunck 1980) or "demons" in 3D (Thirion 1998), fluid flow (Christensen, Joshi and Miller 1997), or free form deformation via B-splines (Rueckert, et al. 2006). ImMap and its variants use an atlas-based version of fluid flow described in (Davis, et al. 2004).

[4] The class of affine transforms is indeed linear in homogeneous coordinates.

A method for interpolating quickly between images according to such a registration field for the purposes of visualization is described in the later section 4.1, *Volumetric Animation*, and is a key method in the target applications described in section 5.1, *MGR Applications in Adaptive Radiotherapy*.

In contrast with rigid or affine registrations, which can be determined algorithmically, deformable registrations are usually determined by optimizing a registration metric in a very high dimensional space. In the simplest case of representing the registration by an independent displacement vector at every voxel, there will be 3 x the voxel count parameters to optimize and possibly more if the registration uses multiple time steps. The registration metric is usually comprised of two parts, an image similarity function such as sum of square differences between the source and registered target image, and an irregularity penalty that attempts to keep the deformation organized and legal under various definitions. Because of the very large number of parameters, these regularization terms are usually too simple to rescue the process from significant problems with local minima. Current research is frequently focused either on dimensionality reduction (decomposition into control paths, *etc.*) or on introducing more sophisticated and finely tuned regularization terms.

Registration layer information is a useful input to many of Model Guided Rendering's algorithms, but it is insufficient for the regional image-to-image mapping algorithms described later. While registration of a target image to common atlas coordinates, for example the Talairach brain (Talairach and Tournoux 1988), can give object-labels to each voxel, determining qualities such as local orientation requires a complete object-based volumetric coordinate system fit to each image region. Such coordinate systems are a natural by-product of the image segmentation methods described in the next section.

2.2.2 Image Interpretation

The image interpretation input to Model Guided Rendering comes from a type of deformable shape model. Statistical shape models were originally developed independently in (Kendall 1984) and (Bookstein 1989) for structures identified by a few important landmarks found in each of many patient images. Modern statistical shape models for tiled

surfaces were proposed in (Cootes, et al. 1993) as a method for image segmentation.

As with the deformable registration framework described previously, segmenting an image via deformable shape models relies on a two term optimization balancing an image match against a model penalty. However, parameterized shapes are much lower dimension with respect to the number of training samples usually available, and they can be restricted to not only a legal (*e.g.*, non-self-interpenetrating) or regular (*e.g.* smooth) shape subspace, but to credible shapes (*e.g.*, the liver is liver-shaped) as well.

As an example of shape credibility, consider a simple image term consisting only of a threshold based edge finder. As shown in Fig. 32, an edge-finding segmentation algorithm with no penalty for deviating from a head-like shape will incorrectly label the sinuses as skin. This segmentation is legal and smooth, but it is not credible. High dimensional deformable registrations tend to suffer from the same class of problems because it is difficult to embed notions of shape credibility in the regularity term.

Fig. 32 A non-credible but legal skin segmentation with a simple edge detector.

Following (Mumford 1994), the image match and model penalty terms optimized during image segmentation by statistical deformable shape models (SDSM) can be interpreted as probabilities and the framework is usually called "Bayesian probabilistic" image segmentation.

The SDSM itself is an object-specific shape model that characterizes shape change and likely intensities relative to a "typical" instance. The typical shape usually is taken to be the mean shape of a population of training shapes, and relative shape changes are encoded as a limited number of important modes of shape variability. These modes are derived by applying principal component analysis (PCA) or another decomposition method on the parameters of the training shapes. This results in an even lower dimensional parameterization where any particular shape in the space defined by the training set can be completely and uniquely identified by a few coefficients to within some small truncation error. Distances in this space provide a metric for shape that can be used as the basis for numerical methods of shape discrimination, comparison, and interpolation.

In the context of Bayesian image segmentation, the SDSM provides the fitting optimization with both the initial shape estimate and a straightforward metric for shape credibility. In Bayesian image segmentation, this metric can be interpreted as a prior probability distribution on shape and is usually called simply the "shape prior". The SDSM additionally provides the fitting optimization with a coordinate system for expressing image intensities in model-relative terms. The optimization, then, becomes a compromise between the shape prior and the shape suggested by the image intensities. (Fig. 33) shows an example of an SDSM's initial and optimized fit to a target image.

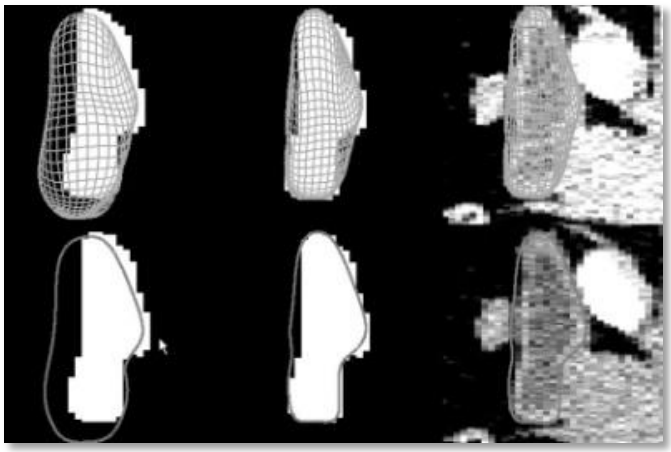

Fig. 33 Fitting a statistical deformable model to a target training image. Top, 3D surface views and bottom, single sagittal slice views of bladder template geometry. Left, initial shape estimate coarsely aligned to a target training image; middle, deformably fit to that image; and right, in the context of the actual grayscale data. (From (Merck, et al. 2008))

Shape Models

Many types of shape and intensity representations have been proposed as bases for probabilistic segmentation, and all have different utility and drawbacks. In particular, certain types of SDSMs can provide MGR with an object-centric coordinate system for local image regions. The most common shape model in the literature is boundary-only, sometimes called a point distribution model or PDM proposed in (Cootes, et al. 1993). PDMs are, however, insufficient for MGR's requirements because they do not parameterize the interior of the object, so they do not explicitly establish volumetric correspondence between 3D source and target regions. Such volumetric correspondences are the basis for the color mapping algorithms that are proposed in the next chapter.

The parameterization used by Model Guided Rendering is the discrete medial representation called "m-reps" – although the methods could be generalized to other shape representations with a well defined volumetric coordinate system. M-reps are well suited to image segmentation tasks because they capture not only position information

at an anatomic region's boundary but also model-relative orientational information throughout the region's interior. For the purposes of MGR, the parameterization handles plausible physical changes such as local twisting and bending in a way that is naturally reflected by the volumetric coordinate system.

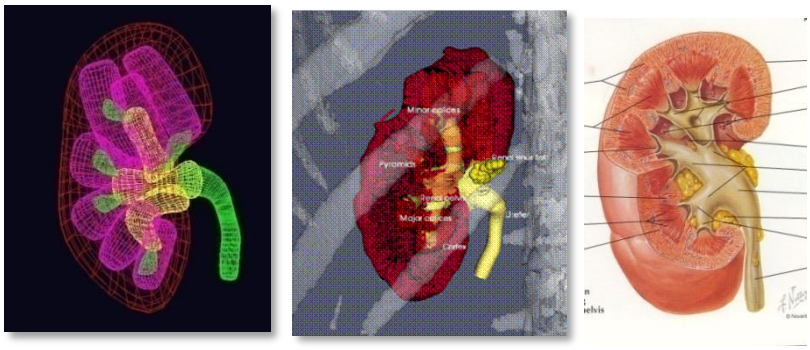

Fig. 34 Left, M-rep figures and sub-figures fit to a kidney and its internal pyramids and calyces. The complex nested volumes and smooth surfaces are represented with a few hundred parameters. Middle, an early MGR image with detailed internal orientations generated from this model. Right, a reference illustration from Netter.

The m-rep parameterization is described in section 3.1, *Creating a Scene Catalog*, but because Model Guided Rendering is a post-segmentation task, the detailed mechanisms by which data can be automatically segmented have been omitted. See (Pizer, et al. 2008) for an overview of the representation and segmentation pipeline, see (Broadhurst, et al. 2006) for a detailed description of the embedded intensity statistics, see (Fletcher, et al. 2004) for an explanation of the governing statistical model, and see (Merck, et al. 2008) for a description of the shape training process.

M-Rep Software

The primary m-rep editing and fitting tool is called **Pablo**[5]. Pablo is a useful tool for developing new Model Guided Rendering scenes. It is available from UNC's Medical Image Display and Analysis Group (MIDAG) under a research license. M-reps are also implemented in the commercial male pelvis segmentation system MxStruct MP, developed by Morphormics (http://www.morphormics.com). See section 5.2, *MGR Applications in Adaptive Radiotherapy*, for details of mgrView's shape representation and supported file loaders.

[5] The program was named based on the following quote from Pablo Picasso – *Computers are useless. They can only give you answers.*

2.3 Classic Medical Image Visualization

The field of medical image visualization is very broad. This section is focused exclusively on classical single-image visualization methods that are based entirely on local properties of the data, *e.g.*, by intensity or derivatives of intensity, or by scene geometry, *e.g.*, distance from the viewer. This review further restricts its scope to the historical and methodological influences of Model Guided Rendering. Later sections discuss precedents for interpretation driven scene design, such as importance rendering. This section is divided into three topics.

1. A very brief review of **surface rendering** for medical visualization. While surface driven visualization methods are generally inadequate for complex anatomic scenes, a major goal of Model Guided Rendering is to integrate surface and volume rendering and to use each where it is most beneficial to comprehension.
2. The topic **object-order and image-order direct volume rendering** (DVR) compares ray cast volume rendering as proposed by Levoy with rasterize-and-blend volume rendering as proposed by Drebin. Ray casting or "image-order" rendering is the grandfather of DVR methods, and many of the basic concepts from Levoy map directly onto the mgrView toolbox. Rasterize-and-blend or "object-order" volume rendering can be much faster than ray casting because it relies on graphics hardware acceleration, but it has traditionally been limited in quality by the fixed function hardware rasterization pipeline. However, programmable shader hardware has lifted many of those restrictions, and mgrView's rendering core is actually implemented with this method.
3. **Independent-image scene design** reviews some major naïve methods for pseudo-coloring such as so called "volume illustration" and for simple clipping. Many of these image-order methods have been re-implemented in mgrView as fast object-order methods.

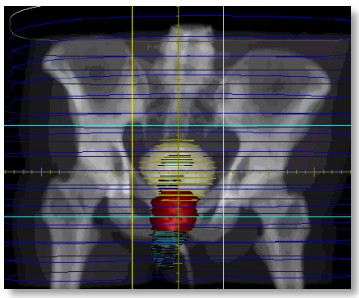

Fig. 35 Multislice segmentations (contours) and tiled surfaces rendered in PLUNC. Stacks of manually drawn contours are knit together into target region surfaces by the "FKU77" algorithm (Fuchs, Kedem and Uselton 1977).

2.3.1 Surface Rendering

Extracting a 2D manifold of data from a 3D data set is an easy to manage method for isolating certain kinds of important features and suppressing the rest of the volume data. Virtual colonoscopy, where 3D visualization is well adopted (Fig. 6), relies on surface extraction for

both the soft pink plastic looking walls and the flythrough camera path planning.

Surfaces of interest in the volume may be extracted according to purely geometric methods, such as **cut planes**, or may rely on data driven methods such **isosurfaces.** Surfaces may also come from the boundaries of manual or automatic image segmentation as discussed in the previous section. Once extracted, surfaces can be scan converted into label volumes or be used to compute normals, which are vital for **lighting**. Much of the later discussion on volume shading revolves around estimating local normals in the absence of surfaces for the purposes of lighting.

Cut Planes

Trans-axial slices through the data are the simplest and most widely used method for interrogating volume data. Saggital and coronal slices are also used in some applications, and occasionally cuts in all three axonometric directions will be combined to explore a particular region of interest, as shown in Fig. 36. Such views are sometimes colloquially called "2.5D display". These simple views are optimal for certain tasks like slice-by-slice manual segmentation, but they do not provide the user with very good context information about the 3D shape of the anatomic structures.

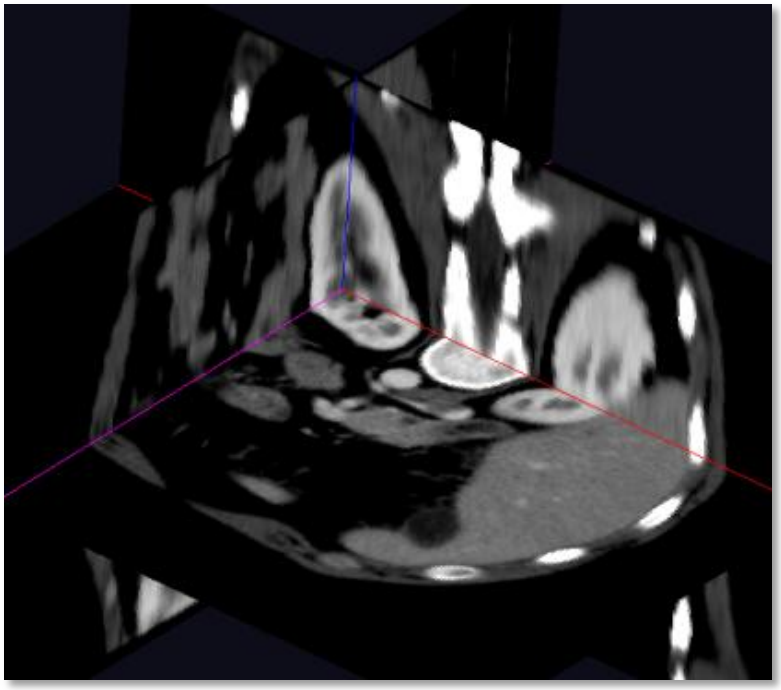

Fig. 36 2.5D view of a CT data set of the male pelvis with intersecting axis aligned cut planes.

Axis aligned cut planes were originally used exclusively because it is particularly simple to sample an axis aligned planar texture from a 3D data set. Using 3D texturing hardware, however, planes of any orientation can be sampled just as quickly by simply transforming the texture coordinates at the corner vertices in OpenGL. mgrView allows any number of independently positioned and oriented cut-planes to be attached to a particular scene. Cut plane pose can also be aligned according to other objects in the scene, for example, to be normal to the medial axis of a modeled region (Fig. 37).

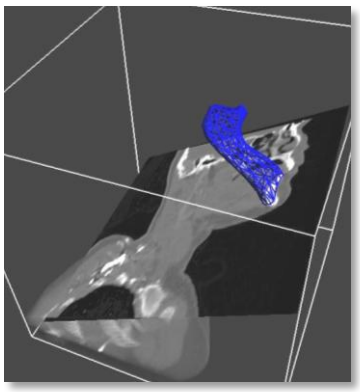

Fig. 37 A clip plane aligned to be normal to the along-direction of the mandible.

Surface Extraction

Finding the isosurface of a binary label image or the isosurface of a gray image with anatomic structures that have distinct values (essentially segmenting by threshold) is one method for identifying regional boundaries in an image.

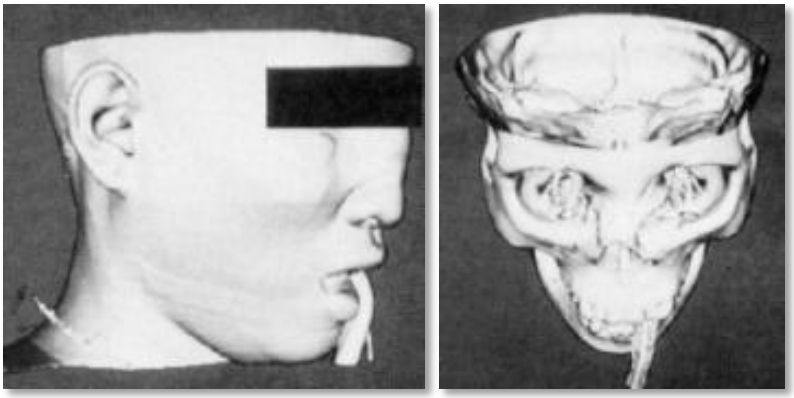

Fig. 38 Marching cubes iso-surface extraction with different reference values and views from (Cline, Lorensen and Ludke 1988)

The isosurface method was originally developed at GE Labs and famously proposed as the “marching cubes” algorithm in (Lorensen and Cline 1987). The authors subsequently applied the general method particularly to medical image visualization in (Cline, Lorensen and Ludke 1988). (Cline, Lorensen and Ludke 1988) has nice example of visualizations of bone, skin, and muscle (shown in Fig. 38), which happen to be the only anatomic tissue types in CT that are clearly discernible by scalar intensity alone. Most features in medical volumes do not have such distinct values with respect to their neighbors, so isosurfacing is not generally useful. Marching cubes/isosurfacing is currently not directly implemented in mgrView, but it can be computed offline using Kitware’s VTK (http://www.kitware.com) or another package and then loaded as a regular set of surfaces.

Surface Lighting and Texturing

Local lighting is typically dependent on combinations of four terms: the normal direction, the view direction, the light direction, and the local surface color. Lighting in volume data follows similar rules, although the normal direction must be approximated because it is not well defined. Diffuse lighting based on the dot product between the surface normal direction and the light direction (see (Phong 1973) for the complete model) provides default lighting for any surfaces displayed in mgrView. Methods for extending the lighting model to surfaces with solid textures are described in detail in the next chapter. Other shading models such as the so-called "Non-Photorealistic" (NPR) tone shading (Gooch, et al. 1998) and cell shading are also implemented for surfaces in mgrView and can be assigned to regions either programmatically or through the default UI.

$$I = \left(\frac{1 + l \bullet n}{2}\right) k_{cool} + \left(\frac{1 - l \bullet n}{2}\right) k_{warm}$$

Eqn. 5 Formulation for Gooch tone shading in terms of the normal direction, *n*, the light direction, *l*, and colors *k*.

Tone shading, sometimes eponymously called "Gooch shading", was originally designed for technical illustrations: hot and cool colors are assigned to theoretically balance the perceived intensity of the light and dark regions, so that surface features will not be obscured by lack of diffuse lighting. The Gooch model is shown in Eqn. 5 and a rendering from mgrView shown in Fig. 39. Cell shading is a nearest neighbor version of Phong with only a few "steps" of possible light values. Contour shading is similar, but it renders only those faces or edges that are most nearly orthogonal to the view direction (Fig. 40). Contour-finding is an important part of MGR's extension of cast shadows to "importance stenciling" described in a later chapter. mgrView's NPR surface rendering modes are based in part on ATI's white paper on NPR fragment shaders (Card and Mitchell 2002).

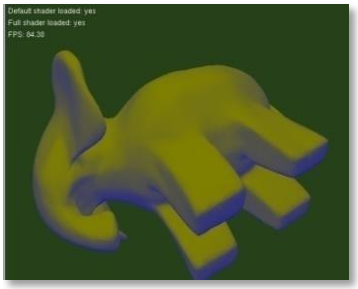

Fig. 39 Gooch tone shading in mgrView.

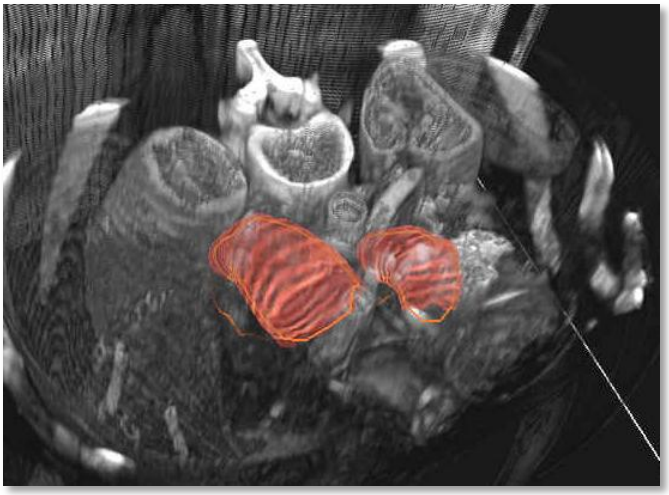

Fig. 40 Contours can be an effective technique for showing surfaces without occluding the underlying data.

2.3.2 Object-Order and Image-Order DVR

Image-Order Volume Rendering

Among other things, surface rendering suffers from occlusion problems – surfaces can be hidden behind other surfaces. A partial solution to the occlusion problem can be found by projecting the entire 3D volume simultaneously onto the screen. This method, pioneered by (Levoy ACM88) and extended in (M. Levoy 1990), is known as "direct volume rendering" (DVR). Levoy's original methods were based on following rays into the scenes; this method is variously referred to as "ray casting", "image order", or "backwards" DVR. Image-order DVR can be most easily understood by considering a set of rays projected from the eye through the image pixel grid, and passing through the patient data (Fig. 41). Each ray is sampled along its path, and each voxel traversed contributes to the shading and opacity accumulated for the ray.

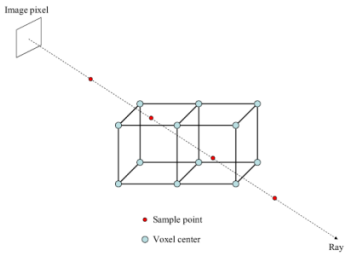

Fig. 41 A ray is sampled along its length in such a way as to cover all voxels (taken from (Borland07))

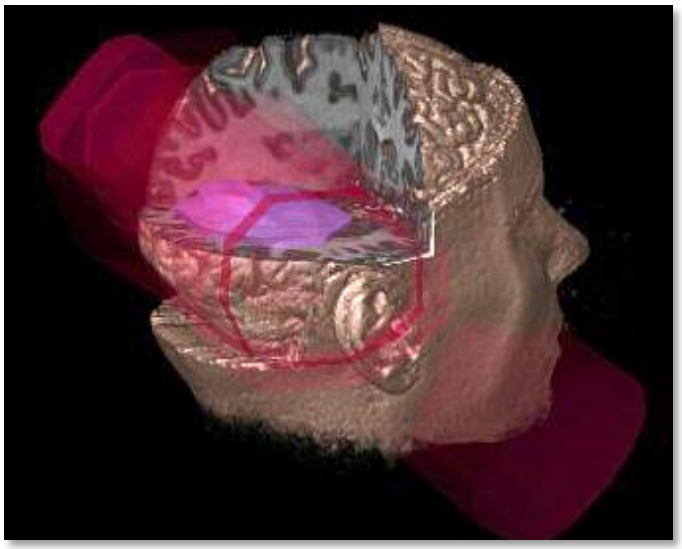

Fig. 42 A pseudo-DVR visualization for radiotherapy planning from (Levoy VBC90), also rendered as a white light hologram in the lobby of UNC-CH's Sitterson hall.

Images from the early Levoy papers were, in many ways, similar to isosurface renderings. Levoy's major insight in DVR was recognizing that regions of high gradient magnitude contained more information about the internal structure[6], so those areas are weighted more heavily compared to areas of low gradient. This assumption amounts to noticing that regions of high gradient magnitude tend to represent surfaces and that the local gradient direction corresponds to the local normal direction. This is an excellent assumption at interfaces such as

[6] It is unclear whether Levoy originated this idea, Cline suggests that an even earlier 1986 Höhne paper proposed the same idea.

skin-to-air and meat-to-bone; *i.e.*, those same places where isosurfacing is particularly effective.

Image order volume rendering is derived from (Kajiya84), which describes an algorithm for ray tracing volume data (transparent gasses with interfaces), and from (Kajiya86), which generalizes the method and defines the 'rendering equation', a mathematical formulation for how objects, appearance properties, lights, occlusions, and transparencies in a scene can be combined and projected onto an image according ray casting.

Most widely used research volume rendering engines, such as that in Kitware's VTK and its derivatives, VolView and Paraview, as well as Analyze and MRIcron, are based on ray casting engines. Because of the common root with physically based rendering, ray cast DVR is very flexible and can be easily combined with other physically accurate light transport algorithms, such as lighting, reflection, soft shadows, and anti-aliasing. Lighting and shadows in particular are very strong shape cues. Ray cast DVR is also amenable to speed ups like oct-tree traversal (Levoy's original proposal), early-exit for opacity saturated rays, and incremental sweetening by increasing resolution. Because the rays are independent, ray casting algorithms are easily adapted to multi-threaded architectures.

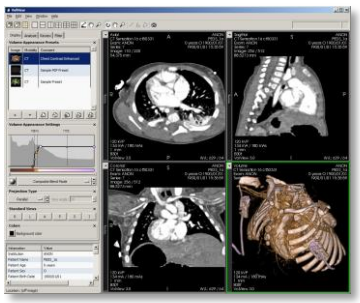

Fig. 43 Interface and rendering from VolView.

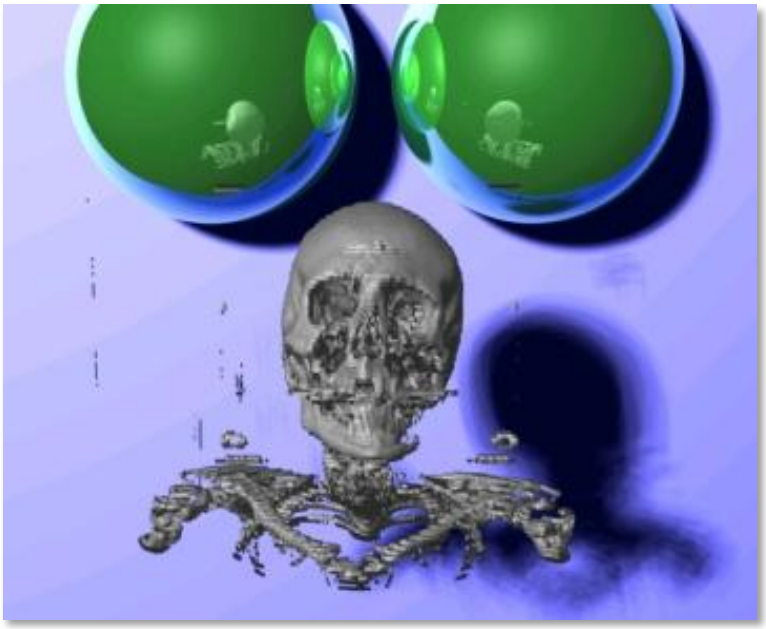

Fig. 44 An image from the original ray-casting core considered for mgrView. Effects such as reflections, soft shadows and super-sampling can be easily implemented in a ray-casting framework.

However, ray casting is very computationally expensive, and MGR has significant additional overhead in doing a large number of trilinear interpolations for external image source lookups. Therefore, without access to advanced hardware, ray casting's slowness outweighs its relative advantages. Fig. 44 shows a view from an early ray-casting implementation considered for mgrView's rendering core; it took on the order of 10 minutes to render a 512 x 512 pixel image.

Object-Order Volume Rendering

An alternative volume rendering implementation originally proposed in (Drebin, Carpenter and Hanrahan 1988) is to rasterize geometry that has been assigned a corresponding texture in the patient image, and then blend the textured geometry together using standard graphics acceleration hardware (the graphics processing unit or "gpu"). Drebin[7] originally proposed cutting through the volume with stacks of planar geometry (Fig. 45), although other geometry has also been used, such as Gaussian footprints ("Splats")(Westover 1990) and spheres or other curved manifolds. Because this method works in "object-space" rather than "image-space", as ray casting does, these methods are sometimes called "object-order" or "forward" DVR.

Fig. 45 The otter with an extra wrist bone from (Drebin, Carpenter and Hanrahan 1988). When shown this display, scientists discovered a hitherto unknown wrist bone. This is one of the few examples that the author has been able to find of volume rendering actually contributing novel scientific utility.

To review the basic method:

1. The back-to-front marching order[8] is determined by computing the dot product of the view direction with positive and negative cardinal directions and taking the min.

[7] Drebin's original paper is also of interest because it proposes a local *maximum a posteriori* segmentation and gradients on that as part of the rendering process. Thus, like Levoy, the underlying assumed surfaces are made explicit in the algorithm.

[8] The order can be reversed, from front to back by using a different blending formula.

2. Starting with the back face and moving to the front face ("painter's algorithm"), texture a piece of rectangular geometry (a "quad") the size of the slice with the volume data samples and project it onto the screen, using the volume texture to control both the color/shading and transparency of each fragment[9].

Speed-ups such as early exit are unnecessary since operations are done in parallel; but incremental sweetening can be done by adding additional samples (*e.g.*, planes) along the rays as opposed to adding additional rays.

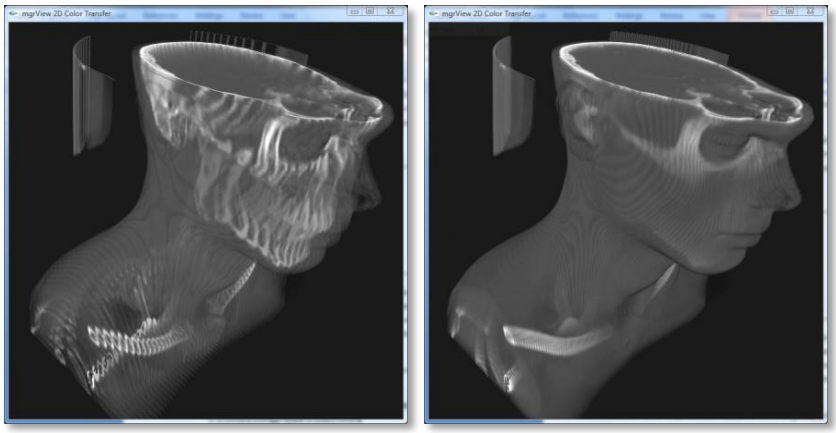

Fig. 46 Scenes renderered with mgrView using 64 planes (left) and 192 planes (right).

Because this method of plane-compositing maps directly onto graphics hardware, this method has low overhead and can achieve interactive rate volume rendering on even modest work stations.[10]

When these object-order methods were originally proposed, they suffered from three main problems.

1. Under-sampling oblique paths through the data
2. Limited storage space in video RAM
3. Fixed functionality hardware pipeline not amenable to per voxel effects such as can be achieved by the "transfer functions" discussed in the next section

[9] Terminology: A "fragment" is the data necessary to generate a single pixel in the frame buffer. Pixels are rgba (red, blue, green, alpha, where alpha measures opacity) image samples; fragments are *potential* rgba image samples that include extended properties such as depth and texture coordinates.

[10] For additional insight into how this algorithm works, look at Joe Conti's vol3d.m function for Matlab. http://www.mathworks.com/matlabcentral/fileexchange/4927

The least significant problem is with regularity of sampling. With ray cast methods, one can regularly interrogate values from the data along a ray. Compared to a ray perpendicular to the volume sampled evenly along every voxel, a different ray entering the volume at an angle may need to take multiple samples within a voxel to achieve the same spacing. Using axis aligned plane casting, sampling is fixed at each plane, which means that angled rays will be undersampled with respect to rays that are perpendicular to the volume face. This is also the source of the "cornrow effect" sometimes seen at the boundaries of dense regions, where the edges of the individual planes can be clearly seen, as shown in Fig. 47.

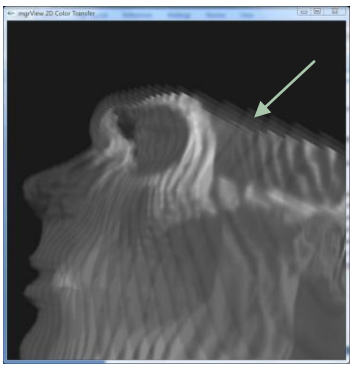

Fig. 47 Close up of "cornrowing" effect at the edge of a 92 slice volume using mgrView.

Cornrowing can be addressed most simply by adding simply additional planes to the display. However, a more principled approach is to slice the volume with screen-aligned cut planes, as proposed by (Cullip and Neumann 1993). Screen-aligned plane casting was originally an ambitious idea, since it required the volume data to be interpolated in 3D, rather than 2D, and thus required a supercomputer to do at any reasonable speed. However, within the last few years 3D texture interpolation has been included on most new graphics cards, and the method has become quite easy to implement. Each compositing plane and its texture coordinates are simply rotated by an extra matrix multiply to be normal to the view direction. The renderer in mgrView supports either axis or view aligned volume renderering. Cutting the volume with a set of nested spheres rather than planes can also be used to normalize the number of volume samples per pixel.

The issue of storage space on the fast video RAM used by the gpu has been trivially surmounted as faster, cheaper memories became commoditized over the last few years. The size of the target image has become essentially irrelevant with respect to object-order DVR for medical image sizes in our clinic; volumes up to 512 x 512 x 128 fit easily into texture RAM. Stored as single bytes (truncating 4 bits), such a data set uses only 33 Megabytes of what is typically at least a 256 Mb store. Because the image gradient is expensive to sample in real time, the gradient of the image can be pre-computed for lighting and stored as an additional 3-channel texture array where each channel carries the directional difference information with respect to the x, y, and z

directions (Fig. 48)[11]. This can require an additional 100 Mb, and adding further high-resolution 3D color images such as the scene catalogs and color atlases described in the next chapter can indeed stress system storage if not managed correctly.

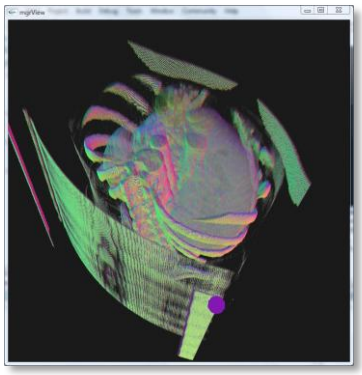

Fig. 48 The gradient volume of an abdomen image stored as rgb channels and rendered directly with mgrView.

Older graphics accelerators implemented a fixed functionality rasterization pipeline, which restricted the extensibility of object-order volume rendering. Higher order effects, in particular, intensity windowing and lighting, are trivial to compute in the ray casting framework, but they are precluded by the fixed functionality hardware pixel processing.

With a few exceptions (such as (Dachille, et al. 1998), which proposes using a simplified look-up table for pre-computed lighting) explicit per-pixel effects were not deeply explored for object-order volume rendering until programmable shaders were introduced, and even then they were not practical until compliant hardware became ubiquitous. (Westover 1990), who proposed voxel-wise compositing by elliptical Gaussian 'footprints', or volume 'Splatting' as it is colloquially called (Fig. 49), indirectly addresses the problem by using extremely small elements[12]. Westover did not focus on this advantage, but because individual voxels are composited it is slightly easier to approximate per-pixel shading effects without relying on a programmable shader. For example, a normal can be assigned to each voxel for lighting purposes. A splat rendering core was also considered for mgrView (see subsequent Fig. 119 for an image generated from it). However, the large number of primitives required to be rasterized made it fairly slow, particularly since programmable graphics hardware has lifted most of the restrictions of the plane casting method.

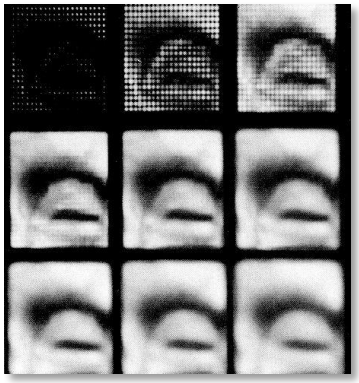

Fig. 49 Image from (Westover 1990) illustrating the effects of variously sized "splat" kernels. The kernels in the top row are too sharp, giving inadequate coverage of the scene. The kernels on the bottom row are too broad, causing unnecessary blur.

Using programmable shaders, object-order methods can easily achieve high quality results at interactive rates. The fixed functionality shading pipeline can be overridden by writing short programs in a c-like language. mgrView implements programmable shaders in the OpenGL

[11] This technique is used again later for representing local displacement fields. As an historical side note, (Fuchs and Pizer 1986) suggests using the frame buffer for (x,y,i) data in their patent for 3D display using a varifocal mirror.

[12] It is also to this paper that I owe the taxonomy of volume visualization techniques as backwards, forwards, and surface based used in organizing this discussion.

shading language (GLSL) (http://www.opengl.org/documentation/glsl/), but similar results could be achieved using other gpu languages such as nVidia's Cg or DirectX's HSL. Fig. 50 shows the standard OpenGL graphics pipeline along with annotations regarding how MGR objects are classified as image or geometry data and noting where fixed functionality can be overridden by custom vertex and fragment programs to support various MGR methods. The MGR methods described in the next two chapters are all designed to be efficient in this context. That is, they are largely independent of one another and require only small amounts of data for their calculations. See (Shreiner, et al. 2005) and (Rost 2006) for details of how the OpenGL pipeline and GLSL work together.

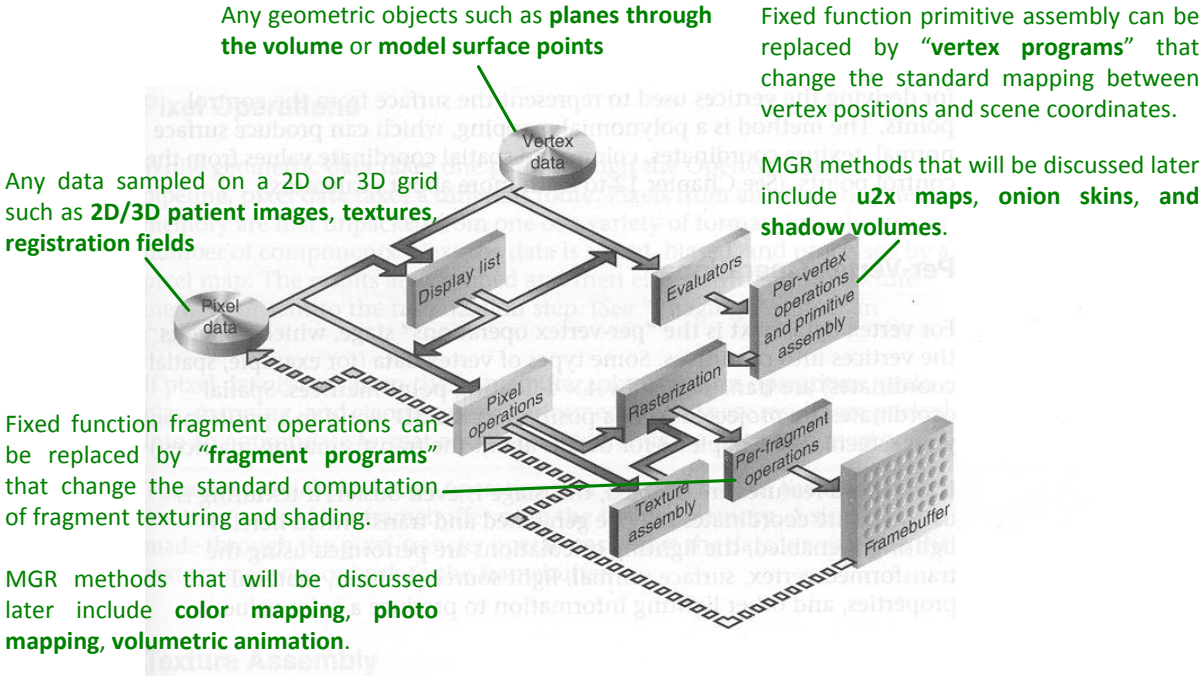

Fig. 50 Annotated OpenGL pipeline originally found in (Shreiner, et al. 2005).

Using a standard single-source volume rendering shader with intensity windowing, mgrView achieves frame rates of over 20 fps for 200 slice standard DVR on a modest laptop with an NVIDIA Quadro NVS 160M graphics accelerator and over 10 fps for 100 slice DVR on an a standard desktop workstation with a NVIDIA GeForce 6200 graphics accelerator. Comparative values for other more complex shader models with color mapping or volumetric animation are given in later sections and summarized in the Conclusion Chapter in Table 4.

2.3.3 Independent-Image Scene Design

Illustrations are designed to **fulfill a communicative intent**, such as showing the location or use of an object. Illustration is based both on style (called 'appearance' in MGR) and on composition (sometimes called 'clipping' in the literature). The idea of computed illustration was first proposed in (Seligmann and Feiner 1989). Though this paper is not algorithmically interesting, nor are the results compelling by modern standards (8k polygons in 8 seconds), the paper lays out the important guiding principles for computed illustration. Seligmann outlines a language for expressing communicative goals (location, relationship, property, difference). Then for each goal he describes different design strategies, which can be, in turn, composed of different styles (highlighted object, visible context, ghost object at previous location, annotate). He then uses a test and evaluate process: several candidate renderings are generated, and then each is evaluated and ranked according to questions like 'Is the object occluded?' and 'How much contrast is there between the object and its context?' While MGR does not rely on such a fully automatic framework for styling and composing scenes, this core way of categorizing the tasks at hand has been thoroughly integrated into the design of the framework.

There are two basic “styles” of DVR: “Levoy-like” Superman x-ray vision, or maximum intensity projection (MIP), which simulates a radiographic view (sometimes called a “digitally reconstructed radiograph” or DRR). Following Seligmann, it could be argued that both of these modes suffer from two main problems besides speed.

1. **Understandability** – it is difficult to identify important features in the scene. Scene understandability is addressed by MGR’s **appearance** components.
2. **Occlusions** – it is difficult to focus the view to see relationships between these features. View focus is addressed by MGR’s **composition** components.

These issues are typically addressed in independent image rendering by designing increasingly complex transfer functions or increasingly complex geometric dependencies that attempt to implicitly tease out important anatomic structures by bringing tangential external information to the scene. In MGR these issues are addressed explicitly by working in the proper coordinate systems for local anatomic structures.

Independent Image Appearance Design

The goal of assigning appearance in classic volume rendering methods is to shade implicitly identified regions in a scene so that they are more easily identified, if not more easily understood. This has typically been addressed by assigning "**transfer functions**". In image-order algorithms, the idea behind a transfer function is that as a ray traverses the volume data, it keeps track of what data intensities it has passed through and accordingly accumulates or modifies the pseudo-color that will be returned to the initializing pixel. Fig. 51 shows an example of an impressive rendering and VolView's user interface for controlling the transfer function. Pseudo-color transfer functions are not implemented in mgrView, although there is no particular reason that they could not be. The detail that follows is to give an idea of what competing non-multi-source volume rendering methods have to offer in terms of rendering styles.

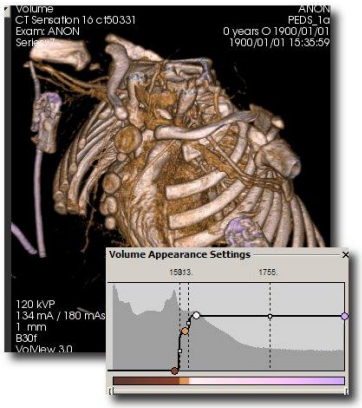

Fig. 51 Detail of VolView's 3D rendering and transfer function interface from Fig. 43. The transfer function interface shows a histogram of the intensities in the scene. Leftmost is air, rightmost is bone. The overlaid line controls the opacity for each intensity value (transparent at air, approaching opaque at bone). The bar on the bottom shows the color assignments for each intensity value (brown for soft tissue, white-pink for bone).

As proposed, transfer functions were originally quite simple, *e.g.*, linear and clamped to a range, much like standard intensity windowing, with additional surface finding from gradient magnitude. Such transfer functions can easily be focused on the absolute intensity of voxel with a direct mapping between value and a pseudo-coloring scheme. For regions that can be simply identified according to intensity, this is quite useful – bone (high HU) can be shaded white, and muscle tissue can be shaded red or pink. Unfortunately, most of the detail that is important in a scene is not distinct by HU value alone, so a considerable amount of effort has gone into creating complex and precisely targeted transfer functions to discover correspondingly complex relationships in the data. In a sense, the goal of these multi-dimensional transfer functions is to identify important regions by doing a local on-the-fly data segmentation according to automatically extracted features (again, typically implicit surfaces or derived properties thereof like local curvature (Fig. 52)).

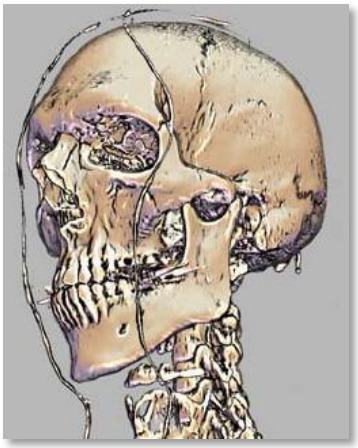

Fig. 52. One of my favorite volume renderings, using a curvature based transfer function from (Kindlmann, et al. 2003). Note that it is very similar to a *surface* rendering.

As the controls become more complex, the views become quite fragile. Because no clinician could be adequately trained to manipulate the many parameters in the most detailed systems, research turned instead to methods for ameliorating the onerous tuning with more intuitive controls. (Kniss, Kindlmann and Hansen 2001), for example, describes a simplified framework for manipulating multi-dimensional transfer functions.

Beyond transfer functions is a set of volume non-photorealistic (NPR) methods such as those described previously in this section. Volume equivalents to tone and cartoon shading and contour rendering, as well as the application of pen and ink or pencil textures have been proposed. NPR shading for volumes was first described in (Ebert and Rheingans 2000) and was then expanded in (Rheingans and Ebert 2001), which coins the term "volume illustration". Rheingans proposes a variety of rendering enhancements, but despite the title, tone shading is the major NPR technique she invokes. Unfortunately, none of their renderings would seem to be particularly more understandable than a standard Levoy-view for an untrained viewer (see Fig. 54 for an example). (Tietjen, Isenberg and Preim 2005) extends the idea of "volume illustration" (even appropriating the term) to account for segmented objects (Fig. 53).

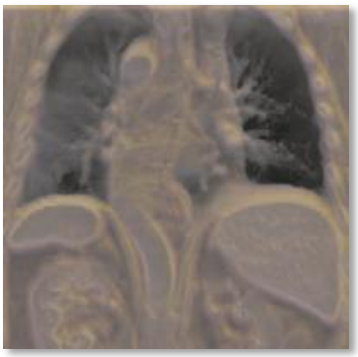

Fig. 54 Tone shaded illustrative rendering of the thorax from (Ebert and Rheingans 2000).

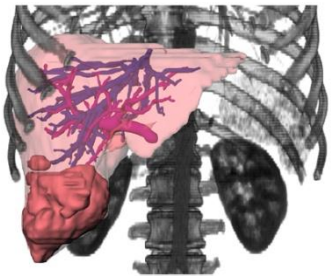

Fig. 53 (Tietjen, Isenberg and Preim 2005) describes a method for combining segmentations with DVR to create hybrid illustrative renderings.

There are a few papers that propose truly novel styles, such as applying "pen and ink" methods to volumes. (Lu, et al. 2003) proposes a method for volume rendering with stipples (Fig. 55 left). This style is based on the artistic pointillism technique and has nice results for a non-model-based method. An interesting insight in this paper is that a large number of 3D stipples can be precomputed offline for each voxel, and then subsets of stipples can be taken as the view-to-voxel function changes, so that individual stipples do not move as the number of stipples at each sample changes. (Fischer, Bartz and Strasser 2005)(Fig. 55 right) describes a similar NPR half-toning for nested iso-surfaces that works in image space after projection, much like PLUNC's z-vol program, which simulates diffuse shading by looking at the 2D gradient of the z-buffer as a post-process.

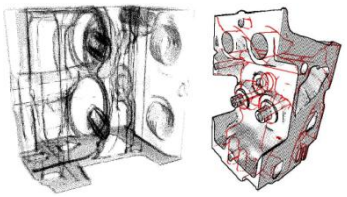

Fig. 55 Left, (Lu, et al. 2003)'s volume stippler and right, (Fischer, Bartz and Strasser 2005) renderings of the engine block data.

One of the most complete rendering systems is described in (Svakhine, Ebert and Stredney 2005), a paper nominally on applying illustrative motifs to volume rendering. This paper provides an overview of their complicated volume rendering pipeline and the large number of user interface widgets required to interface with it. Few implementation details are presented here, but more details can be found in their earlier papers. The goals of the Ebert/Svakhine system are somewhat similar to the goals outlined for Model Guided Rendering.

1. To highlight and show structure near the focus
2. To remove occluding material
3. To use simple rendering for context areas.

(Svakhine, Ebert and Stredney 2005) also suggests that model guidance can be included in their pipeline, but they do not explain how. They instead focus on implementing complex multi-dimensional transfer functions. Like mgrView, their system has a hardware accelerated object-order core, and it runs at comparable speeds: 20 fps preview and 4 fps sweetened.

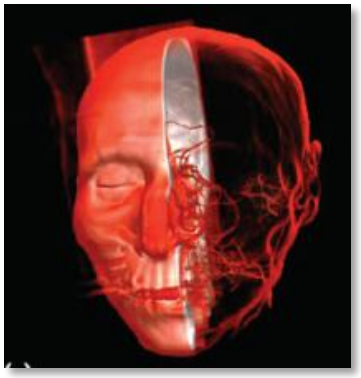

Fig. 56. Image from (Svakhine, Ebert and Stredney 2005)

Independent Image Scene Composition

The problem of occlusions in volume data was immediately recognized in Levoy's original work when he suggested that his algorithm could display all of the information in a volume only if the data was "monotonically increasing along the ray". This is, of course, unlikely to be true for any natural image (as Levoy recognized), especially for an anatomic CT image where skin << skull >> gray matter. In radiography this effect is seen when denser objects such as bone obscure all less dense objects both in front and behind them. In DVR this problem is even worse because particular transfer functions can be defined that additionally allow dense features to be obscured by less dense features in front of them. As a degenerate example, consider a transfer function that sets air to full opacity.

A variety of tools have been proposed to get around this "monotonicity proposition" without taking recourse to explicit external information such as the "importance rendering" described in (Viola, Kanitsar and Groller 2004) and (Borland, et al. 2006). (Importance rendering is discussed later when similar MGR methods for scene composition are proposed). A few of the more interesting data or scene geometry driven suggestions, along with mgrView's simple implementations of them, are discussed here.

Levoy's original insight was that **high gradient magnitude regions corresponded to surfaces** in the image and were therefore more likely contain interesting shape information than low-gradient magnitude regions. For the same reason that the gradient direction could stand as a suitable proxy for a normal direction, the gradient magnitude itself could stand as a measure of how interesting this voxel is likely to be in the scene. (M. Levoy 1990) gave a straightforward image-order implementation of this with a transfer function that accumulates color separately from opacity. Implementing the same function in GLSL is simply a matter of using a fragment program (recall Fig. 50) to modulate the CT intensity by the local gradient magnitude, as shown in Program 3.

```
// mgrView px shader for gradient magnitude opacity modulation
main (void ) {
  gl_FragColor = texture3d( source_im, world_coordinate );
  vec3 gradient = texture3d( gradient_im, world_coordinate );
  gl_FragColor.a *= length( gradient );}
```

Program 3 GLSL with per-pixel gradient magnitude opacity modulation as in (M. Levoy 1990).

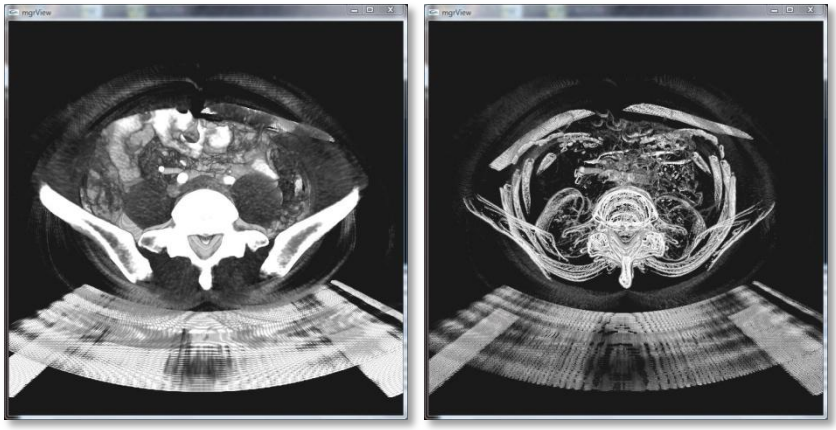

Fig. 57 Left, a standard view of an abdomen data set in mgrView. Right, the same view with specularity-based opacity modulation.

A variety of methods for surface based scene composition are formalized in (Diepstraten, Weiskopf and and Ertl 2003). One of the most intriguing suggestions in Diepstraten's treatment is to use areas with high specularity – surfaces that are relatively flat and oriented towards the viewer, to modulate transparency. This is also simple to implement for a volume in a shader, as shown in Program 4 with results in Fig. 57. Here specularity is approximated by a diffuse term using the gradient direction as a proxy for the local surface direction and the camera position as the light direction. This approximation picks out the same kinds of interfaces that are flat and oriented towards the viewer in the volume data. The effect is quite close to the earlier described "contour shader" for volumes.

```
// mgrView px shader for specularity-based opacity modulation
main (void ) {
  gl_FragColor = texture3d( source_im, world_coordinate );
  vec3 gradient = texture3d( gradient_im, world_coordinate );
  gradient.unpack();  // Recover signed values
  float diffuse = dot( normalize(camera_position), gradient );
  gl_FragColor.a *= (1.-diffuse);
  // Could also use the complement for "lighting" the volume w.r.t. the
  // camera, i.e., gl_FragColor.rgb *= diffuse;
}
```

Program 4 GLSL with per-pixel specularity modulation as in (Diepstraten, Weiskopf and and Ertl 2003).

Distance based opacity modulation is proposed in several sources, including (Lu, et al. 2003) and (Bruckner, Grimm, et al. 2006), which has possibly the most compelling results shown in Fig. 58. mgrView's fragment shader can be simply extended with similar functionality by using Program 5 with results such as those shown in Fig. 59.

```
// mgrView px shader for distance-based opacity modulation
main (void ) {
  gl_FragColor = texture3d( source_im, world_coordinate );
  float z = ((gl_FragPosition.z/gl_FragPosition.w) – near) / (near – far);
  gl_FragColor.a *= 1.-z;}
```

Program 5 GLSL with per-pixel opacity modulation from distance.

Fig. 58 (Bruckner, et al. 2006) uses cut-away views driven by distance from the viewer to maintain a visual context.

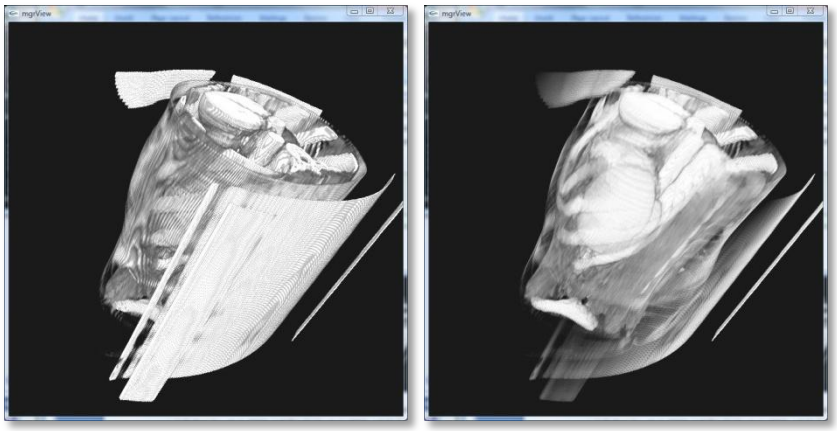

Fig. 59 Left, a scene rendered normally in mgrView. Right, the same scene with per-pixel opacity modulation from distance as in Program 5. The table and ribs have been removed, allowing a clear view of the kidney. The effect is quite striking when interactively rotating the object.

3 Model Guided Appearance for Medical Images

Following (Seligmann and Feiner 1989), discussed earlier, MGR's methods can be roughly divided into those concerned with **scene appearance** and those concerned with **scene composition**. This chapter presents tools for designing scene appearance by shading important structures in a scene to meaningfully reflect their anatomic function. This can be done either by visually labeling and orienting regions with a texture or according to atlas colors, or by superimposing data drawn from another more suitable source, such as a cross-modal mapping showing MRI data within a target region in the context of geometrically accurate CT data elsewhere (Fig. 60).

Fig. 60 Male pelvis scene rendered primarily from CT data but with red tinted MR data mapped into the prostate region to show distinction between soft tissue types within the prostate.

Both of these approaches rely on being able to create regional mappings from multiple image sources into a target patient rendering. MGR does such mappings online in an object-order context by using a combination of programmable graphics hardware shaders that add relatively little computational overhead to the rendering. The algorithms are controlled by a data structure called a "scene catalog". A scene catalog is a description of where every important object in the

scene is, how it is oriented, and how it corresponds to other similar models in other data. Scene catalogs also form the basis for associated model guided texture synthesis methods.

A scene catalog for a rendering consists of two parts.

1. A world-to-model coordinate transform, called an "X2U map", where X is a world-space (xyz) triplet and U is an object coordinate (*uvt*)=(along,across,through) with an additional possible fourth component, o, for object label in multi-object scenes.
2. One or more "rules" for transforming each region's model coordinates into a texture space or the world space of a related image such as a color atlas.

Model-to-texture transform rules include the following cases.

1. World-based, which uses only the *label* component of the X2U map
2. Model-based, which uses the *label* and *model coordinates*
3. Model-to-model, which uses the *label* and *model coordinates*, as well as an inverse mapping, called a *U2X map*, to convert model coordinates back into the world coordinates of a source image

This chapter is divided into four sections.

1. The first section, **Creating a Scene Catalog**, describes in detail how to quickly convert a collection of discrete medial models fit to regions in a scene into a X2U map.
2. Rendering with regional appearances based on model or world mappings is discussed in the section **Simple Texturing for Volumes**. MGR uses simple texturing for both such things as generic oriented muscle fibers and for patient-specific synthetic textures that have been generated "in-place" for a scene. Library solid textures can be used to add surface detail to a coarse underlying segmentation by perturbing local normal directions for lighting.
3. An interesting case of a world-based rule is **2D Color Transfer from Patient Photos**, where a patient image, for example, can be mapped onto skin voxels according to a *function* of world-position. A fast method for rendering with 2D color transfer from photographs is presented in the third section of this chapter.
4. The final texturing rule, called **3D Color Transfer** in MGR, is the most complex but also the most flexible for mapping information from an additional data source, such as a color atlas or cross-modal image, into the target image space. The creation and application of U2X maps for this purpose is described in the final section.

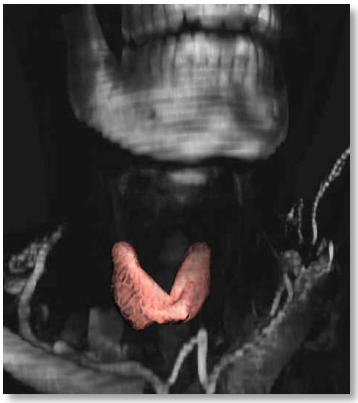

Fig. 61 Simple world-mapped solid texture applied to the thyroid region of the target patient.

3.1 Creating a Scene Catalog

MGR provides a framework for synthesizing multiple data sources into a single visualization on a region-by-region basis. Many patient-specific and atlas data sources are connected to one another though shared object-coordinates, and the rendering engine shades each voxel conditionally according to the rules described in the introduction.

Determining a governing rule and applying the relevant color-mapping algorithm for a particular voxel requires an estimate of the locally governing model's (*uvt*) coordinates at any point in world space. While the discrete medial coordinate system is well suited to transforming from parameter-to-world coordinates (the U2X transform) directly from the given medial parameters at the samples, the reverse transformation (X2U) has no closed form and typically requires an optimization driven search as described in (Han 2007). This function would be extremely expensive to generate point-by-point according to standard means.

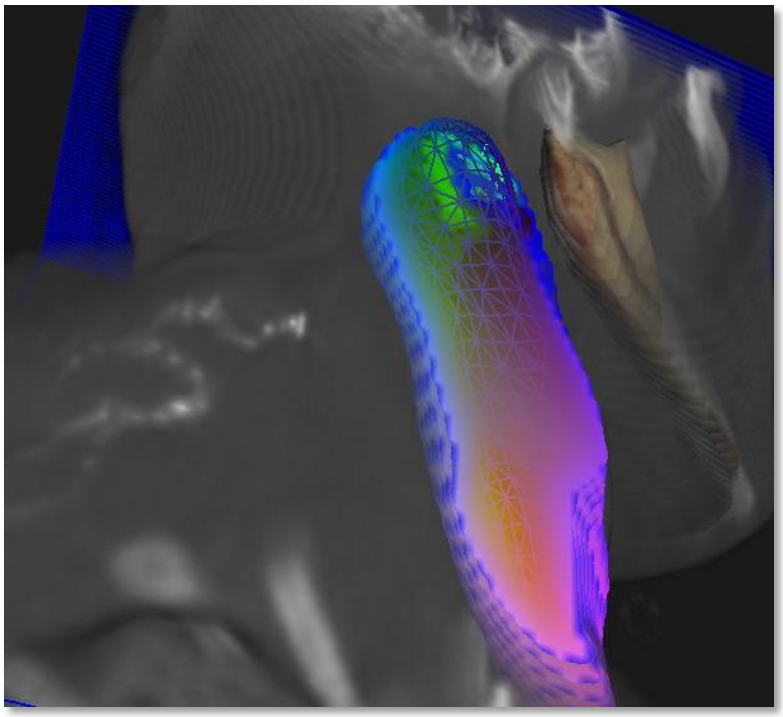

Fig. 62 Direct display of the X2U map near the right sternocleidomastoid (scm) muscle. The red channel encodes *u*, the direction *along* the object, the green channel is *v*, *around* the object, and the blue channel is *t*, the *through* direction. The boundary surface of the scm is superimposed as a similarly colored mesh.

MGR's solution is to use the graphics accelerator to do a fast coordinate scan-conversion by rasterizing densely packed geometry that has been colored by object coordinate and then clipped to individual planes. Collating these planes into a volume gives a world-space lookup table (LUT) of (*uvt*) coordinates at every voxel (see Fig. 62). In a sense, this

transform is a variant of a 3D distance map transform, but it tracks not only distance from the manifold (radius normalized distance along the spoke from the medial axis, in this case), but also the originating object label and (*uv*) coordinate on the medial sheet. This X2U LUT, along with any inverse U2X maps required for the scene, is called the "scene catalog" in MGR. Storing the scene catalog as a collection of 3D textures related to the coordinate systems of the various data sources participating in the scene allows mgrView's object-order rendering pipeline to access and interpolate model coordinates as a function of world coordinates very quickly. This novel method for being able to quickly access multiple images in model-relative coordinates has a broad range of applications in the medial shape realm where such access is thought to be the speed-limiting issue for many posterior optimization schemas.

Scene catalogs are generated from shape models that have been fit to important regions in the patient images by the methods described in section 2.2.2, *Image Interpretation*. This section begins by reviewing the details of the discrete medial shape parameterization introduced previously, and then it describes the process for computing single- and multi-object X2U LUTs.

3.1.1 The Discrete Medial Parameterization

The geometry rasterized to compute the X2U table is a collection of onion skins between the region's medial axis and boundary surface where each vertex's (*uvt*) coordinate has been encoded as color. Medial geometry and the discrete medial representation are briefly reviewed here as a basis for explaining how these onion skins are computed and colored.

Medial geometry was originally described in (Blum 1967) and has been most recently and most thoroughly described in (Siddiqi and Pizer 2008). Medial geometry describes 3D objects in terms of a skeletal surface, a 2D curved sheet lying midway between opposing boundary surfaces of the object, and a set of spokes extending to the object boundary from both sides of the skeletal surface. The medial manifold, M, of a three dimensional object has eight parameters at each 2D point (u,v): $\mathrm{M}(u,v)$ = {position (3), spoke length (1), and two spoke directions

(2 x 2)}. Some additional complexity is introduced along the edges of the medial surface where the parameterization wraps around the object.

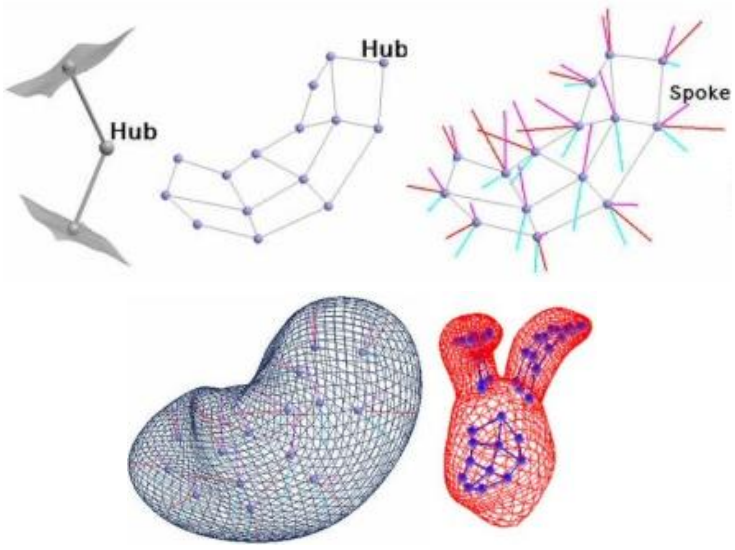

Fig. 63 M-reps. Top left, a medial sample with two equal length spokes touching opposing surface patches. Top middle, a sampled skeletal sheet with neighbor relations marked. Top right, spokes at each medial sample describe the orientation of the implied surface at that hub. Bottom left, a densely sampled surface can be interpolated from the medial samples. Bottom right, a prostate model with sub-figures defined for the left and right seminal vesicles.

The discrete medial representation (m-reps) samples the continuous manifold on a grid, yielding a set of 8-dimensional medial "hubs and spokes", which taken together act as control points for the object's volume, as shown in Fig. 63, top. Additional medial points can be interpolated according to (Han 2007), which in turn imply a denser surface sampling. Alternatively, additional surface points can be approximated directly using a modified Catmull-Clark subdivision algorithm (Catmull and Clark 1978) with additional constraints on the normal directions(Thall 2004). An object made from a single grid of medial samples is called a figure. A *single-column* grid with additional spokes implies a tube figure; a *multi-column* grid implies a slab figure. Similar parameterizations are used for both slabs and tubes, although each has slightly different methods for such things as surface generation and measuring sample regularity. Indentations and protrusions on the object are handled as attached subfigures. A figure along with any associated sub-figures is called a model, shown in Fig. 63, bottom left.

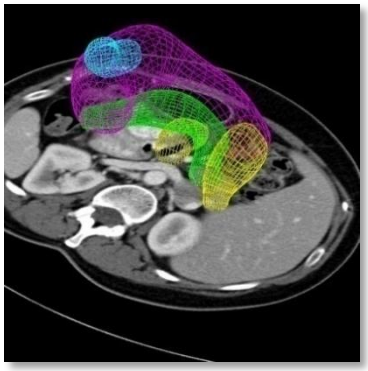

Fig. 64 Surfaces implied by m-rep parameterizations of a target patient's stomach, pancreas, and duodenum.

The sampled M implies a volume filling approximately[13] hexahedral mesh that is useful for various tasks that require volume filling coordinates, such as computing mechanical deformations according to finite element methods. There are several alternatives for

[13] These are six-sided structures that do not necessarily have planar faces.

parameterizing M. UNC's medial shape fitting software Pablo[14] uses a "single sided" representation, where u and v are the same on both the top and bottom of M and direction is determined by the sign of t (τ in Pablo parlance) with -1 on the bottom surface, 1 on the top surface, and taking intermediate values along the crest regions. MGR uses a "shrink wrap" representation, where u and t are fixed with regard to sidedness, but v wraps around the medial sheet, running from [0,0.5) on the top and (0.5,1] along the bottom, with a seam where it cycles on itself. The shrink-wrap representation is more flexible for working with skeletal rather than fully medial representations because spoke lengths and directions can differ on the top and bottom of the sheet. However there is a seam in the v parameter across the medial sheet (Fig. 65), which adds some algorithmic complexity in the next section on applying simple textures. The later section 5.2, *MGR Applications in Adaptive Radiotherapy*, describes the mapping between Pablo's single-sided and MGR's shrink wrap coordinates.

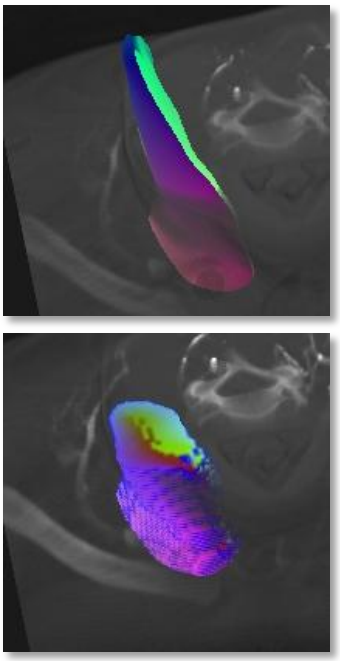

Fig. 65 Top, t=1 surface colored by (*uvt*) and bottom, cross section normal to du of the scm's X2U map. Using the shrink wrap parameterization there is a singularity in *v* (green) at the seam and across the medial sheet.

3.1.2 Computing the X2U Map

MGR's object-order pipeline is well suited to both precomputing an entire X2U map very quickly and to accessing the X2U map saved as a lookup table (LUT) in memory for its various color mapping algorithms.

A space-filling X2U map can generated by repeatedly rasterizing a large number of onion skins color-coded with (*uvt*) coordinates, with each rendering pass clipped to an individual plane through the scene. A single-object 256 x 256 x 256 X2U LUT using 64 onion-skins can be computed in less than a second on a standard desktop machine using this algorithm. The alternative approach of doing an optimization driven search at each world-space position to find the corresponding model-coordinate may take a similar amount of time to do *each* search, making this approach several orders of magnitude faster for generating a space-filling map. Program 6 summarizes the algorithm and particulars follow.

Each vertex is assigned a color (rgb) = (*uvt*) according to the shrink wrap parameterization, such that on any given onion-skin surface red ranges with u from 0 at the "bottom" to 1 at the "top", green wraps around the object with v < 0.5 on the "anterior" and v>0.5 on the "posterior" with a

[14] See the UNC MIDAG website for more information about Pablo.

seam where the values wrap along one of the crests, and blue is fixed for t < 1.0 for an interior onion-skin and >1.0 outside .

Rasterize X2U LUT
For each k = 1 to k_{max} (*z* resolution)
  Compute sandwiching planes at that depth
  Set forwards and backwards facing clip planes
  For i = 1 to t_{max} (*t* resolution, typically 32)
    t = 2.0*i/t_{max} (one radius outside the boundary surface)
    Generate the "onion skin" surface colored by (*uvt*) at each vertex
    Render the geometry inside the clipped volume to a buffer
    Merge the buffer with the working LUT

Pick a different cardinal direction for the view and repeat
  Fill in holes by taking the (*uv*) value with the smallest (*t*)

Program 6 Pseudo-code for the X2U LUT scan conversion algorithm.

A large number of onion-skins colored by this scheme are generated, as seen in Fig. 66. Vertex positions are computed as a *t*-weighted interpolation between the spoke tail and tip. That is, $X(t) = X_0+t*(X_1-X_0)$ (or, in GLSL, by the single instruction "$mix(X_0,X_1,t)$"). Fixing *t* and rendering the entire surface with such modified vertex positions produces a smoothly colored onion-skin between the medial axis and the boundary surface. Iterating this over *t* between 0 and 1 produces a set of nested onion skins. Sandwiching clip planes are used to restrict the rasterization of the nested onion skins to each plane of voxels in the LUT. The stack of planes is combined to create a volumetric LUT.

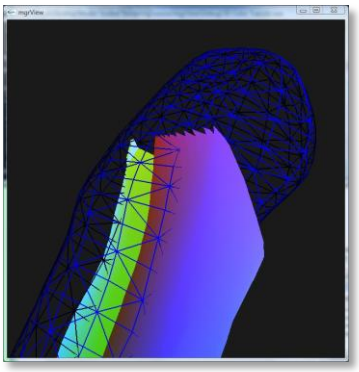

Fig. 66 A cut-away of a ten onion-skin representation of the scm. Each layer has fixed t or blue value. Each ring about the object has fixed u or red value. Each line along the object has fixed v or green value.

The initial pass can leave holes where the view direction is nearly orthogonal to the spoke direction. These can be resolved by marching the clip planes through the volume along two or three of the cardinal directions and merging the results by preserving the candidate model-coordinate at each voxel with the lowest *t* value, under the assumption that samples taken closer to the medial sheet are more likely to have the correct (*uv*) coordinate.

The X2U map may also be useful in the "collar" region of the object a small distance beyond the boundary surface. Therefore, the X2U map actually encodes values of t between 0 and 2 by packing the blue channel with t/2.0. Medial shapes generated from m-reps are constrained in such a way that no spokes can cross in the interior of the object, which implies that every interior point has exactly one possible (*uvt*) coordinate. However, it is impossible to enforce such a constraint beyond the boundary, which implies that certain exterior points may

have more than one possible (*uvt*) coordinate. The same minimum-*t* preserving method used above to merge LUTs is adopted here. This heuristic can lead to unexpected discontinuities in the collar region, so (*uvt*) coordinates assigned outside of the object should be inspected carefully.

The X2U map is ultimately loaded onto the graphics accelerator as a texture unit. Its purpose is to accept world-space (*xyz*) coordinates and convert them to model-space (*uvt*) coordinates. Because it relies on hardware trilinear interpolation, such conversions will have the effect of smoothing the data. However, since the underlying data is produced using linear assumptions (by evenly spacing the onion skins) this does not affect the results.

There are many nested loops in this algorithm, but the deepest vertex and surface operations can be compiled into a single hardware operation (an OpenGL "display list" with a varying *t* parameter to generate the nested onion skins in a vertex shader program). The tight clipping planes mean that most of the vertices are rejected early and so never contribute fragment overhead. The computation time for generating the X2U map is therefore bound either by the sampling grid (pixel size, number of planes, number of directions) or by the time that it takes to read the data out of the screen buffer and cache the control map for later use. The sampling grid need not be particularly dense; most of the examples here are 256 x 256 x 256, or 16 megapixels total. Common modern gpus have theoretical fill rates (number of pixels that can be rendered in second) measured in gigapixels, suggesting that the actual pixel processing described here could be done in a fraction of a second.

The current rate limiter on this algorithm is reading out the frame buffer many times so that the LUT can be compiled and merged in main memory. The computation is expected to be extremely fast if the result is never read out of the texture-memory, but this has not been tested yet since no present applications for MGR require generating dynamic X2U maps, for example, to support solid texturing in deforming models.

3.1.3 Consolidating a Multi-Object Scene Catalog

A scene catalog may consist of an X2U transform for each of multiple objects. However, programmable fragment shaders have hard limits on

the number of textures that they can access, so it is preferable to consolidate all of the look up tables into a single texture unit, as shown in Fig. 67.

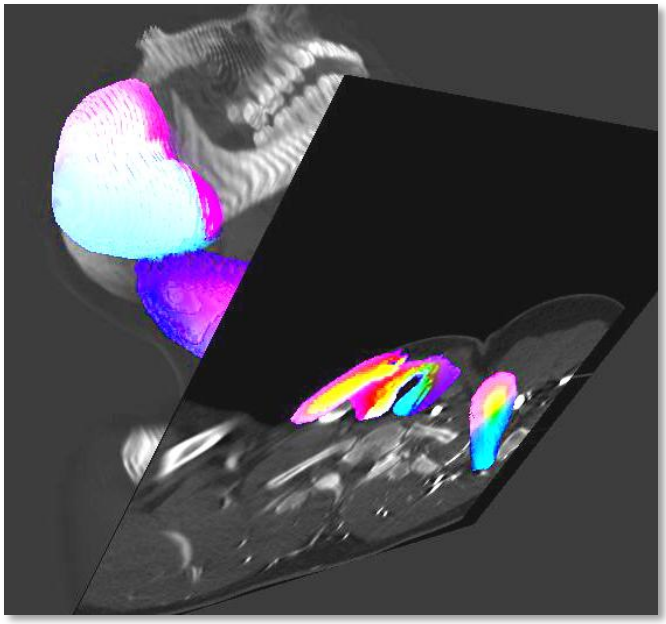

Fig. 67 A slice through a CT image colored by the underlying multi-object X2U LUT. The sternocleidomastoid's exterior values overlap with the neighboring parotid and thyroid. The object label for each region is invisibly encoded in the alpha channel.

For a multi-object scene the X2U transform for object *interiors* should have no overlap between objects in world space; therefore, each object's interior U2X entries can be OR'd into the grid without loss of data. However, the X2U map for nearby exterior objects may overlap, so a decision is made to give the voxel to the object with the smallest t value, *i.e.*, the object whose boundary surface is closest to that voxel.

This raises the problem of identifying *which* object governs for a particular world position. This is solved by attaching an object-label integer (0 to 255) into the alpha channel of a 4-channel rgba texture. Then, given a world coordinate X, the X2U function returns uvt plus the governing object label, called an (*uvto*) coordinate. mgrView includes a Matlab script that generates such combined multi-object scene catalogs.

In a single-object X2U map, since alpha is not being used for the object label, it can be used to track other variables, such as likelihood of disease or importance rank. If the X2U map is not going to be used directly for rendering but rather for texture synthesis (see next section) or another application, an arbitrary number of such additional per-object-coordinate variables can be attached to the sampling grid.

3.2 Simple Texturing for Volumes

Having fast access to reliable volumetric object coordinates makes several otherwise difficult rendering tasks much more straightforward. Assigning textures and colors inside a volumetric region or on an object's boundary surface is easily done in MGR by assigning "simple textures" based on either world or model-coordinates. The techniques for context sensitive shading described in this section have been developed in collaboration with Ilknur Kabul, who builds on some of them in her own work.

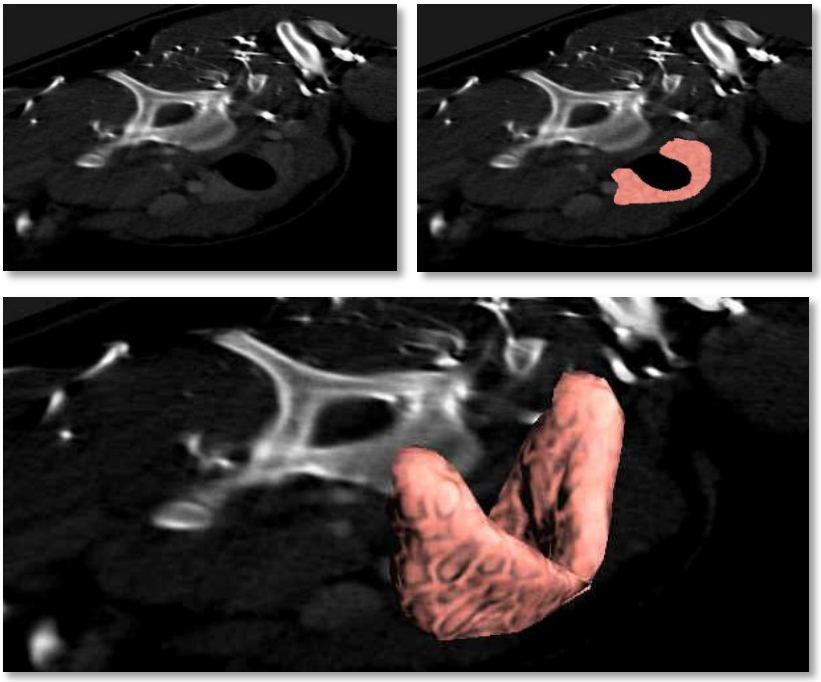

Fig. 68 Top left, the thyroid is difficult to identify in the gray data. Top right, adding a pink texture to the clip plane. Bottom, texturing the entire thyroid surface.

Simple texturing in a volume requires only the object label from the scene catalog X2U map for **world-space texturing**, or requires both the label and model-coordinates for **model-space texturing**. Regions can also be given the appearance of additional detail by **bumping the solid texture** with respect to the light direction.

Simple ("diffuse") lighting is computed by modulating the surface color by the cosine of the angle between the surface normal and the light direction. This gives the appearance of being fully lit when the surface faces the light source and falls off to unlit where the normal is orthogonal to the light direction (*i.e.*, on the contour). "Bumping" a surface involves varying the normal per fragment according the surface texture to provide extra detail in the lighting. "Bumping" a solid texture

embedded in a volume is difficult because the normal perturbation must be smooth from fragment to fragment. This is trivially achieved for 2D textures. The perturbations are computed with respect to the cardinal directions of the texture, which are smoothly varying on the surface. However, for solid textures, the model coordinates of the underlying region can be used to align a continuously varying set of tangent planes which can be used to sample the texture and compute local normal perturbations.

In principle, applying MGR's simple texture methods to a sufficiently large number of regions identified in the target image could incrementally transform even a slice-by-slice visualization from gray into a more easily comprehended Netterly equivalent, such as the slice shown in Fig. 69

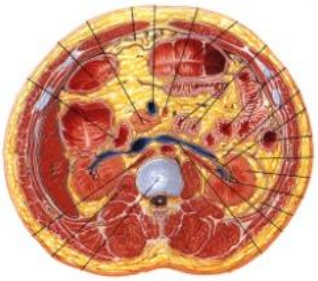

Fig. 69 Cross section drawn by Netter.

3.2.1 World- and Model-Space Texturing

Mapping from Solid Texture
1. Consult the X2U map to determine fragment region membership and object coordinates[15]
2. Determine the target texture for that region
3. Transform the object or world coordinates by a standard GL affine transform to index into texture space

Program 7 Pseudo-code for the simple texturing fragment shader.

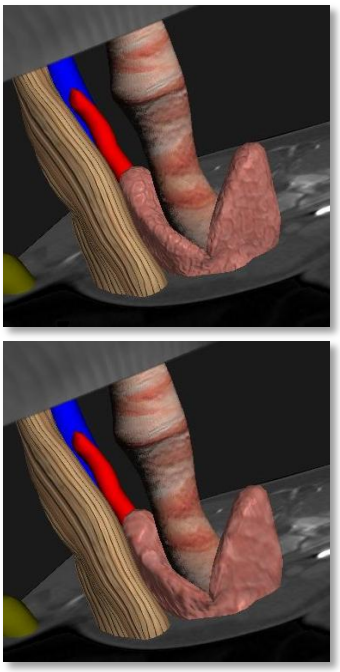

Fig. 70 The same solid texture for the thyroid with two different texture scaling factors. Top, a larger scaling factor (30); bottom, a smaller scaling factor (15) results in relatively larger features.

World-Space Texturing

The most straightforward texturing rule is to assign a texture to a region with coordinates given with respect to a world space. This can be useful for purely isotropic textures like blobs for fatty regions or the thyroid shown in Fig. 68 and Fig. 70. As each fragment of each plane through the volume is processed, the corresponding position in the x2u map is queried to determine region membership. When a fragment is determined to have a membership in a world-space textured region such as the thyroid, the world-space coordinates (*xyz*) are used directly to index into texture coordinates. The specific (*uvt*) model-coordinates at that fragment are irrelevant because the region is being treated as if it is homogeneous throughout. A standard GL similarity transform that controls the global size, orientation, and offset of the texture elements

[15] Recall from earlier footnote 9 that a fragment is a potential pixel. Sampling the X2U map determines the (*uvt*) coordinate and region membership (which anatomic type) for that fragment.

(texels) with respect to the region may still be applied. Fig. 70 shows an example of scaling the same solid texture to produce qualitatively different appearances.

Model-Space Texturing

Given model-coordinates, either directly from the surface or sampled from the X2U LUT, textures can be oriented relative to the region's along, across, or through directions by mapping model-coordinates (*uvt*) directly to texture coordinates (*pqs*). Fig. 71 shows an example of using a 2D texture with (*pq*) oriented according to (*uv*) model coordinates to imply the direction along the duodenum surface. In traditional 3D modeling, assigning such texture orientations requires significant manual editing or multiple geometric proxies, such as a set of cylinders whose parameters can be used to compute texture coordinates for the vertices they enclose. It is a significant advantage to get these coordinates essentially for free.

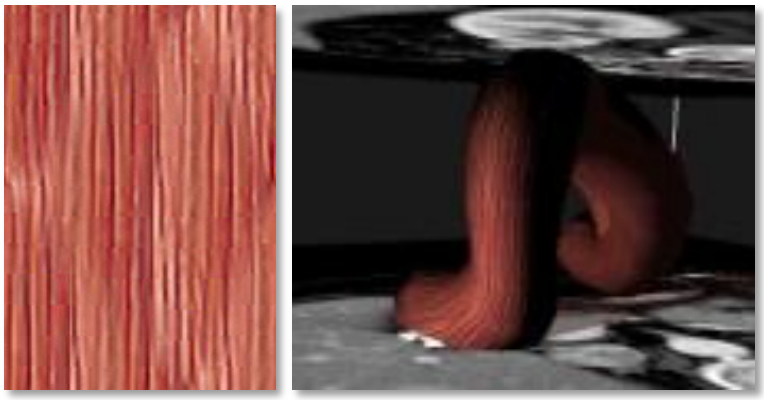

Fig. 71 Left, a 2D texture patch based on strokes from (Netter 2006). Right, the duodenum surface with the texture oriented along the *u* direction.

Seams and Singularities

The X2U map as described previously has a discontinuous seam in v (recall Fig. 65) across the medial axis as well as other singularities on the surface that necessarily occur in vector fields on shapes with spherical topology[16]. When using medial coordinates for model-space solid texture mapping, this presents a problem in interpolating values of v for fragments that are near the medial axis. In particular, a fragment that is exactly on the medial axis with a value of $v = 0.2$ on the "top" and $v = 0.8$ on the "bottom" will have an interpolated value of $v = 0.4$, which is completely wrong.

[16] By the so-called "hairy ball theorem" (Eisenberg and Guy 1979)

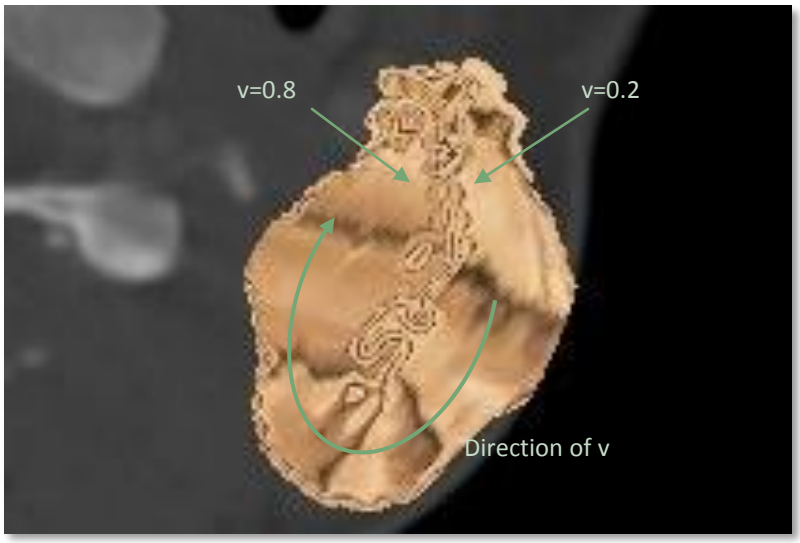

Fig. 72 Texturing across the seam in the medial sheet results in bad interpolated values of *v*.

When interpolating model coordinates on the surface, this same problem expresses itself as a seam where v wraps around from 0 to 1. This could be taken care of by using multiple compatible coordinate charts. In particular, an explicit value renumbering so that each element contributing to the interpolation has consistent coordinates – *e.g.*, that each vertex on the seam is labeled with $v = 1$ or 0, whichever is closer to the other non-seam vertices of the tile. Unfortunately, such special cases are difficult to implement in the hardware fixed function bilinear texture sampler that provides one of the main speedups to object-order DVR.

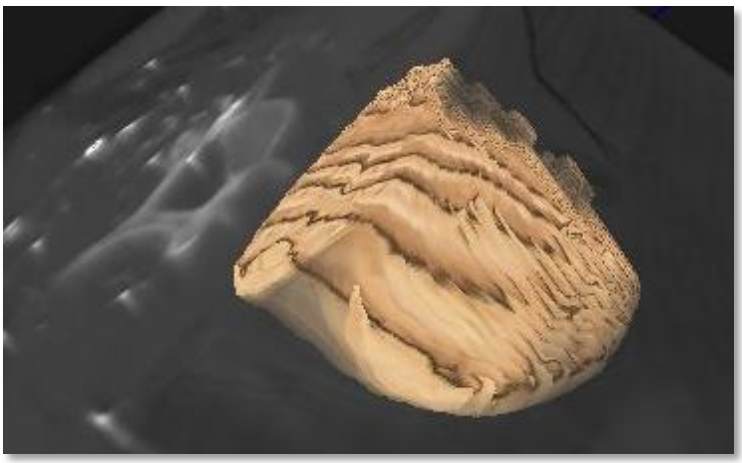

Fig. 73 Split texture mapping.

One solution is to alter the mapping slightly to reflect the seam in the texture space, somewhat like splitting a loaf of bread in half and unrolling it. Instead of letting (*uvt*)->(*pqs*) directly, the mapping is done as shown in Eqn. 6 and results in a texture such as that shown in Fig. 73. This has the slight disadvantage of requiring additional complexity in

texture generation if the texture needs to reflect interior and exterior regions. In parameter space, exterior regions are now at both the s=0 and s=1 planes, and the interior is at *s*=0.5. Additionally, the texture on the *q*-ends must be reflected across the *t*=0 plane (see Fig. 74).

$$p = u$$

$$q = \begin{cases} 2v & v \le 0.5 \\ 1 - 2v & v > 0.5 \end{cases}$$

$$s = \begin{cases} 0.5 + t/2 & v \le 0.5 \\ 0.5 - t/2 & v > 0.5 \end{cases}$$

Eqn. 6 Formulae to introduce a splitting seam into the (pqs) texture space.

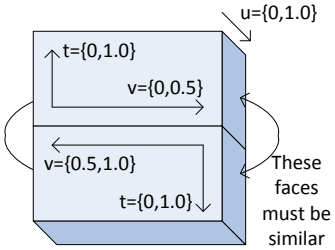

Fig. 74 Introducing a seam in the texture cube to counteract the seam in model coordinates.

Another potential solution is to interpolate values in a spherical space. For example, (*uvt*) values can be taken as coordinates in a spherical space, the values can be projected into the equivalent Euclidean space where interpolation works, and then the interpolated value can be brought back by the inverse mapping. A standard spherical map that considers (*uvt*) to be a longitude, latitude, and thickness (φ,θ,ρ) is unacceptable because it is degenerate – all elements with t=0 map to the same point in the Euclidean representation, so the original u and v values cannot be recovered by the inverse transformation. However, a variant of the spherical map called an "oblate spherical map" has the desired properties of being both cyclic in longitude (transformed *v*) and one-to-one. Fig. 75 shows an example of regular sampling in (*uvt*) projected into the equivalent "Euclidean" space, where interpolation performs as expected. Seams and necessary singularities in the *original* parameter space are preserved by this operation; the spherical space merely enables interpolation between samples across seams without requiring difficult to parallelize case-by-case rules.

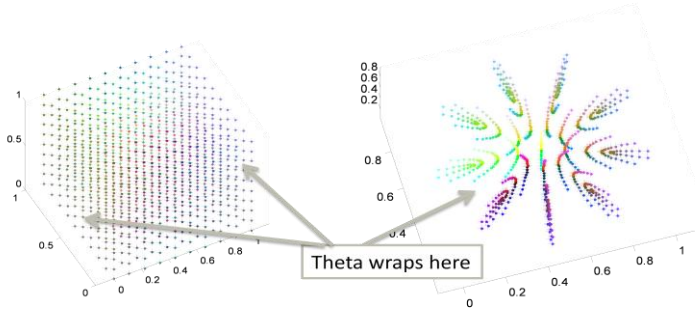

Fig. 75 Left, regular sampling in (φ,θ,ρ) taken as oblate spherical coordinates becomes a squashed spheroid in the Euclidean equivalent on the right. Interpolating theta across the seam in this space produces correct values without a conditional when mapped back to the parametric space.

This is a kind of dimension "lifting". Consider that a single onion skin in (*uvt*) space has only two parameters, *u* and *v* vary but *t* is fixed – it is a single plane in Fig. 75, left. In the Euclidean equivalent oblate spherical coordinates, it is actually an ellipsoidal surface, as expected. All three parameters (*xyz*) are used for interpolation on the onion-skin and the result is pushed back onto the original plane in (*uvt*) space. Unfortunately, this method has two major drawbacks. First, it is expensive to compute per fragment on the gpu; it requires hyperbolic and inverse hyperbolic trigonometric functions which are not implemented as single instructions on most commodity graphics

hardware. And second, it produces extremely blurry texture mappings due to limited fixed-point precision on the gpu. The oblate spheroidal equations are available online (Wolfram, Wikipedia) and are described in several primary sources including (Abramowitz 1972).

While the split-texture solution works for the current texturing applications of MGR, it is insufficient for the color transfer method described in the last section of this chapter. Determining a suitable parameterization for solid objects is an open area of research that is particularly important in the context of solid texture synthesis.

3.2.2 Solid Texture Bumping

To give the impression of a higher resolution for a region surface or cut plane, the solid texture can be "bumped" with respect to the light direction and the implied surface that is being used for lighting. Using bump or normal maps generated from 2D texture maps to simulate denser surface resolution is well understood (see (nVidia 2004), for example) although the implementation given here for solid textures is novel. The general idea of bump or normal mapping is to simulate surface detail by lighting each fragment as if the fragment were not on the surface but on a height field near the surface. Taking derivatives of the height field suggests a new normal direction at each fragment. A "bump map" stores the height field only, requiring the shader to do some local computations to determine the normal offset. A "normal map" essentially precomputes the new normal direction and stores it explicitly. The height field may come from a variety of sources including an actual higher resolution surface, but the height field is frequently derived from the underlying texture intensity itself.

Bumping a 3D texture is similar in principle to 2D bumping, but it requires some additional computation. The main difference is that instead of taking derivatives of the texture in the cardinal texture space, solid bumping requires texture derivatives taken with respect to directions on a local tangent plane. If the derivatives are computed with respect to the cardinal axes, for example, the differences sampled are likely outside or inside the surface and therefore irrelevant.

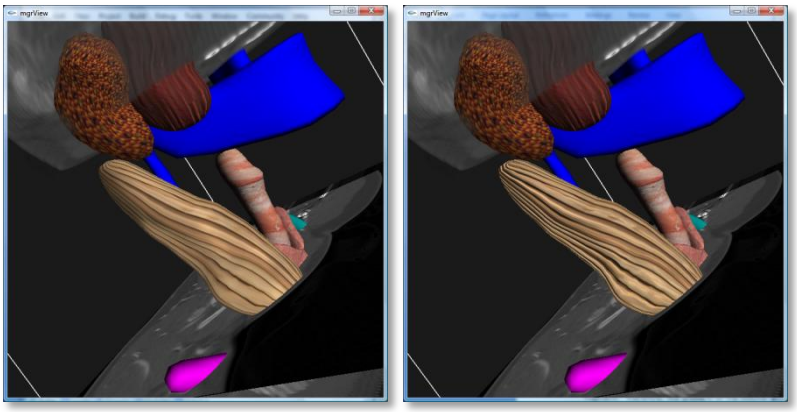

Fig. 76 Left, solid wood texture with standard diffuse lighting. Right, the same texture with a significant normal "bump" in the direction of the texture gradient.

This texture sampling requires a smoothly varying reference direction for the results to look coherent from fragment to neighboring fragment. This reference direction is the equivalent of the "tangent" direction in normal mapping, with the cross-product of the normal and the reference direction providing the equivalent of the "bi-tangent". For bumping on surfaces from shape models with object coordinates, the projection of *du* onto the tangent plane can serve exactly this purpose. For bumping on cut planes, the tangent can be determined by picking the cardinal direction with the longest projection onto the tangent plane (*i.e.*, the most nearly orthogonal to the normal).

Given directions in the tangent plane, computing the gradient by a forward difference requires only three lookups, one at the original coordinate and one in each of the tangent and bi-tangent directions. While taking gradients of large volume data sets is typically quite expensive to do online, most of the 3D textures used in MGR's sample programs are very small, so there is good cache coherence, and sampling for the gradient is fairly cheap.

Pseudo-code is given in Program 8, and an actual GLSL implementation of a solid texture bumping fragment shader is shown in Program 9. The implementation given takes an additional parameter C that multiplies the effect of the normal swing.

Surface Bumping Algorithm

1. Determine the texture gradient with respect to (du,dv) and "bump" the surface direction. Candidate surfaces include model-space surfaces, like the boundary or an onion skin (du,t), or a world-space surface such as a cut-plane (du,cp) or the view direction (x,view).
2. Sample the texture at (U+du) and (U+cross(du,n)) to compute a gradient
3. Use the gradient direction to 'tip' the normal by the gradient magnitude

Program 8 Pseudo-code for solid texture bumping.

```
// mgrView GLSL fragment program for solid texture bumping
uniform Sampler3D color_im0;
uniform float C;          // Strength of normal adjustment
uniform float psz;        // Distance to next sample in [0,1]
uniform integer MODEL_MAP;    // Use model or world coordinates
varying vec3 X, normal, du; // From the vertex shader
vec3 tex_coord, frag_color;
vec3 TipNormal(vec3 _n, vec3 _du, float sample_r ) {
  _du = _du – dot( du, _n ) * _n;  // 1 step of Gram-Schmidt
  vec3 _dv = _n.cross( _du );
  // Estimate gradient with finite difference
  float drdu = sample_r - Texture3( color_im0, tex_coord + psz*du ).r;
  float drdv = sample_r - Texture3( color_im0, tex_coord + psz*dv ).r;
  return normalize( normal + drdu*du* C + drdv*dv*C );
}
main ( void ) {
  // Flag to determine world- vs. model-space mapping
  tex_coord = (MODEL_MAP)?gl_TexCoord[1]:X;
  frag_color = Texture3( color_im0, tex_coord )
  normal = TipNormal(normal, du, frag_color.r);
  StdShading();  // Call the standard Phong shader with the new normal
}
```

Program 9 GLSL fragment shader for texture bumping with a gradient from finite differences in the solid texture. The program extends trivially to higher order differences at reduced speed.

3.2.3 Sources of Synthetic Textures

Given that MGR enables color transfer directly from an empirical color atlas source such as the Visible Human, the question arises of why use simple texturing at all? The most obvious answer is that another modality or an atlas that shows the desired property may not be available or may be of too low quality to use in the region of interest. Furthermore, atlas images are never specific to the target patient.

From the aspect of quality, as Netter realized, a library of anatomic textures frees the viewer from being restricted to a photo-realistic look based on long-dead cryosections. This is particularly valuable when composing a scene with a more “illustrated” feel or when rendering shapes that cannot be identified or have a poor appearance in the reference atlas. From the aspect of patient specificity, synthesized textures can support not only organ type and orientation, but also per-patient features such as whether particular sub-regions are healthy or sick. A patient-specific synthetic texture may also be used in a case where the image values are untrustworthy in a particular region,

perhaps due to artifacts. It may be useful to synthesize an approximate replacement texture from the target image itself and "in-paint" away the original values.

MGR does not implement any solid texture synthesis directly, but it has been designed to support a solid texture synthesis module, as is described in the related work (Kabul, et al. 2010). This subsection briefly reviews some of the background and main considerations for using model coordinates exported by MGR as the basis for solid texture synthesis.

Textures might be synthesized appropriately for either the model-mapping or the world-mapping cases described previously.

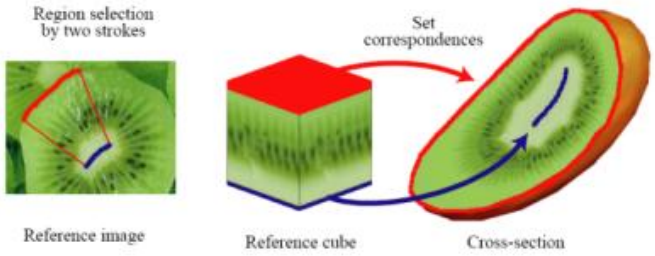

Fig. 77 (Owada, et al. 2004) creates a mapping from 2D textures to 2D cut-planes to simulate a volume texture. Though the authors do not discuss it, the proposed mappings rely on manually indicating the surface and medial axis in both shape and texture.

For **model-mapped textures**, the solid texture patch is synthesized in a Euclidean (*uvt*) space and is then deformed into the region by MGR's rendering engine according to the local object-coordinates. Such textures can naturally reflect sub-regions like deep interior (t near 0) and boundary (t near 1) as well as along, across, and through directions. Having these properties by construction is a significant advantage in texture synthesis; many high quality methods, such as (Owada, et al. 2004) shown in Fig. 77, expend substantial effort enabling a user to "pick" corresponding medial-like properties on the object and texture.

Fig. 78 Glyph packing from (Kindlmann and Westin 2006) formed the basis of the earlier rendering in Fig. 34.

The textures created for MGR so far are generally model-space mapped, but the mapping infrastructure has been designed to support **world-mapped textures**. For world-mapped textures, the solid texture patch is either completely isotropic, such as the blobby texture used for the thyroid in Fig. 68, or the texture is synthesized directly in the patient-space, using the X2U map to determine model-coordinates at every point and then using the gradient of the X2U map to determine orientation. As mentioned in the section describing scene catalogs, additional patient-specific feature channels, such as variance of local intensity or shape from normal or likelihood of pathology can also be

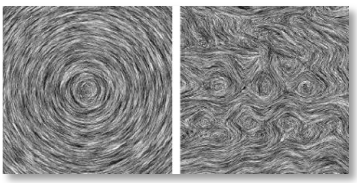

Fig. 79 2D line-integral convolution from (Cabral and Leedom 1993)

identified at each voxel. Working in such a space also allows the synthesis to be more flexible in terms of local scale – if an object packed with marbles narrows, a cardinal *uvt* space texture will shrink the same number of marbles to fill the space. A texture synthesized in patient-space however may decide to normalize based on thickness and pack fewer marbles into a narrower region.

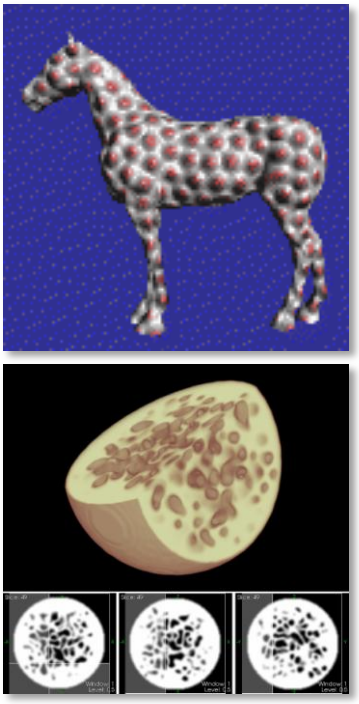

Fig. 80 Top, example of a reaction-diffusion surface texture from (Turk 1991). Bottom, volume rendering of a regional 3D reaction diffusion considered for the spongy interior of the bone (or cheese).

Procedural vs. Data Driven Textures

Procedural textures are textures that can be generated algorithmically from a limited number of inputs. The texture may be based on position alone, such as a simple solid brick wall algorithm that returns the intensity for mortar or brick depending on the (*xyz*) position given, or they may take more complex inputs. Typically procedural textures use a noise or turbulence function to simulate natural processes. Noise-driven procedural textures were originally described by (Perlin 1985) as a means of generating synthetic granite or wood. Several procedural texture methods, such as glyph packing (Kindlmann and Westin 2006) (Fig. 78), line integral convolution (LIC) (Cabral and Leedom 1993) (Fig. 79), and reaction-diffusion textures (Turk 1991) (Fig. 80) come from the realm of vector or tensor data visualization. (D. S. Ebert 1994) provides an excellent review of basic methods for procedural texture generation.

While it is straightforward to consider the gradient of the X2U map as a an input vector field for some of these methods[17], procedural texture synthesis tends to suffer from three main practical shortcomings with respect to Model Guided Rendering. First, it can be extremely slow to compute, although this can be surmounted by moving texture synthesis to an offline process. However, the second consideration is more serious: procedural texture synthesis tends to be extremely fragile; useful results are sparse relative to the many parameters required by most frameworks. Finally, no one procedural method is particularly suited for all of the many textures types that we wish to be able to produce, nor is it obvious how to "blend" between regions using a patchwork variety of methods.

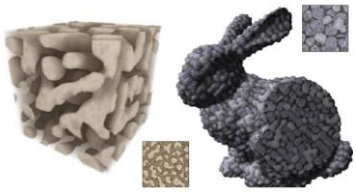

Fig. 81 State of the art exemplar based solid texture synthesis from (Kopf, et al. 2007). Several of the sample images in this document use wood or cobblestones from Kopf's solid texture library.

The method that MGR explicitly supports works by synthesizing a 3D patch from one or 2D exemplar patches. Such "data driven" rather than procedural algorithms can be thought of as generative versions of the

[17] A small caution is to use only single-object X2U maps rather than consolidated maps when computing gradients.

active appearance model means of image understanding proposed in (Cootes, Edwards and Taylor 1998). These methods similarly rely on finding distances in a reduced dimensional space of feature vectors. (Doretto and Soatto 2006)'s "Dynamic Textures" for generating smoke, waves, and other moving textures based on short image sequences presents the method in 2D+time. (Kopf, et al. 2007) presents what is currently the most effective solid texture synthesis from 2D exemplars in the literature (Fig. 81). The method suggested here is similar in principle.

Fig. 82 shows some 2D examples of early experiments done by the author in this area. The basic method is to convolve the exemplar images with a feature kernel to generate an observation tuple at each pixel that describes that pixel's intensity relationships to its neighbors. For an exemplar with P pixels and a kernel that looks at D neighbors (*e.g.*, 4-connected, 9-connected, *etc*.), this results in an D x P observation matrix.

A proposed solution is initialized to noise. Over several iterations each pixel in the solution is examined under the same feature kernel to generate another set of observation tuples. At each iteration, the nearest neighbor of each solution observation is found in the observation matrix, and the intensity of the solution pixel is nudged towards the intensity of the corresponding exemplar pixel. This approach to data-driven texture synthesis is summarized in Program 10.

Basic Exemplar Based Texture Synthesis

1. Apply a 'feature kernel' of size D to an exemplar image with P pixels to build a DxP observation matrix
2. Initialize S pixels of the solution to noise
3. For each pixel in the solution:
   3.1 Find the most likely feature in the observation matrix
   3.2 Step towards that representative intensity

Program 10 Basic exemplar-based texture synthesis algorithm

This basic method is very slow, taking on the order of hours to compute the example shown in Fig. 82 using a basic Matlab implementation. Scale spaces, dimension reduction, clustering, and search trees can be used to speed the same implementation up to require on the order of minutes. Fig. 82 middle also shows an example of the coarse (small D and S) and fine (large D and S) scale stages. Dimension reduction can be done by applying principal component analysis (PCA) to the observation

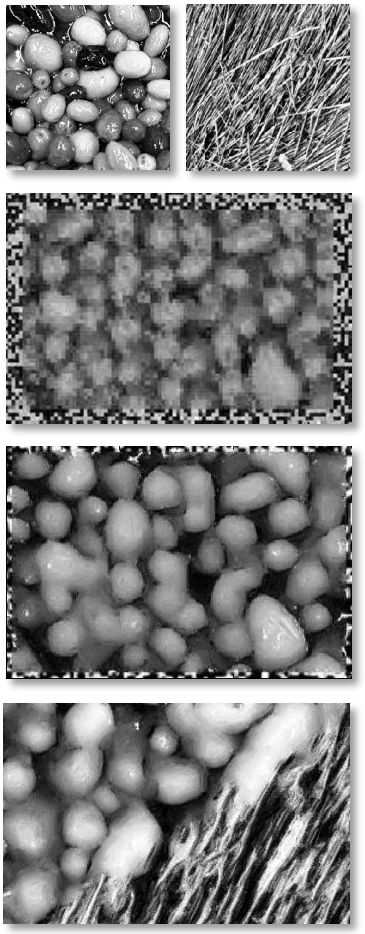

Fig. 82 2D multi-exemplar based single channel texture synthesis at multiple scales. Top, two exemplar textures, possibly for fat blobs and muscle fibers. The middle two images are endpoints of single exemplar synthesis at multiple scales. The coarsest scale took 10 seconds for 10 iterations. The finest took 10 minutes for 10 iterations. Bottom, a synthetic texture that blends the exemplars between two regions.

matrix to reduce the size of D. Clustering by k-means restricts the number of candidate pixels (P) that must be checked for each of the S pixels. And a local search tree within each cluster speeds up the nearest neighbor search.

The information encoded in the X2U map, namely the local category (object type, surface or interior, sick or healthy), orientation, and scale, can be used to seed a more sophisticated synthesis. Local orientation and scale information can either be applied *post hoc* at render-time (producing textures suited to the so called "model-mapping" method discussed earlier) or in-place at synthesis-time by rotating or resizing the feature kernel as it is applied to each individual pixel in the solution (the so called "world-mapping" method).

Applying category information to show different textures in different regions, for example to blend between sick and healthy textures or between surface and internal textures, is somewhat more complex because it requires a method for blending between exemplars. Several methods are possible depending on at what stage the blend occurs. One possibility is to completely synthesize two alternate textures and then blend them in the final image space, another possibility is to compute several blends of the exemplars themselves and generate multiple compatible observation matrices for the blend regions. The result shown in Fig. 82 was done by combining both exemplars into a single large D x 2P observation matrix and appending an additional category coordinate onto each observation. The blending was then done in the algorithm itself by blending the category coordinates in the solution and running the algorithm as normal.

The model guided texture synthesis engine (MTS) described in (Kabul, et al. 2010) works as a companion to MGR. It extends the data-driven texture synthesis framework described with multiple variants for generation and blending, 3D color texture synthesis, and other types of feature guidance, such as local edginess. A control language interface is being developed to allow MGR to request synthetic anatomic textures from MTS with regional control of exemplar type, color shift (*e.g.*, tendons are white muscle), and relationships to local model orientations and scale (*i.e.*, hypertrophic structures scale texture elements, and hyperplastic structures pack additional texture elements into the space). 2D exemplars suitable for synthesizing anatomic

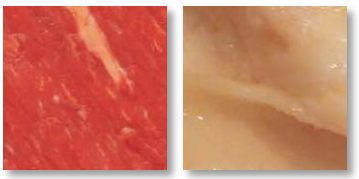

Fig. 83 Candidate exemplars for muscle (left) and fat (right) from the Dosch Design website. (www.doschdesign.com)

structures with MTS can be taken directly from illustrated sources such as (Netter 2006) or from specifically designed anatomic texture catalogs such as the one found at Dosch Designs (Fig. 83). Fig. 84 shows an example of a solid color texture generated by MTS guided by model-coordinates taken from the scm.

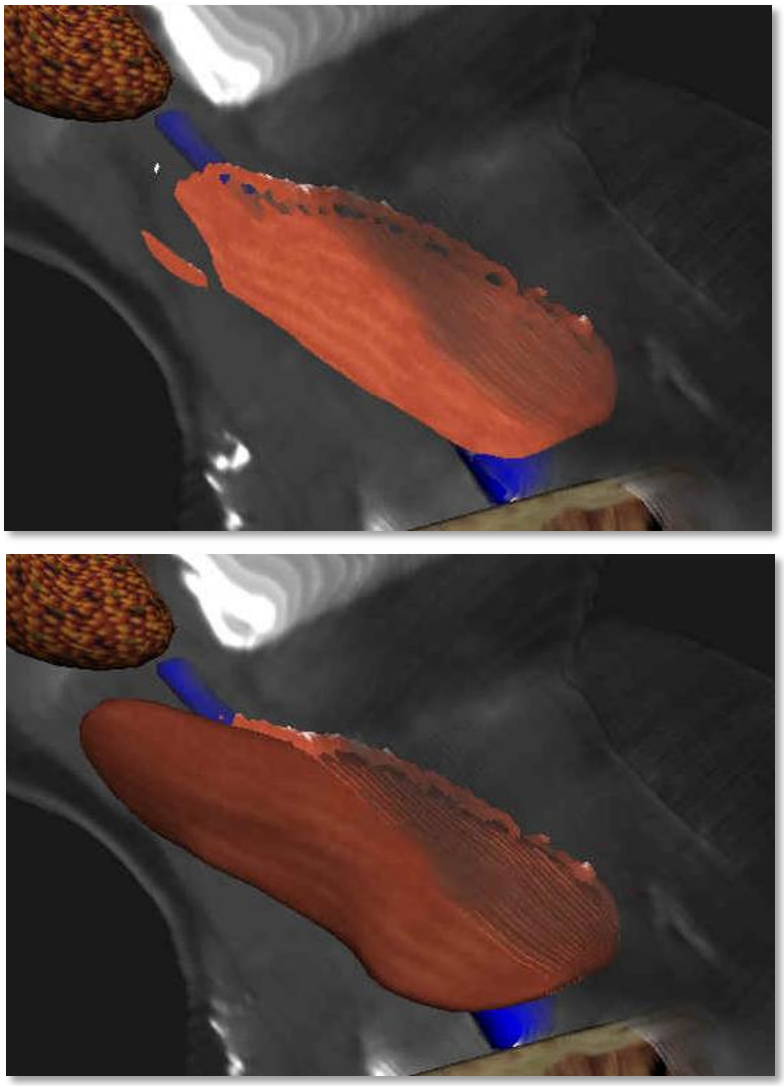

Fig. 84 Top, slice through oriented solid color texture generated by MTS for the scm region. Bottom, the same texture on the region's boundary surface with standard diffuse lighting.

3.3 2D Color Transfer from Patient Photos

Whereas collecting patient-specific 3D color volumes can be quite invasive, collecting patient photographs is quite simple. Cameras have become a common treatment room accoutrement for patient setup and monitoring, but they have been overlooked as a data source for 3D visualization. This section describes an interesting application of a world-mapped texture by superimposing patient photography onto a target 3D image (Fig. 85). Renderings of such registered data can serve to show relationships between internal structures and visible surface features. Animating anatomic change with respect to a fixed surface, or animating surface images with respect to fixed anatomy can further expose such relationships over time. Potential applications of this technology include improved 3D procedure planning, reconstructive surgery, improved diagnosis of surface pathologies or skin reactions, and patient setup, which are discussed later in chapter five, *Bringing MGR to the Clinic*.

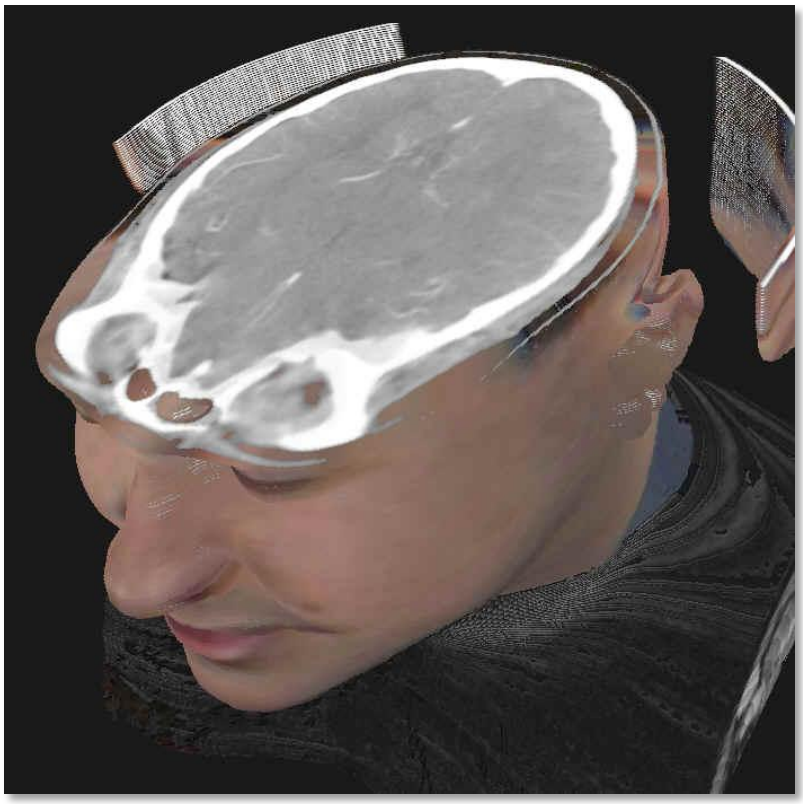

Fig. 85 A CT+photograph fusion rendering using the author's photograph and a research patient's CT scan.

3.3.1 Method for 2D Color Transfer

One or more photographs of a patient are captured when they are scanned. The photographs can be registered to the 3D scan either through explicit calibration or by feature-based estimation of a camera pose relative to the imaging device. When a 3D view of the patient data is volume rendered, the surface pixels are found either implicitly by considering a function of intensity like gradient magnitude, or explicitly by querying the scene catalog. Then the photographic data is referenced according to the camera model to determine the relevant surface texture to apply.

Once the decision to photo-map a voxel has been made, the core of the rendering method is the transformation between a 3D coordinate and a patch of a photographic image. Different methods must be used depending on whether the photographic source is configured as a **cylindrical image** or as a standard **planar image**.

For the purposes of animating longitudinal surface changes, surface appearance may be assigned by blending multiple photographs according to when they were captured.

The basic rendering algorithm is described in detail in the following section. Program 11 summarizes the algorithm and it is diagrammed in Fig. 86.

Photo-Mapping Algorithm
1. For each voxel, determine membership in skin/surface of interest:
   Implicit (intensity threshhold + gradient)
   Explicit (voxel label)
2. If not of interest, pick a value according to the DVR transfer function
3. Otherwise, for **cylindrical images**
   3.1 Compute image coordinate as a non-linear function of stretch, rotation, axis, and world-coordinate
4. Otherwise, for **multiple planar images**
   4.1 Determine which image governs shading by dotting gradient dir and camera dir
   4. 2 Pass the world-coordinate through the appropriate world-to-image transform

Program 11 Pseudo-code for the photo-to-volume mapping algorithm.

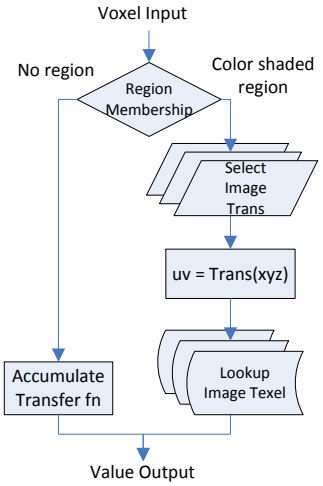

Fig. 86 Diagram of photo-mapping decision tree.

Input

The method described is flexible with respect to its input. 3D images may be of any modality (computed tomography, magnetic resonance imaging, etc.) but all require a method for identifying voxels subject to

photo-mapping. For CT or MRI combined with surface imaging, where there is a well defined intensity different between air and the patient, identifying surface voxels can be a straightforward application of edge-finding. For photo-mapping laparoscopic or endoscopic images onto internal regions, this may require 3D path information. For photo-mapping anatomic images such as a reference image of a liver, this may require more sophisticated image segmentation.

Photographic images may be of two different types, **Atlas Images** or **Patient Images**. **Atlas images** are pictures of a reference patient which can be carefully built once offline, then mapped many times onto different patient scans. Such images are useful for adding race and sex cues to visualization or for identifying the surfaces of internal anatomy, such as the liver or a muscle. **Patient images** are unique to a given patient and require specific processing between capture-time and render-time. Patient images are required for any patient-specific diagnostic or longitudinal studies. Some applications rely on non-visible light patient photography, such as thermographic imaging, but the same principles hold. Atlas and patient images can be easily combined, if, for example, a clinician desired a rendering with patient-specific surface features but also with a liver atlas image embedded in the 3D data.

3.3.2 Rendering From Cylindrical Images

The easiest case to consider is photo mapping from a cylindrical image of the patient. Cylindrical images tend to be harder to collect and more expensive to transform into CT coordinates, but the decision tree is straightforward and the renderings are ultimately much more robust to mismatches between the photograph and the 3D geometry. Consider that the image in Fig. 85 is the author's photograph mapped by this method onto a completely *different* patient's CT data, and the image in Fig. 87 uses a non-cylindrical image of the monkey.

Fig. 87 Top, a capuchin monkey MRI with a pseudo-cylinder photomap from a reference image, bottom.

By aligning the cylinder axis of a cylindrical photo of the patient captured at treatment time and the cylinder axis of any patient scan, the two images can be registered with a small number of variables. As the 3D image is rendered, the voxels corresponding to skin are identified by image analysis methods and are "painted" using a mathematical cylinder transform to query a patient photograph for the relevant skin texture. This section describes the mapping in detail.

Using an OpenGL object-order renderer, the patient's 3D CT image is bound to texture unit 1, and a patient photo is bound to texture unit 2. For simplicity, suppose that the 3D patient image has been preprocessed and includes a label for skin voxels in the alpha channel. For realistic lighting, further suppose that the gradient direction and magnitude of the 3D patient data have been precomputed and loaded as rgba values to a third texture binding.

A custom fragment shader (recall Fig. 50) then processes each fragment. Each fragment has a world position, X=(x,y,z), that is used to retrieve I(X) = (h,s), where h is the local intensity value of the patient image in Hounsfield units and s is a binary label for skin or not skin. If the fragment is not skin, it is composited normally. If it is a skin voxel, the cylindrical coordinate of the patient photo is calculated, and that texel is used to determine the surface value. If desired, additional lighting can be calculated by using the local gradient as a proxy for the normal direction at this fragment. The gradient can be easily computed online by finite differences by testing local texture values. However, in practice, these additional tests can be quite slow, and because the gradient direction is fixed within a scalar even under intensity windowing, it is usually easier to simply pre-compute the gradient and load it as a color texture, as discussed previously. The gradient direction at the skin fragment can then be linearly interpolated and renormalized, which requires only a single texture lookup.

A cylindrical texture map is a restricted case of the spherical texture map first described in (Blinn and Newell 1976). For a simple cylinder mapping, Cartesian coordinates (x,y,z) can be converted to cylindrical coordinates (ρ,θ,z) according to Eqn. 1. For a general cylinder map, additional parameters for stretch (κ), rotational offset (ϕ), axis origin (x_0,y_0,z_0), and axis direction (x_1,y_1,z_1) can be included. Capturing the cylindrical image relative to the imaging or treatment couch with the cylinder axis aligned along direction that the patient will move through the scanner eliminates the axis direction parameter, so only five parameters (κ,ϕ,x_0,y_0,z_0) are required. For a fixed capture device all five parameters can be determined empirically and will remain fixed for all subjects. The world-to-cylinder transform is non-linear and requires some trigonometry, which can be slightly more expensive to implement on a gpu but still requires relatively few operations to compute.

$$\rho = \sqrt{x^2 + y^2}$$

$$\theta = \begin{cases} \arccos\left(\frac{x}{\rho}\right) & 0 \le x \\ \pi - \arccos\left(\frac{x}{\rho}\right) & 0 > x \end{cases}$$

$$z = z$$

Eqn. 7 Formulae to convert from Cartesian coordinates (x,y,z) to cylindrical coordinates (ρ,θ,z) given aligned origin, offset, orientation, and stretch.

Ultimately, for a cylindrical image that is aligned with the central axis of the 3D image, such as the one shown in Fig. 89 (bottom), the pixel coordinates (p,q) are ($\theta/2\pi+\phi$, κz). Cylinder mapping implemented as a GLSL fragment shader is shown in Program 12, and Fig. 85 shows a resulting image.

```
// mgrView Fragment GLSL for Volume Rendering + Photo Mapping
#define ONE_OVER_TWO_PI 0.159154943
varying vec3 light_dir;
uniform sampler3d patient_im3d, patient_im3d_gradient;
uniform sampler2d patient_photo;
uniform float kappa, phi;
uniform vec3 axis;
vec3 CylinderMap( vec3 point, vec3 axis, float kappa, float phi ) {
  float x = point.x - axis.x; float y = point.y - axis.y; float z = point.z - axis.z;
  float rho = length( vec2( x, y ) );
  float theta = (y>=0.0) ? acos( x/rho ) * ONE_OVER_TWO_PI : 1.0 - acos(
x/rho ) *
    ONE_OVER_TWO_PI;
  theta = theta + phi; theta = theta - floor(theta);  // Cycle
  z = kappa*(1.0-z);
  return vec3( rho, theta, z );
}
void main() {
  vec4 source_color = texture3d( patient_im3d, gl_TexCoord0.xyz );
  float intensity = source_color.r;
  // Skin membership is encoded in source.alpha
  if ( source_color.a != 1.0 ) color = vec3(intensity);
  else color = texture2d( patient_photo, CylinderMap( gl_TexCoord0.xyz,
axis, kappa, phi ).zy );
  gl_FragColor = vec4( color, source_color.r );
}
```

Program 12 GLSL fragment shader program for basic 3D+photo fusion rendering. The rendering in Fig. 85 was produced using this program.

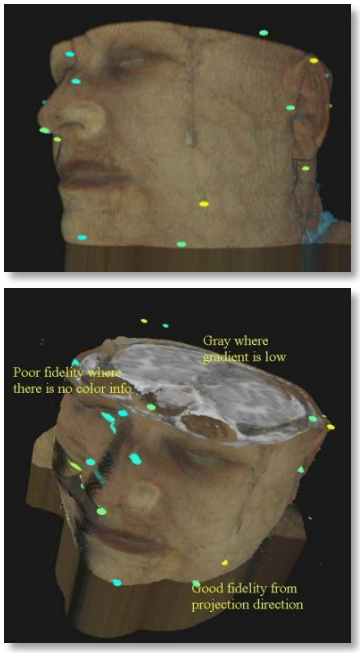

Fig. 88 Top, a synthetic view of the Visible Human from a known camera. Bottom, the synthetic photograph pushed back onto the target patient's 3D image using a direct planar mapping.

3.3.3 Rendering From Planar Images

The cylindrical image approach outlined above pushes the method's complexity into the generation of the cylindrical image, which can be, in turn, answered with hardware. Cylindrical images may be captured directly using a specialized slit camera (Fig. 89, right) or by blending photos from an array of standard cameras positioned around the patient as discussed in the next section. However, similar results can be achieved using planar images directly. Mapping from world space to photo space given a single image from a calibrated camera is a simple variant of the common world-to-image transform used in computer graphics projective mapping such as the OpenGL 4 x 4 "Model-view-

projection" matrix. This matrix multiplies a homogeneous world-coordinate and the result reduces to a 2D image coordinate. Such matrix multiplications can be done quickly and inexpensively on a gpu.

A camera's intrinsic properties (field of view, center point, pixel size, skew) can be derived empirically using a tool such as the Camera Calibration Toolbox for Matlab (Bouguet 2008). The extrinsic properties or "pose" of a fixed camera can be estimated similarly or determined mathematically from the rig or gantry geometry. These camera calibration parameters can be directly transformed into a standard OpenGL-type 4 x 4 matrix transform. The ARToolkit (HIT Lab 2007) includes an open source version of such a transform for reference purposes.

While simple planar images are easier to collect and cheaper to transform for rendering, planar images are much more sensitive to measurement error and mismatches between the photograph and the 3D geometry. Cylindrical image maps have few parameters, and the parameters have intuitive relationships with the projection. Thus the user can easily manually improve a mapping to account for mismatches. Projective image maps have many parameters (position, orientation, field of view, skew, *etc*.) and multiple conflicting ways to achieve similar results, such as moving the camera vs. zooming the field of view. This precludes the user from doing significant manual alignment.

When using multiple cameras to collect a set of planar images, such as an anterior and a lateral photo, the most relevant camera can be selected for each fragment by comparing the surface normal (the gradient direction) to the camera view direction. Assuming that the camera that is most closely aligned with the surface normal will have the best texture information, the maximum dot product with the gradient over all camera view directions will serve to distinguish a single best camera. A slower but smoother approach is to blend texture information from several cameras weighted by the dot product between the view and normal directions.

Multi-image projective texture mapping has several interesting non-medical antecedents. In particular, (Debevec, Borshukov and Yu 1998) is an important example of using photographs and simple geometry as a means for high quality image based rendering. (Raskar, et al. 2001) proposes a method for projecting synthetic photo-like textures using actual visible light ("shader lamps") to illuminate real world objects.

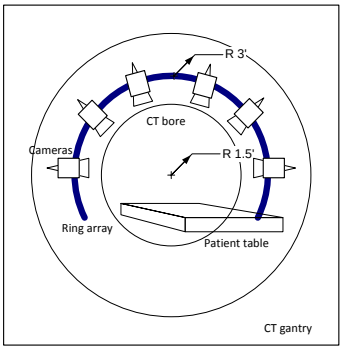

Fig. 89 Top, a schematic of the proposed 6-camera cylindrical array attachment for a CT gantry. Bottom, a cylindrical image of the author collected with a slit camera at The Tech Museum in San Jose.

3.3.4 Camera Arrangement

As suggested previously, the photographic images may come from cameras working across the visible and the invisible spectrum. Thermographic or heat photography is thought to have considerable potential in this domain. In general, planar photographs can be captured using either **calibrated** or **uncalibrated** cameras.

Calibrated cameras, as discussed previously, have known intrinsic properties (pixel size, field of view, skew, etc.) and known position and orientation (called "pose", or extrinsic properties) with respect to the same reference frame used by the 3D modality. Photographs covering multiple angles may be generated sequentially from a single camera moved serially around the patient into several known positions (Fig. 90), or in parallel from an array of cameras mounted around the patient that can be activated simultaneously (Fig. 89, top). The parallel camera array has the advantage of limiting patient motion between frames and motion between being photographed and entering the 3D scanner. The UNC Radiation Oncology clinic has proposed building such an array fit up against a CT scanner gantry so that it can generate the cylindrical image directly before the patient enters the bore where the 3D image will be collected. Mathematical transforms for computing cylindrical images from multi-angle photograph collections are related to those used in creating multi-image panoramas. If the cameras are calibrated and fixed in known positions, no feature matching needs to be done. We expect that the camera array proposed will achieve sub-millimeter accuracy when mapping patient photographs onto 3D CT image.

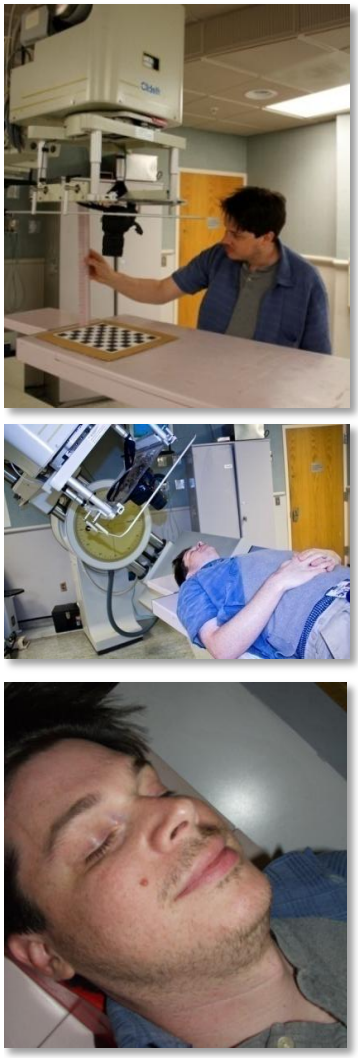

Fig. 90 Top, calibrating a camera. Middle, taking sequential multi-angle photos in a reproducible position using the accessory tray of a linear accelerator. Bottom, a single planar source photograph.

Uncalibrated cameras, such as hand-held cameras used to photograph patients for identification or for *ad hoc* charting of skin lesions, require 2D to 3D feature-match based pose estimation. Such landmark based pose estimation is beyond the scope of this paper, but the general problem of computing projective registrations is an active area of interest in the computer vision community. Consider (Forsyth and Ponce 2002) as a starting point. Given such pose estimates, images from uncalibrated cameras essentially become directly-mapped single planar images as discussed above.

3.3.5 Animating Surface Change Over Time

For longitudinal photo collections an additional parameter can vary to determine a weighting between textures sampled from two or more images. Such visual registrations might be interesting for tracking inter-fractional patient skin changes with respect to a dose field (Fig. 91).

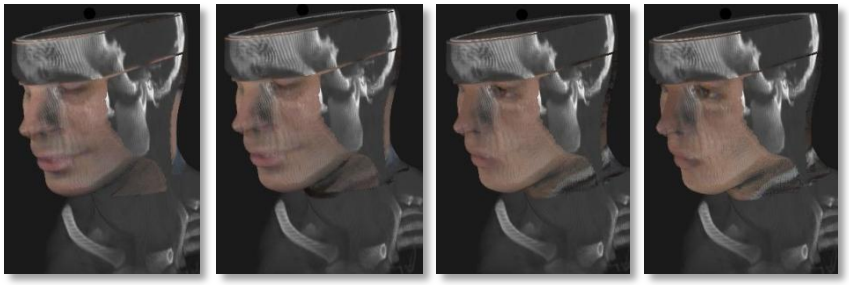

Fig. 91 Example of animating longitudinal surface changes. The left-most frame shows the author's photograph mapped onto a research CT scan; the right-most frame shows a different sample subject. Intermediate images are blends of the two.

A number of additional "key photos" can be added to an animation without appreciably decreasing the rendering frame rates, although restrictions on the number of active texture units within a fragment shader can come into play. However, this can be addressed for collections of images with the same alignment parameters (*e.g.*, collected with the same fixed camera) by loading an aligned stack of color images as a single 3D color texture unit and then trilinearly interpolating a value at the coordinate (*pqt*), where *t* identifies the particular key photo.

Further combining surface color mapping rendering with volumetric animation has possible applications in domains such as planning reconstructive surgery, as will be discussed later.

3.4 3D Color Transfer

Medial shape representations provide a useful object-relative coordinate system that can be used to compare or transfer information across images in object-relative coordinates. Such mappings can combine data from multiple images into a single target rendering, thereby allowing the user to simultaneously visualize data from whichever source is most useful or sensitive to a task at hand.

Given a scene catalog that contains both an X2U map and the inverse U2X map for a particular region, the source image model-coordinates can be transformed into the world-coordinates of a related atlas image, and then the atlas image can be indexed for the appropriate scene texture, as shown in Fig. 92. While the method can be used to map data between any two images with properly modeled corresponding regions, assigning a general label-specific appearance by transferring from a color atlas volume such as the Visible Human is an interesting target case. Fig. 93 shows an example of a target patient rendering with the mandible region colored according to the Visible Human color volume.

This section includes three topics: a **Method for Object-Based 3D Color Transfer** on surfaces and in volumes, a discussion of data representation for **U2X Maps**, and a brief review of other methods that have been proposed for **Using the Visible Human as a Color Atlas**.

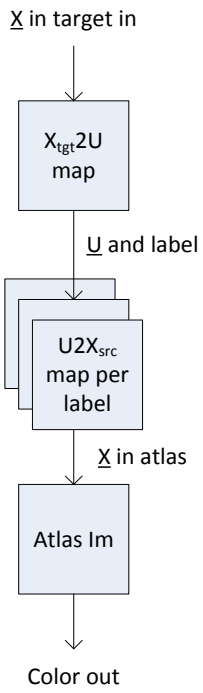

Fig. 92 Model-based color transfer pipeline. Positions in the target image are mapped through model-coordinate based functions to find the color at the corresponding position in the atlas image.

3.4.1 Method for Object-Based 3D Color Transfer

Corresponding regions in both the target and source images[18] are fit with models similar in topology (slab/tube) and sampling (grid size). As discussed previously, this can be done semi-automatically for the target image, but atlas images may be segmented and carefully fit offline. In the prostate scene used in Fig. 60 and later in Fig. 134, the source image is an MRI of the same patient taken at a different time. In the head and neck scene used in Fig. 4 and Fig. 93, the source image is a high resolution color volume from the Visible Human. In both cases, the source data was carefully segmented by hand, and then an m-rep was fit to the resulting binary label image.

[18] Terminology: Following standard convention, the patient image is called the "target" image throughout; any textures mapped into this space regardless of type are referred to as "source" images.

The m-rep fit to the source image is converted to a U2X map, the inverse of the X2U map described previously. The U2X map provides similar functionality to the X2U map – an object-coordinate U is input, and the world coordinate of that point in the atlas image is read back out. MGR represents U2X maps as an image that can be loaded as a texture unit and manipulated in a programmable shader.

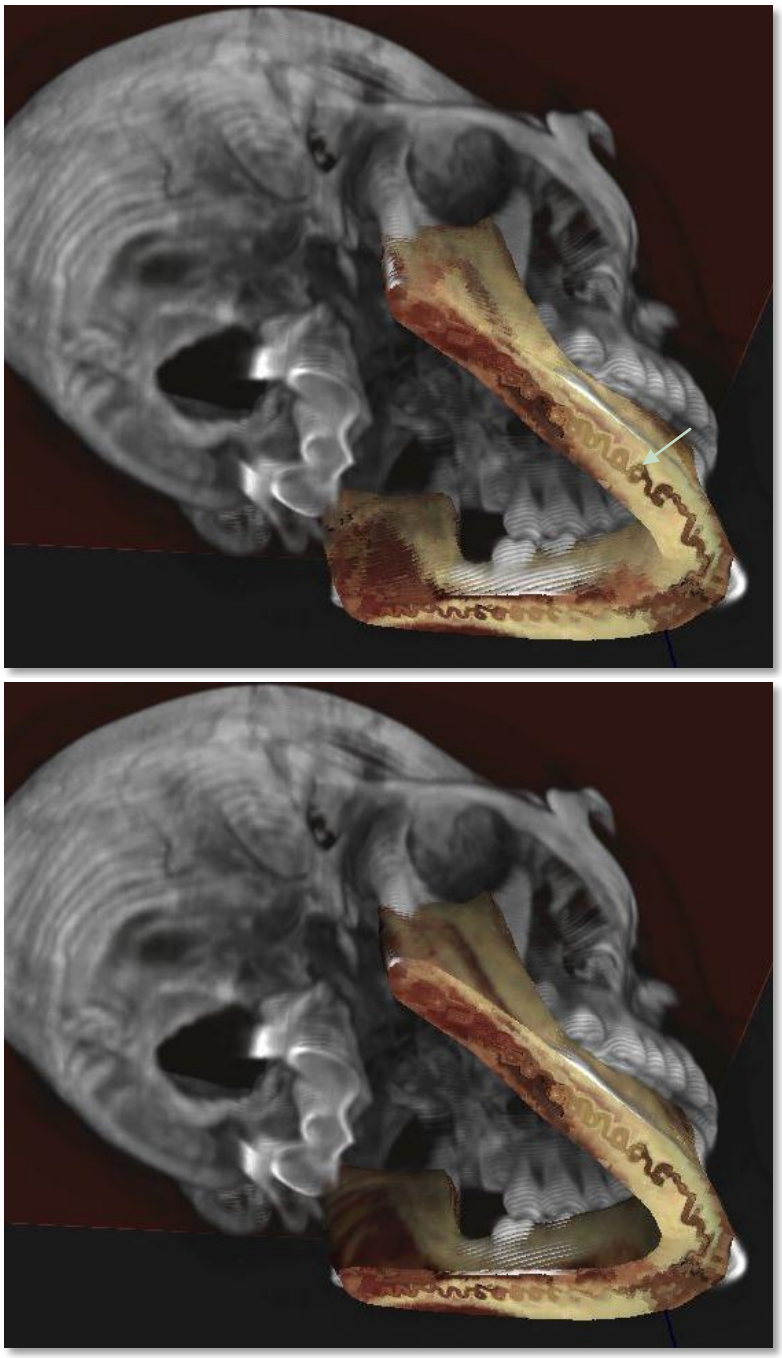

Fig. 93 Top, volumetric color mapping clipped through the mandible. Bottom, adding surface color mapping for lighting. The indicated artifact running along the medial sheet is the same parametric interpolation singularity discussed previously in the section on solid texture coordinates.

Given a source image and a U2X map, two types of color transfer are then possible, surface-only and volumetric (Fig. 93). Both rely on the same basic method, but as with simple texturing, surface-only color transfer does not require the X2U map at all since the only object coordinates of interest are given directly at the vertices. As each fragment of the target image is shaded, the X2U map or model surface gives an object-coordinate which can be transformed through the U2X map into source image coordinates. As shown in Fig. 92, this is effectively a two step texture indirection. Program 13 shows a GLSL fragment shader that does this indirection[19].

```
// mgrView GLSL fragment shader for single object color transfer
uniform sampler2D u2x_im0, u2x_im1, x2u_im, color_im0;
void main(void) {
  source_color   = texture3D(source_im0,  vec3( gl_TexCoord[0] ) ).rgb;
  vec3 uvt_coord  = texture3D(x2u_im,     vec3( gl_TexCoord[1] ) ).rgb;
  if (uvt_coord.b < 0.45 ) {
    // Inside the region of interest
    // Convert u coordinate to x in atlas
    vec3 x_0 = texture2D( u2x_im0, uvt_coord.rg ).xyz;  // Medial pos
    vec3 x_1 = texture2D( u2x_im1, uvt_coord.rg ).xyz;  // Surface pos
    vec3 x_t = mix( x_0, x_1, 2.*uvt_coord.b );          // Coord pos
    vec4 x_a = gl_TextureMatrix[4]*vec4(x_t, 1.);        // Atlas is texture[4]
    vec3 atlas_color = texture3D(color_im0, x_a).rgb;
    gl_FragColor = vec4( atlas_color, 1. );
  } else {
    // Standard shading
    float windowed_value = intensity_window( source_color.r );
    gl_FragColor = vec4( windowed_value );
  }
```

Program 13 GLSL fragment shader program for volume mapping.

If the multiple texture indirections required for multi-object X2U maps are found to be a considerable slow down on rendering speed, it is also possible to "pre-flatten" each object of the source image by prerendering it into a cardinal (*uvt*) space so that the second indirection would become unnecessary.

[19] It appears that for some OpenGL implementations, when doing surface mapping, it is important that the source texture coordinate be computed in the vertex shader and an interpolated value passed into the fragment shader. This can be implemented by moving steps 1 and 2 from Program 13 into the vertex shader and using a standard world-mapped fragment shader as described in section 3.2, *Simple Texturing for Volumes*.

3.4.2 U2X Maps

The U2X map is a data structure for a medial model associating an (*xyz*) world-space coordinate with every parametric coordinate (*uvt*). In the fragment shader the U2X map provides an interface to index atlas data according to model-coordinates (recall Fig. 92). A fragment's world position in the target image is transformed into a model coordinate according to the X2U map described previously. Those model-coordinates must then be transformed back into the world coordinates of a source image.

$$X(uvt) = t * \mathrm{u2x}_0(uv) + (1-t) * \mathrm{u2x}_1(uv)$$

Eqn. 8 Formula to calculate the world-space coordinates (*xyz*) of a model-space coordinate (*uvt*) given U2X maps at the medial axis and boundary surface.

As mentioned previously, m-reps are, by construction, designed to transform medial parameters to world-coordinates at the sample points, but interpolating a continuous transform requires complex math. In mgrView, continuous U2X maps are implemented as a pair of 2D images parameterized such that the nominal x axis is u and the y axis is v. The rgb values at each pixel encode the corresponding x,y, and z values at that (*uv*) coordinate. The first image ($\mathrm{u2x}_0$) is the U2X map at the medial sheet and the second image ($\mathrm{u2x}_1$) is the U2X map at the boundary surface. Each pair of values taken from the same (*uv*) coordinate in the two images implies the tip and tail of a medial spoke. Interpolating non-sampled values from this grid gives vertices with the same level of continuity as the mathematically implied m-rep boundary has. Taking weighted combinations of two images gives a very fast means of computing the (*xyz*) coordinates of a radius-weighted onion-skin between the medial sheet and the boundary surface (Eqn. 8). As with the X2U maps, in certain cases it is useful to extend the map beyond the surface, so that world coordinates can be estimated for the nearby "collar" region at values of *t* greater than 1.

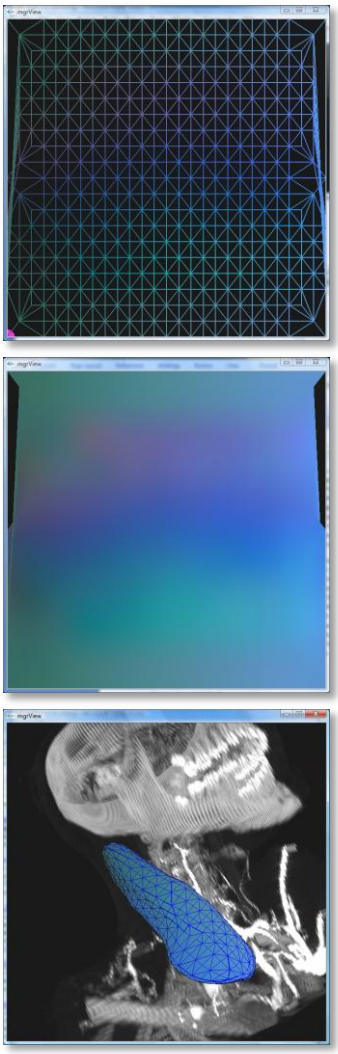

Fig. 94 The surface (t=1) plane for a U2X map of the scm shown in Fig. 62. The u direction is along the X axis; v is along the Y axis. The rgb value represents the (x,y,z) position at that (u,v,1) coordinate. Top shows the wireframe, with evenly sampled (uv); middle shows the barycentric interpolation of (xyz) values; bottom shows the original surface shaded similarly.

Once loaded as a texture pair, the U2X map for a shape can be accessed in both the vertex and fragment shaders. Its application in the fragment shader for inverting the X2U map as part of atlas color mapping has been discussed. However, it can also be useful in a vertex shader to quickly generate vertices at any (*uvt*) coordinate. For example, Fig. 95 shows corresponding world points for uniformly sampled (*uvt*) in each of several target regions. Such sampling could have a variety of potential applications for understanding data according to model-centric coordinates. The local spoke direction can be also easily interpolated, and if one assumes that the surface is orthogonal to the spoke direction (*i.e.*, the surface is "partial Blum"), then the spoke

direction can be used as a proxy for the normal direction. The image derivatives define *du* and *dv* directions for a local tangent plane.

Computing a U2X Map

U2X maps can be created by rendering the surface vertices and faces at their (*uvt*) positions, colored by their (unit cube) (*xyz*) positions. In mgrView this is done in a vertex shader that simply exchanges model- with world-coordinates. Fig. 94 shows an example surface and its U2X map. The small errors in connectivity at the top of the image are the *corners* of the implied surface. Each boundary surface generated by Pablo has four medial-sheet-array-corner crest regions. When wrapping around the object in *v*, the first two corner crests assign coordinates to some of the vertices that are counted again in the later corner crest regions. Thus, a few values of "shrinkwrap" (*uv*) are missing from Pablo-generated surfaces. This has no effect on the rendering since those (*uv*) coordinates are unused.

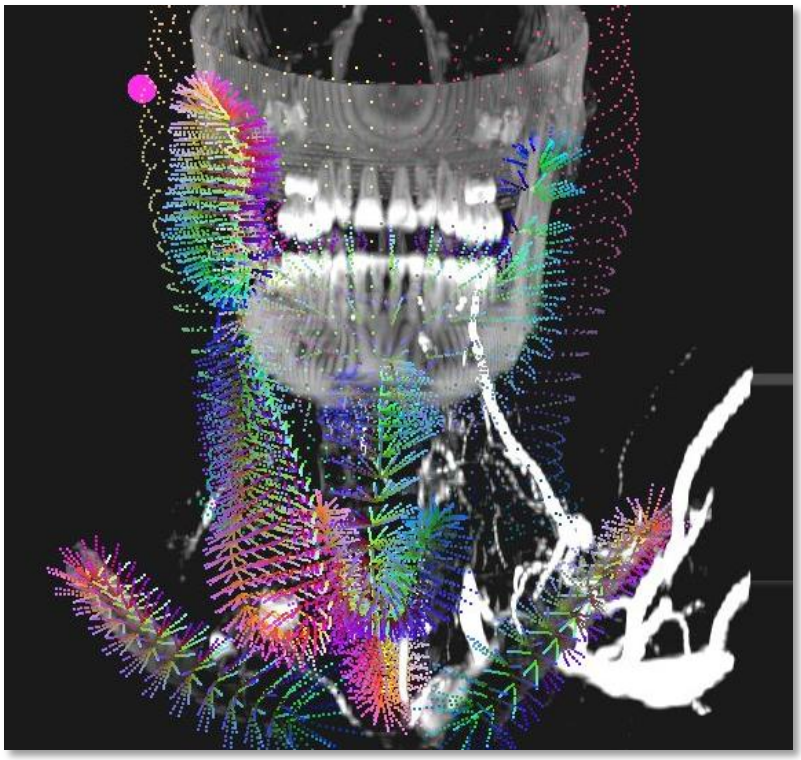

Fig. 95 Passing a uniform sample grid in parameter space through the U2X maps produces a regular sampling of each region in world-space. Here each point is at the world-space coordinate computed from an input object coordinate.

When generating a fully volumetric U2X map, this function is called at least two times with t=1.0 (surface) and with t=0.0 (medial positions). The image buffers are loaded onto the graphics card as 2D texture units. As with the X2U maps, mgrView additionally saves these images out to cache so that they need not be recomputed.

The U2X maps can be quite small: in principle they need only a single pixel for each vertex in a given direction. For example, a shape with 10 medial samples in u and 4 in v (treated as 8 because that implies 4 more on the bottom of the sheet) with a single level of subdivision would require only approximately a 20 x 16 pixel image to capture all the vertices. In practice, because there are additional vertices along the crests that must be represented and because there is no waste in using a power of 2 sized image, image dimensions are square and computed as NextPow2(max(u*1.5,2*v*1.5)). This is still quite small: the U2X example described above is still only two 128 x 128 pixel images. As with all small texture maps, these small images have good cache coherence when manipulating them in a shader program.

One disadvantage of this representation is that its spatial precision is limited by color precision to 256 bins across the unit cube in each direction. Considering that the (xyz) positions of the original vertices may have sub-voxel accuracy and that there are usually 512 voxel units across clinical CT images, this might be a serious weakness in certain contexts. This could be overcome by identifying a region of interest as an additional offset and scaling for the U2X map. This would give 256 bins of precision across the extents of the object, but that has so far been unnecessary for color mapping from the Visible Human.

U2X Maps for Multiple-Object Scene Catalogs

As with X2U maps, it is preferable to collapse multiple U2X fields into the same texture unit; however, in this case the procedure is somewhat more convoluted. The simplest solution is to concatenate each object's grid along the u-axis of a master U2X field. Thus, a 32 x 32 u2x map becomes a 32 x 64 grid for a two-object scene or a 32 x 128 grid for a four object-scene. Then the u value is transformed by adding the object-number divided by the total number of objects. Care must be taken at the edges to not interpolate across objects. Hardware texture requirements mean that the total number of objects representable in the consolidated map is restricted to be a power of two, but there is no particular harm in leaving blank entries at the end.

3.4.3 Using the Visible Human as a Color Atlas

Mapping color from the Visible Human cyrosections allows regions in the target image to be shaded according to one fairly realistic anatomic atlas. Several other methods for high quality rendering using the Visible Human data have been proposed, most notably VoxelMan, shown in Fig. 97. As mentioned previously, VoxelMan rendering is done offline, and it requires exhaustive microsegmentation, so it is not suitable for target clinical patients.

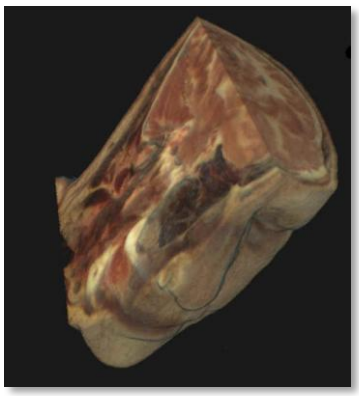

Fig. 96 Direct rendering of the Visible Female color atlas with mgrView.

However, it would certainly be possible to use the color mapping technique described earlier to pull the detailed voxel-wise labeling from certain regions in the VoxelMan data set and map them into a clinical patient, for example, bringing small blood vessels or nerves which are unseen on a target CT but have been identified in the Visible Human back into the target patient scene. This would not require a great deal of work, since many structures in the Visible Human are already targeted for m-rep based segmentation to serve as regional color atlases. Given an additional voxel-wise field from a project such as VoxelMan, *e.g.*, 'blood-vessels', this same segmentation framework could easily be used to pull those additional channel into the target rendering along with the color values.

Fig. 97 Voxel-man renderings from the Visible Human from www.voxel-man.de.

There are other less segmentation-driven methods for augmented rendering such as (Dong and Clapworthy 2005), which proposes a method for enhancing isosurface-type volume rendering of the Visible Human data by looking at small features with high curvature to identify fiber orientations (Fig. 98). In both cases, the rendering target is limited to the Visible Human, in the first case, because it presupposes an exhaustive segmentation, and in the second case, because such detail-

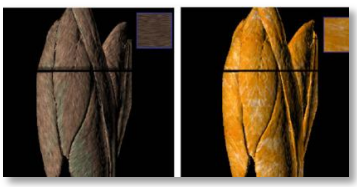

Fig. 98 Volume rendering with two different styles of oriented texture from (Dong and Clapworthy 2005).

oriented methods require extremely high resolution images with no motion artifacts.[20]

There are many educational applications specifically designed to visualize and interrogate data from the Visible Human (see www.nlm.nih.gov/research/visible for a list), but the only approach that, like MGR, allows color information from the Visible Human to be mapped into a new target patient rendering is found in (Lu and Ebert 2005) and associated work from the same group. (Lu and Ebert 2005) proposes using example patches from the Visible Human to synthesize 3D tiling "Wang" texture cubes that can be mapped into broadly defined regions in a new target image (Fig. 99). This is a very flexible approach, but it lacks MGR's ability to use model-centric coordinates to map, for example, the marrow of a bone into the interior regions and the surface of the bone to the boundary.

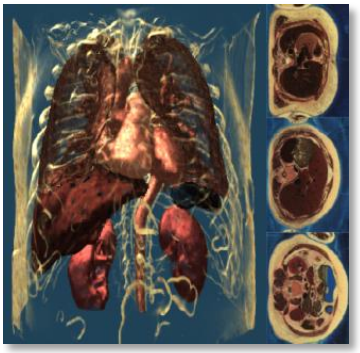

Fig. 99 High quality rendering using textures synthesized from the Visible Human sample colors shown on the right, from (Lu and Ebert 2005).

[20] "We can't get resolution like that in the clinic. For one thing, we're pretty sure that the Visible Human subjects weren't moving when they were scanned." – *Gregg Tracton*

4 Model Guided Composition for Medical Images

All volume rendering suffers from problems of occlusion and obscuration; unimportant regions in the scene get in the way of understanding important features. This is what Levoy was referring to when he made the proposition that no features would be lost in volume rendering as long as the data increased monotonically along the ray. While this statement is true, it is irrelevant because it is *never* the case that anatomic images have this property.

Fig. 100 Left, Vesalius (Vesalius 1973) removed the skin entirely, right, similar view from (Hagens 2007)[21] where the skin has been moved out the way but continues to provide context (*i.e.*, there is a lot of it).

Composing a scene is organizing the rendering so that everything that is most important for understanding a particular problem is revealed. There are three basic approaches for doing this: picking an unoccluded viewpoint and transfer function, deforming the data to move occluders out of the way, or removing occluders entirely (Fig. 100). The first approach is typically used exclusively and is usually left entirely to the viewer although (Bruckner, Kohlmann, et al. 2008) and others have proposed semi-automatic approaches to scene composition via parameterized views. This chapter covers MGR's approaches for enabling the latter two of these approaches.

MGR's **volumetric animation** capability was originally intended to enable interactively pulling aside structures to expose features behind them, but the same methods can be applied to animate the scene with respect to any deformation imposed on the image. The method has received the most attention for its ability to interactively show the

[21] I could write an entire section on von Hagens, but suffice it to say that he is doing "Netterly Rendering" with bodies rather than images: rendering bodies to look more like anatomy than they did when they were alive. And like the Renaissance artists that he fashions himself after, von Hagens has been accused of such unsavory "grave-robbing" as trafficking in the corpses of executed Chinese dissidents, so certain of his exhibits were refused entry into the US.

effects of the local registration fields commonly used in adaptive radiotherapy (ART) or 4D imaging studies. MGR's method for volumetric animation is a straightforward and novel extension of volume visualization to dynamic scenes. Image deformation is bound by the large number of trilinear interpolations required. Having already made the decision to work within the confines of the graphics accelerator hardware, there are a large number of interpolation units available to use for free, and it is simply a question of representing the various images and deformations so that they can interact as texture units.

Clipping approaches in volume rendering usually involve creating complex transfer functions that pick out objects or regions of interest; however, these transfer functions become increasingly fragile and over-tuned to particular data sets. Importance rendering (Viola, Kanitsar and Groller 2004) is a significant recent technological insight that, like MGR, relies on a model for the scene to make smart "importance decisions" for each pixel. The major drawback of current work in this area is that it is implemented with straightforward but slow ray casting algorithms that require pre-determining the scene's importance rules. MGR proposes a novel object-order **fast importance rendering** algorithm that works on ranked surfaces rather than according to voxel values. This algorithm is related to the concept of "shadow volumes" and is fast enough that it enables dynamic importance rendering, by which a scene can be dynamically clipped against an arbitrary shape with interactively controlled position and size.

These two methods for scene composition can be used for a variety of effects. In particular, when taken together they are very useful for showing changes in shape and spatial relations in time, which is the basis for the project *MGR Applications in Adaptive Radiotherapy*, which will be discussed in section 5.2 of the next chapter.

The chapter closes with a very short section on **clipping surfaces with model coordinates**. This technique is straightforward in MGR and can be used to produce some attractive anatomic layering effects.

4.1 Volumetric Animation

Clipping is the most widely use approach for scene composition in volume rendering, however, there are certainly scenes for which no static view completely exposes the important features or for which it is necessary to preserve *all* of the features for context. In particular, scenes which change over time such as an image under the effect of a deformation fall into this category.

As discussed previously, one of the basic problems with doing this kind of operation is slowness of interpolation. For any given time step, several million trilinear interpolations need to be done if every voxel is potentially contributing to the scene. The key insight is simply that there are a large number of parallel trilinear interpolation circuits available on modern graphics accelerators that can be leveraged to do this work if the problem can be cast into an object-space algorithm.

The section includes two topics, one giving an example of **rendering images under a global deformation**, and another describing the general technique for and implications of **rendering images under a local deformation field**.

Fig. 101 Detail from da Vinci's "Babe in the Womb" c.1511, which, along with modern work by von Hagens, was cited as particular inspiration for the methods developed in (S. Bruckner 2006).

4.1.1 Rendering Images Under Global Deformation

(S. Bruckner 2006) describes a method for gpu accelerated volume ray-casting of "exploded" views, shown in Fig. 102 left. This technique is effectively a global deformation applied selectively to the volume image. A similar deformation technique was built in mgrView as an example of how easily such global deformations can be added in this framework (Fig. 102, right).

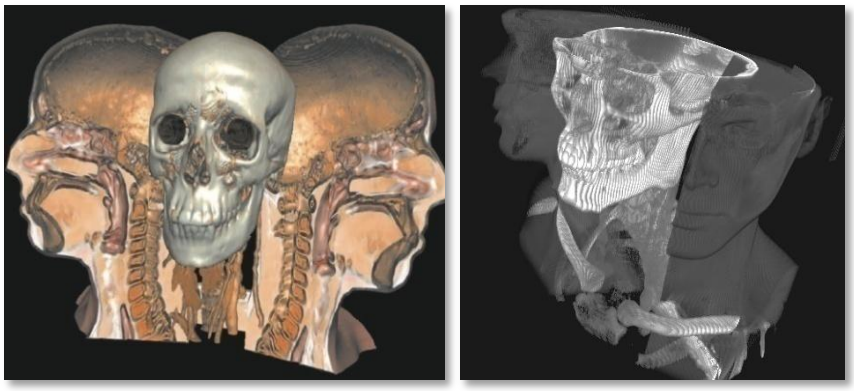

Fig. 102 Exploded view from (S. Bruckner 2006) and similarly deformed view rendered in mgrView.

Consider *z* fixed and the center of the split is at c=(0.5,1.0). Then a 2D position *X* is going to a new position *X'* as a function of θ according to Eqn. 9. The original position of a fragment that is at *X'* can be found by inverting the equation. Finally, recognize that any source position with an x value that is on the wrong side of the center line with respect to the fragment position (*i.e.*, sign(x'-c_x) ≠ sign(x-c_x)) or any fragment that maps to the preserved positions should be rejected. The fragment shader for this is shown in Program 14 and the result in Fig. 102, right. Note that in this sample case, the position of bone is preserved (image values greater than 0.3), but given a scene catalog the clam shell effect could preserve any collection of regions.

$$\theta' = \begin{cases} -\theta, & x < c_x \\ \theta, & x \geq c_x \end{cases}$$

$$X' = (X - c)\mathbf{R}(\theta') + c$$

$$X = (X' - c)\mathbf{R}^{\mathrm{T}}(\theta') + c$$

Eqn. 9 Formula for transforming a world point X by the clam shell operation to find X', and the inverse transform to recover the original position of a transformed X'. **R** is a standard 2D rotation matrix.

```
uniform float time;
vec2 center = vec2(0.5,1.0);
vec2 split_vol( vec2 X ){
   float theta = 0.81 * time * (X.x>center.x?-1.:1.);  // Want to go PI/2 at 1
   mat2 Rt = mat2( cos(theta), -sin(theta), sin(theta), cos(theta) );
   return (X-center)*Rt+center;}
void main(void){
   source_color = texture3D(source_im0,  vec3( gl_TexCoord[0] ) ).rgb;
   // Fragment was soft tissue
   if (source_color.r < 0.3) {
      vec2 X_new = split_vol( vec2( gl_TexCoord[0] ) );
     // Determine if the fragment came from the same side of the split
     // as it started on, this would be more complex if not axis ailgned
      if (sign(X_new.x-center.x)!=sign(gl_TexCoord[0].x-center.x))
          source_color = vec3( 0.0 );
      else {
         source_color = texture3D(source_im0,  vec3( X_new,
                                          gl_TexCoord[0].z ) ).rgb;
        // Fragment was bone, so ignore it
         if ( source_color.r > 0.3 ) source_color = vec3( 0.0 );
     } }
  if ( source_color.r < 0.05 ) gl_FragColor = vec4( source_color, 0.0 );
  else {
     float windowed_value = intensity_window( source_color.r );
    gl_FragColor = vec4( windowed_value );
   }}
```

Program 14 GLSL for the volume splitting algorithm.

The rendering is fairly effective as an animation, but it is expensive to compute. This implementation runs at a respectable 15 fps on a target laptop with an NVIDIA Quadro NVS 160M gpu, but at sub-interactive rates (4 fps) on a target desktop with a NVIDIA GeForce 6200 gpu (see Table 4 on page 142). There may be more efficient means of computing this particular transform, such as rotating world-space half-cube texture

stacks, but this example is merely meant as an illustration of the general idea of applying global deformations in this framework.

4.1.2 Rendering Images Under Local Deformation

Deforming the volume to move particular regions in order to visualize structures that are hidden behind them (Fig. 103 and Fig. 104) requires some kind of physical model. The parametric deformation model described in (Correa, Silver and Chen 2006) produces local appearance effects, but is actually much closer to the global deformations discussed in the last chapter. While m-reps are amenable to local physical modeling such as finite element modeling (see (Crouch, et al. 2003), for example), implementing such a framework is beyond the current scope of this project.

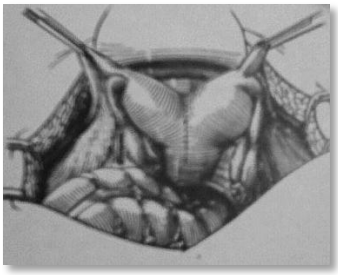

Fig. 103 Image from (Hagen 1992). Retractors are used to reveal hidden internal anatomy.

However, the 3D image registration fields discussed earlier provide an interesting related problem that relies on the same basic method for representing local deformation. Furthermore, such registrations are an important aspect of image analysis but have only limited support among visualization tools.

The method requires at least two patient images, I_0 and I_1, that are related by a rigid transform, **R**, and a dense vector field, **H**, that accounts for the residual change such that $I_1 = \mathbf{H}(\mathbf{R}(I_0))$. The particular registration method is unimportant here. In this example taken from inter-fractional patient images, the registration is done with ImMap, described in (Foskey, et al. 2005).

Fig. 104 Image from (Correa, Silver and Chen 2006) that uses parametric manipulators such as peelers and retractors to visualize a deformed space.

H is stored as a 3D color texture unit, where (rgb) represents the displacement at each voxel (hx,hy,hz). As each fragment is rendered, its position X is used to lookup the local displacement **H**(*X*), and a final intensity is determined by looking up the image value at *X*+**H**(*X*). Trilinear interpolation of the gray values in the deformed space is done automatically in hardware as long as OpenGL is instructed to use GL_LINEAR as the texture interpolation method. With a scalar *time* ranging between [0,1], the volume can be incrementally deformed by indirectly mapping assigning intensity(*X*) to intensity(*X*+time****H**(*X*)).

Program 15 shows the fragment shader for this, and Fig. 105 shows two frames from the resulting interactive animation that shows the entire range of deformation between the source and target images. This idea

is expanded and detailed in the later section describing an application of mgrView for evaluating inter-fractional shape change.

```
// GLSL to SpatiallyTransform a Volume Texture According
// to a Non-linear Deformation Field
uniform sampler3D gray, registration;
uniform float time;
main(void) {
  coord = gl_TexCoord[0].xyz
  coord = coord + time * texture3d( registration, coord )
  source_color = texture3d( gray, coord )
  gl_FragColor = source_color
}
```

Program 15 GLSL fragment shader extending the volume texture shader with volumetric animation.

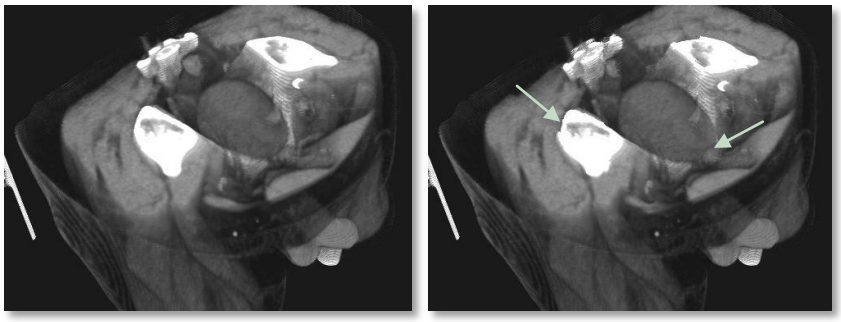

Fig. 105 Two frames from an animation showing the registration between two daily images in a fractionated male pelvis treatment. The change is subtle, only a few voxels in most places, but notice the jog in the hip-bone where the region of interest passed through it and the position of the lower tip of the bladder.

The technique extends to non-linear displacement fields that are represented as chains of displacements fields, such as the curved diffeomorphic paths generated by (Joshi and Miller 2000), by loading each time step as a discrete texture unit and stepping through each in turn, *i.e.*, intensity(X) = intensity(X+time*$\mathbf{H}_n$(time*$\mathbf{H}_{n-1}$...time*$\mathbf{H}_0$(X)))). The problem with doing such chaining is not that there are a large number of interpolations to do but that there are a large number of ordered texture lookups to do for each fragment. Hence, this technique is *considerably* slower than single-step displacements. And while it may be useful for computing general diffeomorphic transformations, in the context of same-patient serial imaging, where the displacements are typically only a few voxels and the difference between the curved and linearized paths is smaller than a voxel in magnitude, the added precision does not provide much visual advantage.

The main drawbacks for this method arise from the requirement that images be loaded onto the graphics accelerator as standard texture units. The next two paragraphs discuss some particular implications of

this, **reduced precision** and **restrictions on relative scaling between images**.

As with gradient data, discussed earlier, each displacement, which typically has sub-voxel precision natively, must be encoded as unsigned bytes (modulo a scaling constant) and then interpolated in low precision. Again, while may not be appropriate for scientific calculation, in the context of these registrations with relatively small displacements, this is not a significant issue. For example, a range of 16 voxels (+/- 8 voxels) still provides 16 units of sub-voxel precision (increments of 0.0625 voxels). This is more than enough precision to understand even relatively small shape changes in objects such as the prostate or bladder that are several tens of voxels in size. If more precision is necessary in some other context, the problem could also be addressed by using more sophisticated graphics hardware that supports half-float data interpolation (a 16 bit float format supported for higher precision computations on some newer graphics architectures).

There is also a difficulty in reconciling texture units with different extents. In particular, most registration fields are computed in a restricted region of interest relative to the original images. Because texture units are all maintained in a cardinal unitary space that also includes any additional voxel padding required to meet most hardware's requirements that textures be sampled with dimensions in powers of two, additional care must be taken to convert cm distances into unit distances that maintain relationships between data sets. The one ameliorating factor is that each texture may have its own independent 4 x 4 transformation matrix for converting world-distances into unit texture extents distances. In ray-casting, by contrast, units can be maintained in their native format and cm distances along a ray can be converted explicitly to voxel-values by straightforward formulae.

Finally, this method also applies to surfaces in a scene. The position of each surface vertex can also be used to index into the displacement field and determine a new position. For surfaces this must be done in the vertex shader, which requires hardware that supports OpenGL's "vertex texture fetch" (VTF). As described in the next section of this chapter, this technique provides a source for dynamic importance clipping in animating scenes – the changing surface provides a new clip frustum at every frame.

Volume Morphing

Most spatial registration algorithms cannot account for substantial intensity differences between the images. Consider a rectum with a gas bubble registered to one without gas. The patient may be the same, the nearby anatomy may be the same, and the shape of the rectum may be the same, but there is an intensity difference that requires either that the registration be "torn" so that the intensities can be inserted, or typically, the gas be "deflated" away to a tiny point so that the registration can remain diffeomorphic.

In practice, with the clinical algorithms that we use, this happens all the time at a small scale. The smoothness constraint either forces the registration to "miss" or allows the registration to produce an unsmooth and likely illegal mapping (preview Fig. 147). A pixel with no good corresponding intensity will obligingly map itself to a good spot suggested by its neighbors. This means that $H(I_0)$ is almost never actually equal to I_1. It is I_0 as close as it can get to I_1 within the smoothness constraint. This has no practical consequence in the clinic, but it can be somewhat disconcerting when visually comparing $H(I_0)$ with the intended target image I_1.

Actually "morphing" between the two 3D images requires both a registration field and an intensity difference term to absorb the residual error in the registration. This problem has been addressed in a formal way for medical image analysis by the "Metamorphosis" method presented in (Trouvé and Younes 2005).

An overly simplified solution is to linearly interpolate between the pixel intensity in I_0 that is t along the displacement vector and the pixel intensity in I_1 that is (1-t) along the displacement vector, as expressed in Eqn. 10. This formulation is easily implemented in mgrView by adding a few lines to the animation shader to sample I_1 and mix the values from the different images.

$$I_t(x) = \frac{tI_0\big(x + tH(x)\big) + (1-t)I_1\big(x + (1-t)H(x)\big)}{2}$$

Eqn. 10 Formula for linearly interpolating intensities at time t from source (I_0) and target (I_1) pixels as they approach each other according to a registration field H.

4.2 Fast Importance Rendering

Scene composition is, at its simplest, about rendering in such a way that important structures are not obscured by unimportant structures. As discussed earlier, most volume rendering approaches suppress or enhance features identified according to local image properties by complex transfer functions. For instance, Levoy's original proposition that gradient magnitude should determine opacity was an implicit assumption that region boundaries were the most important features for understanding a volumetric scene. Other more advanced clipping rules tend to follow this cue and make further assumptions, looking to second derivatives or other feature matching methods to separate important from unimportant features.

Unfortunately, as with assigning scene appearance, there are no good purely data driven transfer functions for automatically determining the relative importance of regions of any general scene. Such labeling is essentially an image segmentation task, so scene composition requires both external information and a robust method for identifying important features in the scene.

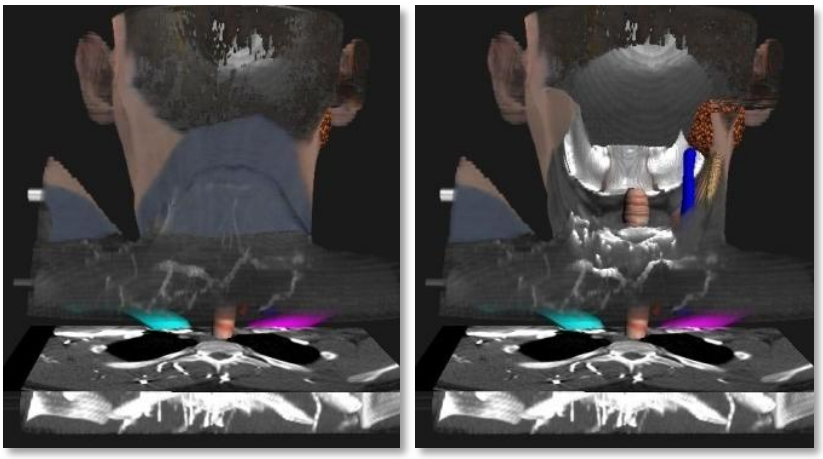

Fig. 106 Left, another perspective of the scene from Fig. 4. Right, the same view with voxels in the mandible's importance shadow culled away.

MGR's clipping algorithm ultimately relies on the same partial segmentations used to guide appearance as guides for scene composition. Fig. 106 shows an example. The image on the left is the same scene shown previously shown in Fig. 4, but rendered from behind. From this perspective, most of the important internal structures are hidden inside the volume. The image on the right shows the same rendering with the volume culled wherever it would obscure a structure of interest, in this case, the mandible.

This basic idea of scene composition based on pre-segmented features is known as "importance rendering" and was originally proposed as an image-order method by (Viola, Kanitsar and Groller 2004) (Fig. 107). Given a partial segmentation of the scene into ranked regions, the problem of scene clipping in MGR reduces to finding an object-order algorithm for importance rendering. The importance clipping algorithm used in mgrView is based on an extension of the standard stencil-buffer based shadow-volume algorithms sometimes used to compute surface-to-surface occlusions. In the example shown in Fig. 106, the mandible surface is used to compute a shadow frustum extruding from the mandible towards the viewer; then each textured plane is stenciled against the shadow geometry as it is composited to clip away the intervening voxels.

Fig. 107 The lizard from (Viola, Kanitsar and Groller 2004) with an importance hierarchy emphasizing the bones and liver.

This section is divided into three topics: first a review of antecedent work in **importance rendering**, second the details of the method for **stenciling with importance shadows**, and finally a discussion of extending **importance stenciling for hierarchically ranked or dynamic objects**.

4.2.1 Importance Rendering

There are two distinct methods for identifying important regions and relationships between them in a scene. Manual identification of important regions is usually based on simple geometric proxies (*e.g.*, solid rectangles or spheres) and called volume-of-interest or **region-of-interest (ROI) clipping**. Alternatively, given an image segmentation, the rendering engine can automatically determine more complex shapes of interest and relationships between them. The term *importance rendering* is usually reserved for such **model-based clipping**. MGR's importance clipping algorithm can be driven by either manually

controlled ROIs or by the underlying scene segmentation, but this section reviews methods of model-based clipping in particular. Previous examples of volumetric clipping with simple manually identified ROIs are reviewed briefly in the later section describing mgrView's "importance flashlight" dynamic clipping tool.

Important structures in a patient image are commonly segmented as part of our clinical pipeline. Scene composition according to such segmentations has been a recent and productive area of research inspired by techniques for cut-away views and ghosting from technical illustration. The term "importance rendering" was coined in (Viola, Kanitsar and Groller 2004), which is a significant and beautiful work (Fig. 107). Viola uses a hierarchy of geometric models with importance values to automatically infer the regions of interest and generate view-dependent effects in an image-order framework.

"Flexible Occlusion Rendering" proposed in (Borland, et al. 2006) (Fig. 108) falls into the same category. This algorithm works by using a transfer function to identify contrasted regions and then resetting its accumulation function when it encounters anatomic intensities again.

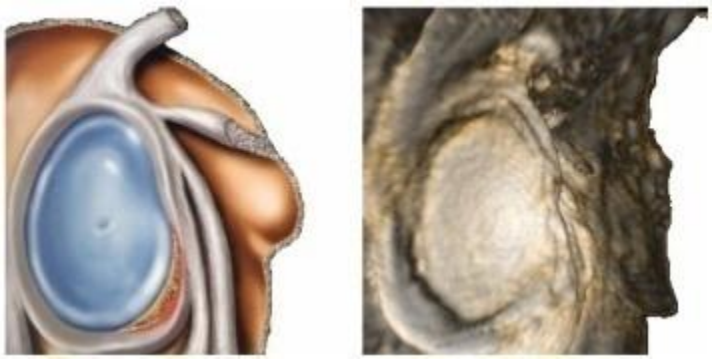

Fig. 108 Left, an anatomic illustration of a shoulder joint and right, a similar view of real data rendered with flexible occlusion from (Borland, et al. 2006).

The main disadvantage of both Viola's and Borland's methods is that they are designed to be implemented as transfer functions and so rely on ray casting, which makes them relatively slow to compute. Computational complexity can be surmounted by fast hardware – indeed, (Quammen 2006) describes a graphics hardware accelerated version of Borland's ray casting method that is quite fast, given the proper hardware. However, Borland's method requires a contrasted image, and Viola's method requires a voxel-wise segmentation with per-voxel importance values. Thus, neither of the methods is amenable to *dynamic* importance descriptions. Enabling dynamic importance clipping – *i.e.*, importance clipping against an interactively controlled

object, is the main advantage that moving to an object-order implementation gives MGR.

Other antecedent work on volume clipping using explicit models includes (D. Chen 1998) and (Bullitt and Aylward 2002). In both cases the models are medial. (Bullitt and Aylward 2002) uses a voxelized label image to guide clipping when rendering the liver and vasculature for surgical planning. (D. Chen 1998) is a direct precursor to MGR in using medial models to guide volume visualizations. However, the methods are primarily concerned with doing near-model clipping and lighting the volume according to local data-driven boundary displacement on medial objects (*i.e.*, using the model normals rather than the gradient direction). An example rendering is shown in Fig. 109. It is unclear why this method required *medial* models since the space-filling volumetric coordinates are not used, but it is an interesting first step towards being able to identify and focus visualization on objects of interest by tile geometry rather than according to simple ROIs.

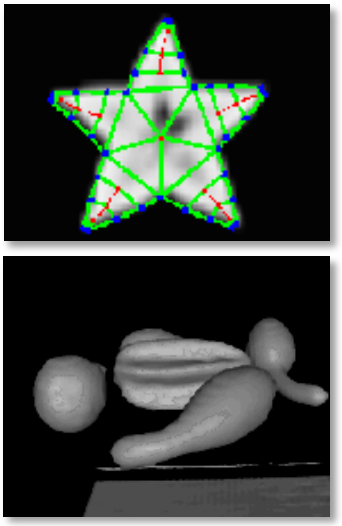

Fig. 109 Images from (D. Chen 1998). Top, a medial model fit to a scanned starfruit. Bottom, medial models fit to the objects in the scene are used for clipping and to smoothly shade the rendering.

Finally, some additional related work on clipping and scene composition for *surfaces* is reviewed in the final section of this chapter.

4.2.2 Stenciling with Importance Shadows

This section presents an object-order implementation of importance rendering suitable for a hardware accelerated object-order rendering framework. In a ray-casting context, determining whether objects are occluded by or are occluders of anther object is a straightforward task – when a surface is intersected by a ray, an additional "shadow ray" is sent to each light source. If the shadow ray intersects with any surface before it reaches the light source, the generating surface point is considered to be in shadow. Keeping track of “importance occlusions” along a single ray, as in Viola or Borland, is even more straightforward and can be accomplished with a transfer function that simply tracks the most important object already seen and resets the pixel intensity if a more important voxel is encountered.

The main problem with implementing such functionality in an object-order framework is that parallelization requires that each fragment is necessarily treated independently and that the computations for each fragment can access only a small amount of local data. Taken together,

this limits the fragment shader's ability to compute complex "transfer functions" for both color and opacity. The key insight for this algorithm is that occlusions in volumetric data are essentially related to cast shadows – and there are known algorithms for dynamically computing the effects of cast shadows using per-fragment operations. MGR's importance rendering mechanism is based on such an object-order cast shadow algorithm that relies on a stencil buffer to track the simple dependency of whether a fragment is shadowed or not across all the fragments that contribute to each pixel. However, instead of determining *surface fragments* that are occluded with respect to a *light source*, MGR's importance shadows determine *volume fragments* that are occluders with respect to *an object of importance* and then stencils them away.

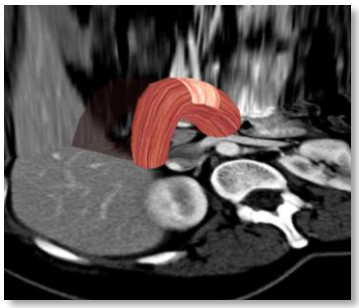

Fig. 110 Cast shadows provide useful visual cues when combining surface and volumes data.

Shadow Volumes & Importance Shadows

The cast shadow method used by MGR is based on "shadow volumes" or "stencil shadows", originally proposed in (Crow 1977) and extended by (Heidmann 1991) and others. In this method, a shadow volume is a derived surface for each potential occluder and light source such that the surface polygons that are back-facing with respect to a light source are projected to infinity. The shadow volume then encloses all the surface elements that are occluded with respect to a particular surface and light source. Fig. 111 shows an example of a 2D shadow area from (nVidia 2004)

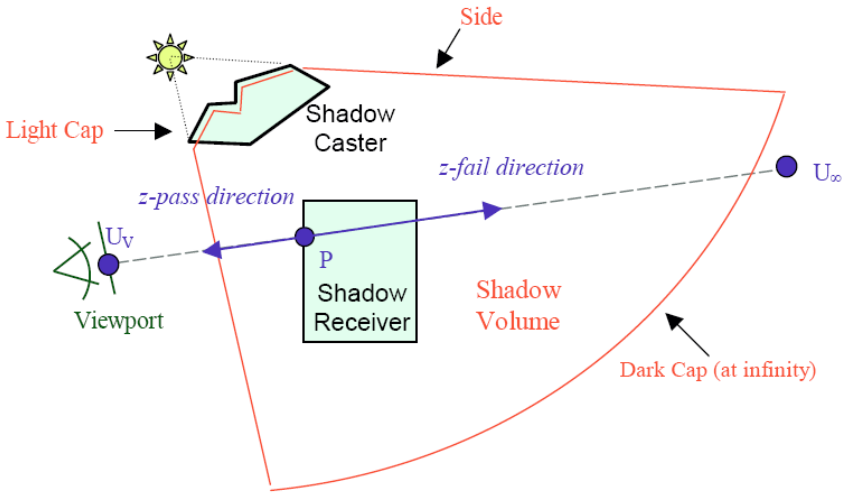

Fig. 111 Shadow volume geometry in 2D from (nVidia 2004)

The scene is rendered as normal. Then the pixels that are inside the shadow volume can be identified by rendering the shadow volume in a particular way that allows the hardware to count how many fragments

project onto each pixel. This is the object-order equivalent of casting rays through the shadow volume and counting the number of intersections before the ray finally reaches the target surface. If the count is odd and the shadow volume has no self-intersections, the pixel is shadowed.

The cleverness in implementing this as an object-order algorithm comes from using the stencil buffer to track the number of fragments from the shadow volume that project onto each pixel. The stencil test is a standard part of the graphics pipeline that passes or rejects a fragment and updates the stencil buffer based on some condition. The stencil buffer can be thought of as an additional channel for the frame buffer. Its original intent was to store a mask that could be used to restrict 3D rendering to part of a window, for example, to reserve a 2D region for a user interface.

In implementation, the shadow volume is rendered without writing to the screen buffer or the depth buffer although depth testing is still used to determine if a fragment is in front of the target pixel. Instead of simply counting every fragment and then using even/odd to determine if the pixel is shadowed, the stencil can be set to increment on front facing fragments and decrement on back facing fragments. Thus, any non-zero value implies that the pixel is inside of the shadow volume regardless of self-intersections.

The rasterization pipeline can then treat the shadowed pixels differently. Typically for cast shadows, the stencil buffer is used to mask a second pass rendering that either rerenders the entire scene without the occluded light or, as a short cut, simply blends a dark color over the fully lit pixels to simulate shadowing (Fig. 112, bottom right).

Program 16 gives a vertex shader implementation for generating a shadow volume as a function of a source position and frustum scaling and Fig. 112 illustrates the shadowing process.

```
void main(void){
   vec3 light_direction = normalize( gl_Vertex.xyz – gl_Light0.position );
   float diffuse = dot( -light_direction, gl_Normal );
   // If vertex is not lit, extrude it
   if (diffuse < -0.05) world_position = world_position + light_direction;
   gl_Position = gl_ModelViewProjectionMatrix*vec4( world_position, 1.0 );
}
```

Program 16 GLSL code for generating shadow volumes using a vertex shader program.

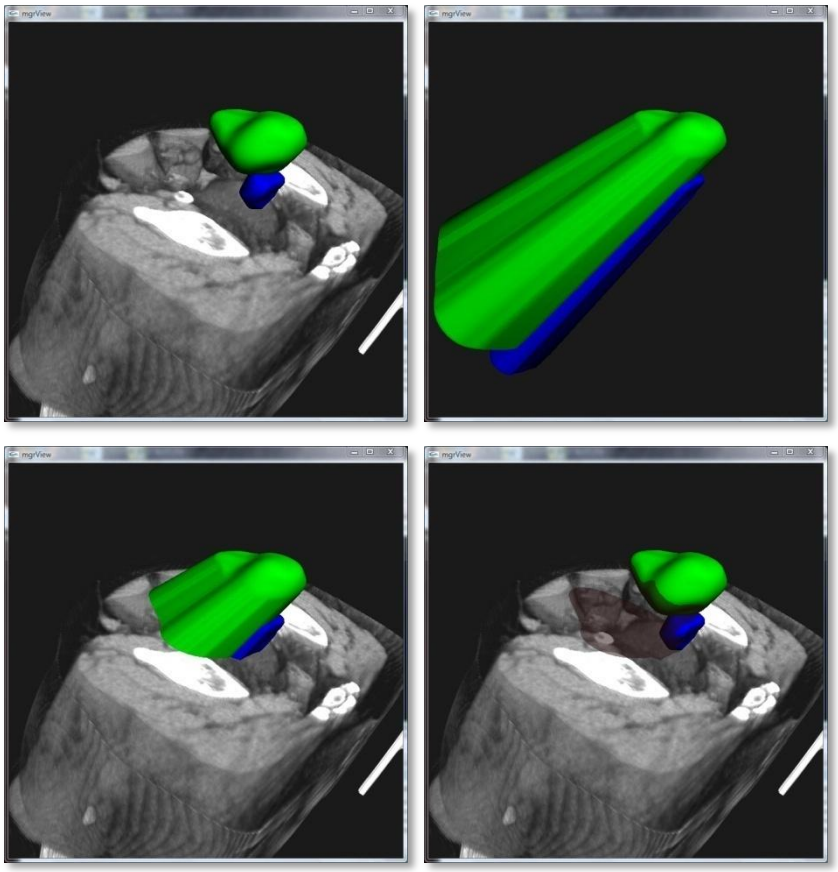

Fig. 112 Shadow volumes rendered in mgrView. Top left, the bladder (green) and prostate (blue). Top right, shadow volumes extruded using a light direction from the upper right. Bottom left, intersecting the shadow volume with a plane in the volume. Bottom right, the dark regions are areas with non-zero stencil buffer entries after the shadow pass.

Implementing this method in mgrView provides the basis of the importance rendering mechanism, and it has the additional advantage of providing cast shadows to improve spatial understanding and verisimilitude of the renderings, as seen in Fig. 110.

Method for Stenciling with Importance Shadows

The insight in MGR's method for importance rendering is recognizing that occlusions relative to important objects can be treated similarly to cast shadows. Important objects in the scene, *e.g.*, the prostate, bladder, and rectum in a male pelvis image, have been semi-automatically segmented with deformable shape models and each model implies a boundary surface. Then, each object's front-facing tiles with respect *to the camera* are projected back towards the camera (the opposite direction of a cast shadow volume). In implementation, this is accomplished by reflecting the camera across the center of mass of the scene regions and using that position as the "importance source" for the shadow algorithm outlined in the previous section. Using the center of mass is an expedient choice in that it can be computed easily and provides reasonable results for multiple regions that are near one

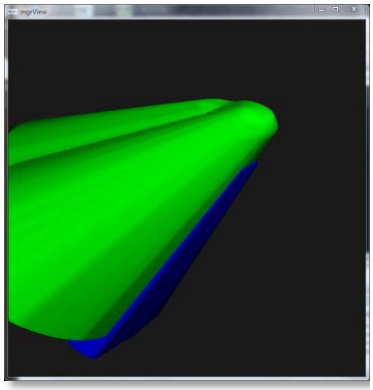

Fig. 113 The shadow volume from Fig. 112 top, right, with the light source "zoomed" towards the center of mass to imply a wider shadow frustum.

another. However, importance shadows from distant regions are better served by calculating individual importance sources for each.

The reflected camera can additionally be "zoomed" towards the center of mass to create a wider shadow frustum (Fig. 113). This effect can be used to open a wider hole through the volume to the object and thereby provide more or less context for the region. Fig. 114 shows a 2D projection of a scene with the importance shadow geometry.

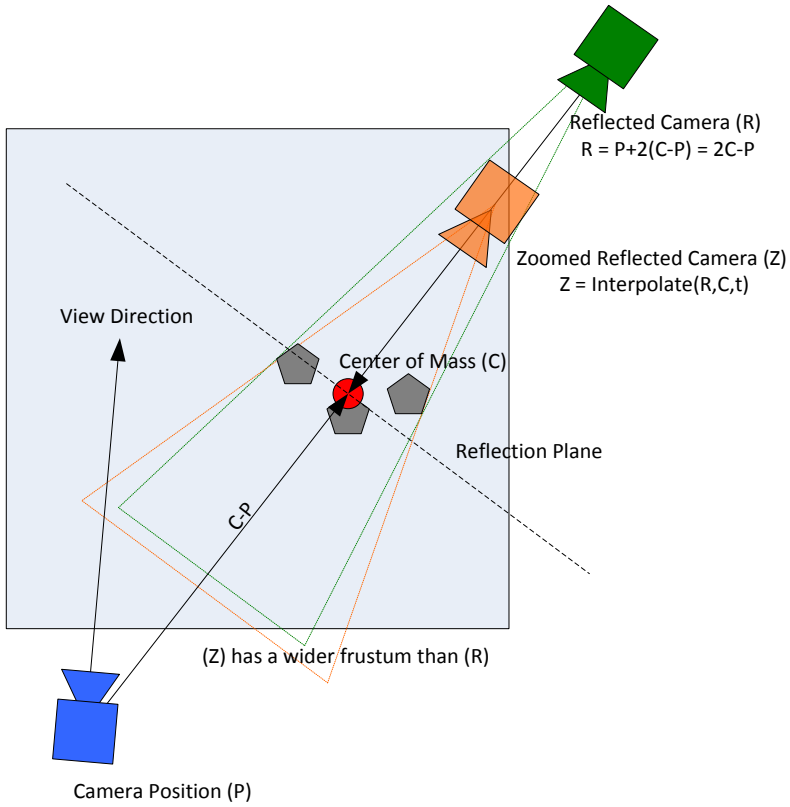

Fig. 114 The shadow volume algorithm is modified by reflecting the camera across the scene, then zooming it slightly to magnify the frustum. The final camera position is then passed as the shadow source to the shadow stenciling algorithm as described above.

After each plane in the texture stack is rendered, the combined importance shadow for all the objects identified as "important" in the scene is rendered into the stencil buffer to create a mask of which fragments on the next plane would obscure the important regions of the data. This mask is then used to cull the fragments of the next plane as it is rasterized, effectively testing that each fragment does not obscure a fragment from an important object. Program 17 adapts the basic object-order rendering algorithm with an importance mask.

The basic shadow volume algorithm can be modified slightly to preserve the interior of the region by flipping the diffuse lighting test from "less than" to "greater than". This causes the closer (lit) cap rather than the farther (unlit) cap to be extruded and so removes the interior of the

object from the "shadowed" region. This is most useful in cases of high contrast objects, such as the mandible (recall Fig. 106) or when applying a (more) opaque synthetic or image texture in the image region, such as is shown in Fig. 115 middle.

Render Volume with Importance
1. Compute importance shadow volumes for this camera
2. Enable depth testing
3. for *i*=0 to number of slices
   3.1 Set stencil to count faces ("increment/decrement")
   3.2 Disable buffer writes
   3.3 Render importance shadow volumes
   3.4 Set stencil to test for non-zero ("cull if on")
   3.5 Enable buffer writes
   3.6 Render slice *i*

Program 17 Object-order importance rendering using the stencil buffer and a shadow volume.

Because this algorithm requires that the shadow volume be re-rendered to the stencil buffer for every slice cast through the volume, it can be a significant speed-up to precompile the shadow volume as a display list. Using a vertex shader to extrude the shadow volume is a significant win here since the display list need not be recompiled each time the camera position changes. Since no fragments are actually promoted to pixels, this is a fairly fast operation. mgrView achieves frame rates over 15 fps for an importance stenciled 200 slice view on a target laptop and around 6 fps for an importance stenciled 100 slice view on a standard desktop workstation. See Table 4 on page 142 for a complete summary.

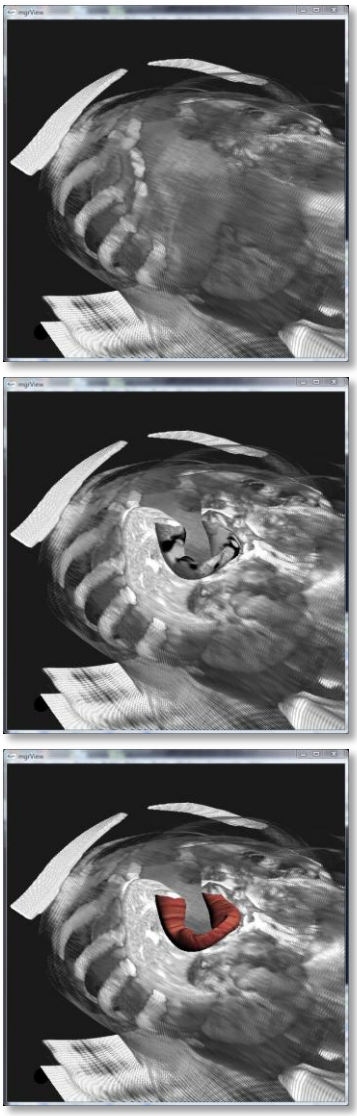

Fig. 115 Images from an abdomen scene focused on the duodenum. Top, the duodenum is completely occluded in this 3D view of the abdomen. Middle, nearly opaque intensities from the image in the duodenum region. Bottom, using a model-mapped texture in the duodenum region.

4.2.3 Extending Object-Order Importance Effects

This section describes some additional object-order importance effects, **importance for dynamic objects**, **hierarchical importance rendering**, and adding **regional effects** to a scene. The section closes with a discussion of **voxel-to-context** importance decisions in this framework.

Dynamic Importance & ROIs

The method described above is fast enough to provide importance rendering for two different types of dynamic scenes: for changing importance surfaces in a static scene or for an animating scene as described in the previous section. An example of a dynamic object embedded in a static scene is mgrView's **importance flashlight** (Fig. 116), which is a simple spherical shape that can be scaled or moved

interactively through the 3D scene to dynamically carve holes in the data. Slight variations of this tool can be used to simulate many different effects and provide many useful and novel ways to interrogate the data without clipping it to a single plane or cut surface. An example of importance rendering for animating scenes is described in the next chapter. Because mgrView's importance clipping method computes importance masks on the fly, mgrView achieves the same frame rates for scenes with dynamic importance as it does for static scenes.

Fig. 116 Volume rendering from an unsegmented image interrogated with a spherical "importance flashlight".

Creating motion in the scene by interactively manipulating the camera viewpoint or the parameters of the carved shape makes the organization of the grayscale image much more understandable than it is in a static scene. Dilating the shadow source along its normal directions or tuning the frustum width by zooming the reflected camera can provide more or less context near the subject. Interactively adjusting the frustum's projection angle by moving the reflected camera orthogonally to the view direction produces slightly oblique cuts that can help clarify the subject's depth and the scene perspective. In Fig. 116, for example, the projection angle is slightly down and left of the camera direction.

The importance flashlight is related to the standard ROI or VOI method of focusing a scene by manually selecting a region of interest. There are many antecedents to ROI clipping and many methods for picking ROI

shapes. Pelizzari[22] was an important early user of volume ROIs for therapy planning to visualize soft-tissue targets through obscuring outer stuctures. (Pelizzari, et al. 1999) (Fig. 117) and other early papers from this group discuss the application of volume visualization for radiotherapy treatment planning and present patient renderings with compelling scene composition.

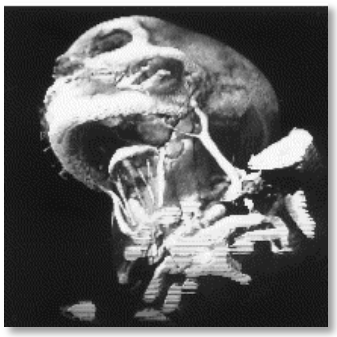

Fig. 117 Image from (Pelizzari, et al. 1999). A left anterior oblique view showing the mandible, hyoid bone, left external jugular vein, anterior jugular veins, left submandibular gland and two associated submandibular lymph nodes.

Other more recent methods for interactively creating clip regions include (Weiskopf, Engel and Ertl 2003) and (Konrad-Verse, Preim and Littmann 2004). Several papers from Weiskopf's group describe an alternative approach for object-order clipping with arbitrarily shaped objects, but their method relies on voxelization of the target shape, so it is not clear that it would be sufficiently fast for MGR's interactive rate rendering goal. Konrad-Verse proposes using a flexible deformable mesh to do 'virtual resection', as shown in Fig. 118. Their goal is not merely to exclude irrelevant shapes but to provide decision support for how particular resection plans will affect hidden internal structures. Their method for determining a clip surface according to cut-lines drawn on planes and 'spheres of influence' could serve as a method for defining complex ROI geometry in any clipping framework.

The "sand away views" in (Davis, et al. 1991) and "volumetric sculpting" in (Wang and Kaufman 1995) describe a grayscale morphology rather than geometrically determined manual clipping. In those works, hidden structures are exposed by eroding the 3D image directly according to grayscale morphological kernels.

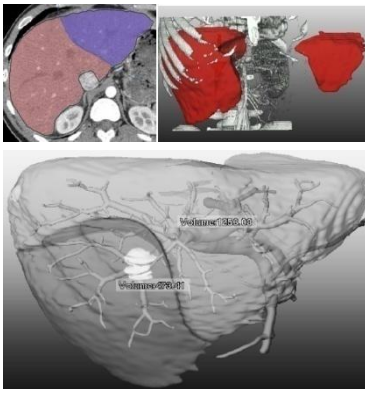

Fig. 118 Virtual resection from (Konrad-Verse, Preim and Littmann 2004). Top left, cut-lines are drawn on each plane. Top right, the object can be separated and 'resected' from its parent. Bottom, the resection clipping surface can be interactively modified so that the disjoint volumes include and exclude different features.

Hierarchical Importance Shadows

This method extends to *hierarchically ranked* importance surfaces. For example, in a given scene the user may wish to see the area near the importance flashlight through intervening important anatomic regions.

There are two possible approaches to address this. One method requires a more complex stencil test that uses the top few bits of the stencil buffer to track the highest object rank at each pixel and the lower bits to count front-back faces as above. This method has the advantage of not requiring the importance manifolds to be presented in any particular order, but implementing it requires using a considerably more complex stencil test.

[22] Pelizzari is probably best known for his 'head-hat' registration algorithm.

A simpler method is to use the single rank method but to order the importance manifolds and stencil their importance shadows in ranked order. This requires only a small amount of additional computational overhead compared to the per-slice shadow passes required for the basic volume importance algorithm.

With either method, because the hierarchical stencil is recomputed completely at each frame, there is no computational penalty for dynamically changing relative importance ranks. mgrView's implementation allows every object in the scene to be assigned a different importance rank, but in practice using more than one or two ranks can lead to very confusing views. Occlusion is a very important and expected depth cue.

Render Volume with Hierarchical Importance
1-3. Render culled volume as in Program 17
4. Clear the stencil buffer
5. For *i*=1 to number of ranked objects
   5.1 Set stencil to test for non-zero ("cull if on")
   5.2 Enable buffer writes
   5.3 Render objects with rank *i*
   5.4 Disable buffer writes
   5.5 Set stencil to count faces ("increment/decrement")
   5.6 Render importance shadow volumes for objects with rank *i*

Program 18 Extending object-order importance rendering to display ranked objects.

Adding Additional Regional Effects

Complex scenes with importance effects can be difficult to interpret spatially since the strong depth cues from occlusions are being reordered in non-intuitive ways. However, because MGR has access to geometric proxies for the regions of interest, regional effects can be added indirectly during a final rendering pass without making the volume data shader itself more complex. mgrView supports three simple additional regional effects during importance rendering, which are **tinting**, **contours**, and **cast shadows**.

In this implementation object tinting is done during texturing with a variation on the standard 3D texture surface shader used in Fig. 115 middle and elsewhere to map voxel values onto surface tiles. As previously discussed in the volume visualization background section of

Chapter 2, contours, *i.e.*, locations where surfaces begin to be self-occluding, (see Fig. 40), are added using a contour shader that performs a dot product between the camera direction and the surface normal and discards fragments that are not nearly orthogonal to the viewing plane. And finally, an actual shadow pass using a shadow volume cast from a regular light source and a shadow volume from the combined region surfaces is used to project cast shadows into the clipped volume and onto region surfaces. Program 19 shows the complete importance rendering algorithm with hierarchy and object effects.

Render Volume with Hierarchical Importance and Object Effects
1-3.Render culled volume as in Program 17
4. Clear the stencil buffer
5. For *i*=1 to number of ranked objects
    5.1 Set stencil to test for non-zero ("cull if on")
    5.2 Enable buffer writes
    5.3 Render objects with rank *i*
        5.3.1 Render tinted object
        5.3.2 Disable depth test and depth write
        5.3.3 Render contours
    5.4 Disable buffer writes
    5.5 Set stencil to count faces ("increment/decrement")
    5.6 Render importance shadow volumes for objects
6. Clear the stencil buffer
7. Stencil shadow volume according to cast light source
8. Render dark quad and blend shadow regions

Program 19 Extending Program 18 with additional regional effects.

Voxel-to-Context Importance & Volumetric Shadows

The scene catalog provides a powerful tool for identifying and categorizing not only voxel-to-neighbor relationships (*e.g.*, local interfaces) but also voxel-to-context relationships. For example, consider a patch of bright voxels. If it belongs to a region that should be bone, it is uninteresting data. However, if it belongs to a region that should be kidney, the patch represents a calcification or lesion and warrants consideration, and indeed warrants focus in the scene. While mgrView implements only shadows from surfaces, cast shadows from particular regions in volumes are also possible in an object-order rendering framework. This raises the possibility of extending the importance rendering method described to clip according to voxel-centric decisions ("Is this voxel interesting compared to its neighbors?") rather than region-centric decisions as described here ("Is this surface more important than its neighbors?").

As with cast shadows for surfaces, cast shadows for volumes are easy to compute in image-order direct volume rendering (DVR) by the simple expedient of casting secondary shadow rays and tracing them on their path through the volume (recall Fig. 44). In the forward rendering context, algorithms have been proposed by (Zhang, Crawfis Vis02) using voxel splatting, and in (Behrens VV98) using texture mapping.

The method for computing and using shadows from volume data originally considered for mgrView is based on a method originally described in (Merck UNC04)(Fig. 119). An occlusion volume is computed similarly to the X2U map computation, by progressively rendering each plane in order into a buffer that accumulates the total attenuation at each voxel from the point of view of the light. The buffer is read back ("unprojected") at each step and stored in a secondary occlusion volume. When the occlusion volume is complete, it can be used to modulate the intensity at each corresponding voxel as it is rendered from the camera's point of view. This method is considerably faster than ray-tracing, but it is still not fast enough to support dynamic lighting.

Fig. 119 Volume with self shadows from an early splat rendering core considered for mgrView.

4.3 Clipping Surfaces with Model Coordinates

The final section of this chapter deals briefly with MGR's capacity for model based clipping or windowing in surfaces. Many anatomic structures of interest, such as the gut or vasculature, are most appropriately described as surfaces, particularly as layered surfaces. The duodenum shown in Fig. 120 is an interesting curved shape with multiple layers. (Revisit Program 1 for the simple mgrView example script used to create it.)

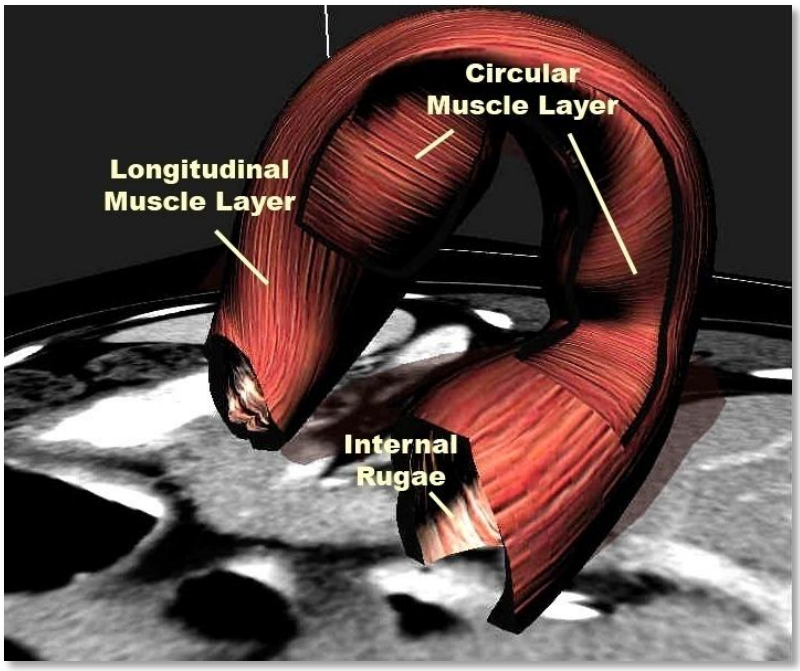

Fig. 120 Duodenum with multiple layers, interior ruggae and circular muscle under longitudinal muscle.

Such overlapping (or interpenetrating) surfaces can be quite difficult to understand. In medical illustration convention, windows in surfaces are frequently aligned with the along-object and across-object directions – which are given in our framework. Little research has been done on mimicking this effect. (Li, et al. 2007) uses constructive solid geometry to identify a medial axis for each object so that windows can be cut transversely or along geodesics on the surface, as shown in Fig. 121. Again, starting from partial segmentations by m-reps, MGR is provided with such directions automatically by construction, making it trivial to assign clipping windows with intuitive coordinates such as between 0.3 and 0.6 of the way along the object and between 0.5 and 0.7 of the way around the object (the actual coordinates of the window in the longitudinal muscle in Fig. 120).

Fig. 121 Composition based on medial properties derived from constructive solid geometry from (Li, et al. 2007).

In implementation, mgrView "clips" the surface to the nearest vertices by simply rejecting any vertices that fall within the model-coordinate clipping window. The clip boundary vertices are marked so that cap geometry can be added dynamically to give the layer the illusion of depth. The end caps on open tubes can be treated similarly. Simple discrete medial models are *de facto* deformations of spherical topologies, so in order to render other topologies, such as hollow tubes for the gut and duodenum, the end caps are clipped with similar object-coordinate windows, for example, clipping vertices with u in the ranges [0,0.1] or [0.9,1] for all v and all t.

A non-model-based method for layer-by-layer clipping is "depth peeling", a method for 2-pass rendering using a depth buffer to selectively cull the front-most polygons of nested surfaces (Nagy, Klein CGA03), (Rezk-Salama, Kolb CGF06). Borland originally called his flexible occlusion rendering "volumetric depth peeling".

Finally, a related field is display of intersecting surfaces, as is explored in (Weigle and Taylor 2005)(Fig. 122) or (Interrante, Fuchs and Pizer 1997). Both these works in particular draw some of their driving problems from external beam radiotherapy planning, the same target domain as MGR does. Both methods propose, among other things, applying oriented textures to surfaces in order to provide shape cues for intersecting objects. In general, these texture orientations are determined by intrinsic properties of the surface, such as lines of principal curvature. However, it seems likely that integrating these texturing methods with model-based surface coordinates as described here might provide additional insights into the relative shapes of the objects.

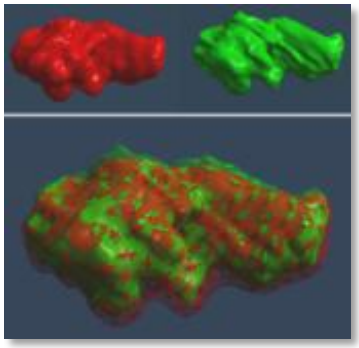

Fig. 122 Two different segmentations of the same tumor rendered relative to one another with the nested surfaces algorithm from (Weigle and Taylor 2005).

5 Bringing MGR to the Clinic

This chapter provides evidence that the MGR techniques described in this dissertation are both **usable** and potentially **useful** in clinical visualization.

The first section, 5.1, *Medical Imaging Applications and MGR*, gives a high level review of the final **application** component in the medical imaging pipeline that was used to organize the background material and is represented in Fig. 123. The section describes the context in which MGR techniques have been developed and discusses some of the ways that they may be applied to image guided medicine.

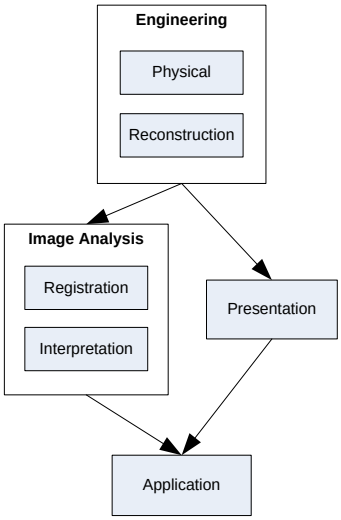

Fig. 123 This chapter contextualizes how MGR fits into the application component of the medical imaging pipeline.

The next two sections provide additional motivation and examples demonstrating how combinations of MGR methods can provide additional useful capabilities for particular important but difficult medical image visualization tasks in radiotherapy and endoscopic guidance. Much of the thinking behind these target problems has been taken from or developed to support grant proposals being written together with Dr. Julian Rosenman and Stephen Pizer.

Section 5.2, *MGR Applications in Adaptive Radiotherapy*, provides a case study describing how to apply some of MGR's methods to improve comprehension in visualizations for segmentation, planning in the presence of error, and patient setup for radiotherapy. This section also refers to an appendix with considerable implementation detail for building the "planning in the presence of error" project using the mgrView software library. The appendix is intended for the reader who may be interested either in reproducing examples from this dissertation or in generating their own projects using mgrView. The appendix covers importing images, dense registration fields, and medial shape models into mgrView, and it also provides a walkthrough of creating a new shader model and integrating it with the standard library functions. The appendix material assumes some familiarity with C++ and OpenGL and can safely be skipped over by non-engineering-oriented readers.

The shorter section 5.3 overviews another target problem, *Enhanced Endoscopy from Multiple Modalities*, and describes how to integrate endoscopic and 3D images to improve online visualizations for biopsy guidance.

The required input for all of these projects is images and data that are commonly collected or computed in the UNC Department of Radiation Oncology's standard clinical pipeline.

5.1 Medical Imaging Applications and MGR

MGR brings several immediate general improvements to medical visualization systems:

- Ability to work in unoccluded 3D views (from importance rendering)
- Ability to work in 4D views, such as segmenting structures at any step of an animating respiratory cycle (from volumetric animation)
- Ability to work in combinations of the most appropriate image modality for each region, for example identifying landmarks in a particular region using MRI or photographs while still working in the familiar context of the CT base image elsewhere (from photo or color mapping)
- Ability to recognize multiple segmented and unsegmented structures in a scene (from regional model coordinates and texturing)

This section attempts to contextualize how combinations of MGR's particular special abilities might map into particular medical image application domains. For the purposes of this discussion, applications of medical images are considered in three main domains:

- Anatomic education
- Diagnosis
- Image guided therapy (IGT)

Following (Yaniv and Cleary 2006), IGT is further broken up into three sub-domains:

- Procedure planning
- Intra-operative guidance
- Postoperative analysis

Some of the main points of this discussion are summarized in Table 2.

5.1.1 Anatomic Education

Medical images used in anatomic education commonly come from a particular "normal" reference subject such as the Visible Human, or they are taken from libraries of reference images of pathological or normal variants. Such reference images serve a complementary role to traditional medical illustration. Medical illustration is frequently concerned with providing what has been called "global" anatomic

Domain	Standard Interface	MGR Interface
Anatomic Education		
Understand anatomic shape, interrelations	Interact with reference atlas	Interact with atlas-like views of many individual reference images
Diagnosis		
Identify deviations from expected shape (3D) or intensity (2D)	Review using static surfaces or slices	Review in 4D, best modality per region, relate surface and deep features
Image Guided Therapy		
Procedure Planning		
Segmentation	Work in 2D, single modality at a time	Work on 2D slice in 3D context, combination of best modalities
Planning & Evaluation	Work using static or serial images	Work in 3D/4D with intuitive region shading, see effects of scenarios such as patient motion or position
Intra-operative Guidance		
Relate world to plan	World object immobilization or fiducial tracking on 2D slice	Project procedure-time images back into the 3D planning scene
Analysis or Tracking	*Same as listed above under diagnosis*	*Same as listed above under diagnosis*

Table 2 Common clinical applications for medical images and potential roles for MGR methods.

information in this dissertation. Global anatomic information is about shape and about spatial and functional interrelationships between anatomic structures. Certain normal reference images may be viewed with similar global goals, as in the case of the heavily preprocessed VoxelMan, which has many prerendered 3D views emphasizing various global qualities of the atlas anatomy. However, library images are typically intended to be interrogated slice-by-slice and used as real-world representatives for *local* image characteristics. For example, a particular reference CT image may be intended to help a radiologist learn to distinguish intensities that suggest healthy versus sick lung tissue.

MGR can augment this learning by providing more global context to these local tasks. The entire suite of MGR methodology is essentially designed to provide the advantages of an illustrated ("Netterly") or atlas-like 3D global view to particular library images without either obscuring local information or requiring exhaustive micro-segmentation. MGR's **regional texture mapping** and **importance rendering** methods allow multiple important structures in a target image to be rendered distinctly in a 3D view. **Color mapping** by pulling

color information from atlas or reference textures can additionally provide estimates for invisible local detail such as likely blood vessels or nerves that are invisible in the base modality or have been obscured by the rendering itself. When combined with a synthetic texture generation module, reference textures can be tuned to the particular anatomic variant being presented. For the example given above, an MGR rendering of an image showing a lung pathology could tune a standard healthy pink lung texture in suspicious regions to make it appear to be full of pockets of air where the interstitial tissue has disintegrated. At the same time, it is always possible to interactively inspect the image data of the base modality by switching off the synthetic texture or by placing an untextured cut plane nearby, so that the visualization can still be used as an example of the local intensity characteristics that suggest unhealthy tissue.

Educational training applications that simulate procedures on partially segmented reference images can be considered an aspect of the procedure planning domain discussed later.

5.1.2 Diagnosis

A primary difference between educational applications and diagnostic or IGT applications is that the latter are patient-specific by definition. Many diagnostic imaging applications are best done slice-by-slice because they are concerned with teasing out local image characteristics in specific regions and comparing them against expectations to identify pathology. However, there are also diagnostic tasks that require 3D shape and spatial relationship information, such as virtual colonoscopy. These 3D tasks have been typically restricted to high contrast structures (*e.g.*, the intestinal tract) and may rely on navigating through a surface-only endoscopic simulation.

MGR methods could be brought to both the 2D/local and 3D/global cases. In the case of characterizing local detail, it would certainly be inappropriate to enhance a rendering with atlas estimates of local features. However, related MGR's **regional color mapping** methods could be used to integrate additional data sources to a view. Region-by-region rendering from multi-modality imaging where corresponding objects have been identified in each source is one example. The next section presents a view that uses a CT image for volume rendering but

embeds MRI data near the prostate to suppress reconstruction artifacts from seeds or markers.

MGR's **photo mapping** method also provides a method for combining 3D images with surface photography in order to visually relate surface and deep features. One possible application for this would be to use serial photographs mapped onto reference anatomy to track surface changes over time. For example, current methods of diagnosing potential melanomas involve taking many pictures of the patient, laying them out, and comparing current pictures to those taken several months prior. MGR could provide a view that enables a physician to actually animate the skin change over time in 3D. Another application might be combining a 3D imaging modality with thermography, which shows near-surface features. Thermography is thought to have some benefit for breast cancer screening, but it suffers from many "false positives". Here the idea would be to present an interactive view where thermographic images mapped onto the surface suggest suspicious regions for closer inspection in the underlying 3D image.

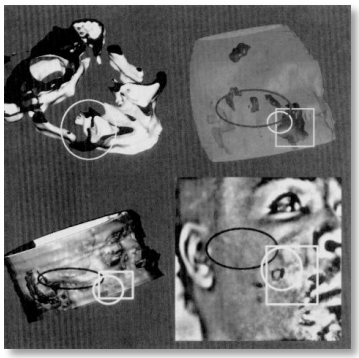

Fig. 124 Conclusion from (Oliver, et al. 1997) showing links between four "modalities", the isosurface of the bones, MRI, CT, and a photograph of the subject. Using MGR, this information could all be collapsed into a single view.

As an aside, Fig. 124 shows an interesting antecedent to MGR's emphasis on linking surface and deep features; (Oliver, et al. 1997) developed a mapping between photographs and a CT scan to determine if bones were broken by improperly swung batons in the Rodney King case (supporting the claim that they were). By using MGR and pre-segmenting a few important bony structures, all this information could be collapsed into a single interactive view.[23]

In the 3D diagnostic case, MGR methods can be used in particular to provide better vantage points. Although simulated endoscopic views have genuine advantages over physical endoscopy in the ease of the procedure (especially to the patient), surface-only renderings such as Fig. 6 provide no *medical* advantage over actual camera images. However, MGR's **importance rendering** methods could provide a real advantage in scene composition. Importance rendering frees simulated endoscopy from its naturally restricted field of view. Proposing an "open" virtual arthroscopy was one of the major results from Borland's flexible occlusion rendering against contrasted regions (recall Fig. 108). MGR extends such open views by making them dynamic and allowing for hierarchical importance from multiple ranked regions. Additionally, MGR's **color mapping** methods could be used in such views to clearly

[23] There was an interactive viewer for Oliver's project, but it has been lost.

distinguish and orient nearby regions. An MGR virtual endoscopy is presented later in this chapter.

One important future application of MGR's **volume animation** capability is interactive 4D rendering for regions such as the heart or lungs; this would allow a clinician to make decisions based on how a region changes shape over time.

5.1.3 Image Guided Therapy

Image guided therapy (IGT) involves three different kinds of tasks: **planning**, **intra-operative guidance**, and **post-operative analysis**. From a high level view, the post-operative phase has similar visualization requirements as the diagnosis domain discussed previously. However, planning and intra-operative guidance have very distinct needs.

Planning by Simulation

Not all image-guided medical procedure planning requires that the image be preprocessed to explicitly identify important anatomic regions. However, many of the most complex planning applications are intended to provide some level of simulation to verify that a proposed procedure is likely to achieve its intended goals. Such "planning by simulation" requires computing the expected effects of a procedure and evaluating the results with respect to particular important regions. An example of this kind of evaluation is precomputing the expected dose distribution of a radiotherapy plan with respect to target objects and nearby objects at risk.

"Planning by simulation" has already been a fruitful domain for 3D visualization. Levoy's earliest volume rendering methods were focused on radiotherapy planning, and many of the medical image visualization techniques referenced throughout this text have been motivated by such planning problems. Plan evaluation is, in some ways, a more appropriate target for 3D rendering methods than diagnostic tasks because plan evaluation is primarily concerned not with local detail but with understanding global shape and spatial relationships. For example, a dosimetrist is more interested in understanding how a configuration of beams intersect various target regions and organs at risk than they are in understanding the specific character of the tissue type in those regions.

There are several planning subdomains with different simulation mechanisms but related visualization needs.

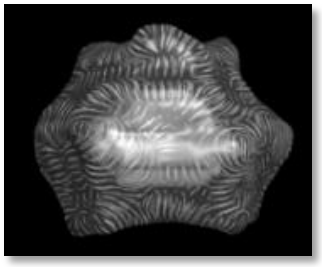

Fig. 125 (Interrante, Fuchs and Pizer 1997) explores the target domain of visualizing the *surfaces* of anatomic shapes with respect to dose distribution.

- **Radiation simulation**, which uses proxy geometry to approximate beam positioning and computes a likely dose distribution that must be evaluated relative to target and at-risk anatomy. Many of the visualizations that have been proposed for radiotherapy planning, such as (Interrante, Fuchs and Pizer 1997) (Fig. 125), focus on evaluating the shapes of static nested surfaces representing an isodose boundary and a target region. An example view presented later in this chapter uses MGR's interactive **volumetric animation** and **importance rendering** to display the volumetric effects of possible error from setup, internal deformations, or time dependent change such as respiration on an expected dose distribution. Interactive control over such a deformable scene could be a major improvement in any planning domain given that most planning assumes only rigid scene motion if any.

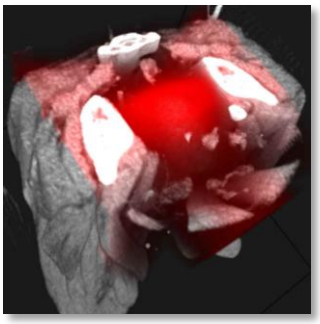

Fig. 126 An MGR view showing a patient image with the expected dose distribution overlaid in red.

- **Mechanical simulation**, such as the target problems described in (Bullitt and Aylward 2002) or (Konrad-Verse, Preim and Littmann 2004) (discussed earlier in the section on clipping), which both use visualization-related techniques to simulate and improve surgical resections. MGR's **model-coordinate-based cut geometry** provides several similar advantages in orienting and evaluating potential resection geometry.

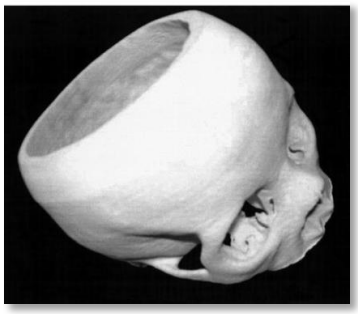

Fig. 127 A life-size plaster model of a virtual craniofacial reconstruction simulation from (Piatt, et al. 2006).

- **Reconstructive simulation**, which integrates computer assisted design with image guidance methods for simulating the results of reconstructive surgery. A proposed application of MGR's **photo mapping** and **volume deformation** methods is to provide "quality control" for craniofacial reconstruction as described in (Piatt, et al. 2006). The reconstruction of the osseous elements (Fig. 127) implies a deformation field on the original 3D patient image. A rendering of the original data could be first painted with a patient image and then warped according to the deformation field to provide an estimate of the surgical results. Such a rendering might be used for both evaluating the procedure plan and as an intra-operative reference.

MGR's **model-coordinate based solid texturing** provides a general advantage across the planning domain by allowing a clinician to be able to identify, orient, and spatially relate multiple important regions

throughout the volume. This is particularly true since a relatively detailed image segmentation of important anatomic structures is likely to be available.

Because planning by simulation tasks require evaluation with respect to important structures, these tasks are usually preceded by a **segmentation** phase to identify relevant anatomy. Segmenting an image has requirements similar to both the 2D and 3D diagnostic tasks. 2D views are required to adequately characterize local detail and create or edit boundaries. 3D views of the segmented surface are more appropriate for understanding whether the structures identified have appropriate shapes and spatial interrelationships.

The same MGR **regional color mapping** methods suggested for application in diagnosis tasks are applicable here as well. The next section of this chapter presents a possible MGR view taken from radiotherapy planning for the male pelvis. In the base CT image the prostate target is difficult to identify, but it stands out in the MRI. A combination view presented in 3D enables the clinician to pick out landmarks in the most useful imaging modality. Providing a volume rendered 3D context from the CT enables the clinician to understand the shape and spatial relationships of the segmented object to both other explicitly identified regions and to nearby unsegmented structures.

MGR's access to **model coordinates** can also be used to automatically visualize additional difficult to segment structures, such as the lymph levels in the head and neck. These regions (Fig. 128) cannot themselves be identified but are determined by landmarks on several surrounding objects. Generating these structures automatically is impossible without model coordinates that can distinguish, for example, the top from the bottom of an object. However, given suitable model coordinates, landmarks in the lymph level guidelines can be converted to model-coordinate uvt format. Then the model coordinates of the segmented neighboring objects imply a set of landmark positions that can be taken together to visualize the extent of these lymph levels. Typically lymph levels are displayed in terms of an atlas patient rather than for a particular clinical subject, which draws a connection between such visualization and the idea of showing "hidden features" discussed in relation to MGR's **atlas color mapping**.

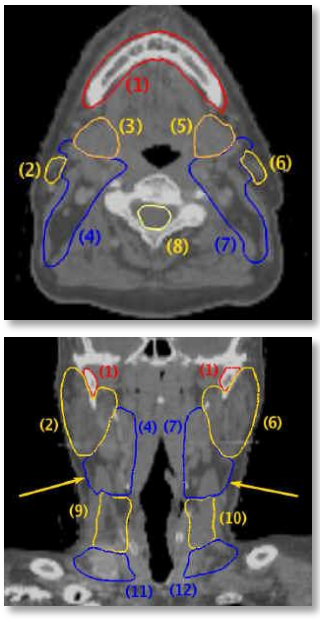

Fig. 128 Lymph levels are derived based on landmarks from nearby structures.

Intra-operative Guidance

Intra-operative guidance is primarily concerned with relating a real world patient on the table to an annotated planning image. Most current planning does not or cannot account for a deformation relationship between the planning image and the patient on the table, but MGR enables interactively controlled views of such relationships using its **volumetric animation** methods. Because most planning assumes rigid relationships between the plan and patient, intra-operative guidance usually reduces to guaranteeing proper patient setup. The simplest means of patient setup is a restraint system that immobilizes the patient in an aligned position, such as a stereotactic frame in open neurosurgery or a face mask in head and neck radiotherapy. Another method is to somehow mark a region of interest with an easy to track "fiducial marker" such as a small metal sphere embedded in an internal structure or a tattoo marked on the surface. Such fiducial markers allow the patient to be aligned with the planning image. However, no reverse mapping has been considered – i.e., given a tattooed surface, there is no way to guarantee that the tattoos have been put on in the right place according to the plan. An example later in this chapter uses MGR's **photo mapping** method to provide such evaluation.

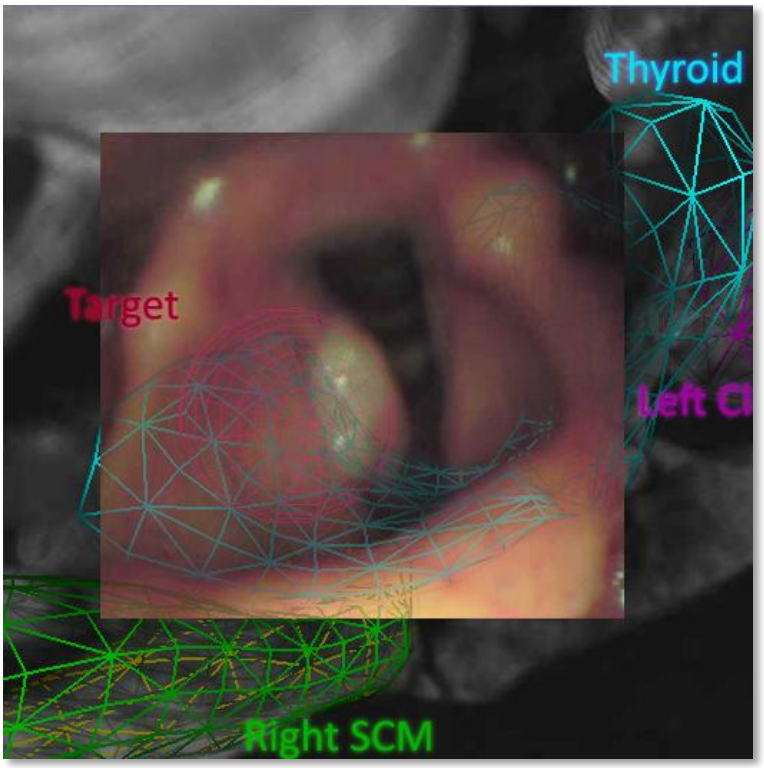

Fig. 129 MGR's endoscopic "guided tour" view discussed later in this chapter overlays 3D targeting and landmark information onto the 2D endoscopic view.

In the setting of minimally invasive surgery (MIS) via endoscopy, intra-operative guidance involves registering the scope position and the planning image. Typically this is displayed as a simple 2D projection of the scope position onto an image slice. MGR can improve such displays by providing context outside of the natural imaging limits of the scope both by mapping data from the planning image into the endoscopic view or from the endoscopic view into an open field of view virtual endoscopy. Again, the planning image, scope image, and any local deformation implied by the motion of the scope are just additional data sources for MGR's pipeline. Enhanced multi-source endoscopy is described in a later section of this chapter, and it is considered as a form of augmented-reality-guided surgery in the conclusion.

Of IGT's major domains, 3D visualization is most significantly adopted in surgical planning and guidance tasks. Applications of image guided surgery (IGS) or computer aided surgery (CAS) have been surveyed in (Grimson WE 1999) and more recently in (Yaniv and Cleary 2006). Many clinical visualization tools have been developed for IGS applications by extending the free open source visualization package 3D Slicer (www.slicer.org), which itself builds on the NA-MIC software libraries such as ITK and VTK. The slicer publications list provides an interesting recent overview of many types of clinical visualization applications. (Taylor 2000) includes a more formal survey of volume visualization applications and has a broad overlap with the bibliography of this dissertation. (Zuiderveld, Meissner, et al. 2005) documents a panel overview on the relevance of volume rendering to clinical applications and observes generally that initial research has been into methods for speeding it up but that the focus is shifting to application and intent-driven research, although few examples are discussed explicitly.

With a few exceptions, image guided radiotherapy (IGRT) has seen relatively less emphasis on 3D planning tools. Paradoxically, this is likely due in part to the compounded information in these scenes from imaging, segmentations, beams planning, and dose distributions, which makes them exactly the scenes that need improved global comprehension. The next section considers how MGR methods could be used to improve important IGRT 2D/local visualization tasks such as segmentation by adding 3D/global information and how some global visualization tasks such as plan evaluation could be improved by moving them into a 3D view without sacrificing local detail.

5.2 MGR Applications in Adaptive Radiotherapy

5.2.1 Clinical Goals

External beam radiotherapy (EBRT) involves a detailed planning phase and has a relatively complex workflow diagramed in Fig. 131. Patients are imaged, and this planning image is segmented to identify target regions and nearby "at-risk" structures. A dosimetrist then proposes a therapy plan by arranging several virtual "beams" around the planning image. The expected dose with respect to each of the important structures is computed, and the plan is refined accordingly. The goal of the planning is to deliver adequate radiation "dose" to the target region while sparing as much as possible the nearby normal tissue.

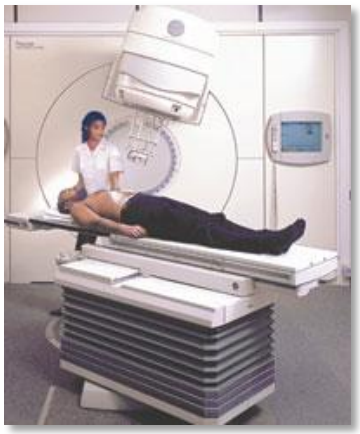

Fig. 130 Linear accelerator used for external beam radiotherapy (EBRT).

Given a plan, the therapy itself is divided up into a course of treatment "fractions" that will be delivered over several sessions. This makes the task of reproducibly aligning the patient with the plan extremely important. Patient setup is typically limited to aligning external markers, such as tattoos, with intersecting lasers that indicate a planning space reference position. However, the patient's internal anatomy may also move or change shape from day to day or even from moment to moment. For example, the target prostate tissue can be shifted and deformed between treatment fractions as the bladder empties and fills. Respiration can cause similar kinds of anatomic changes in the abdomen within a single treatment fraction.

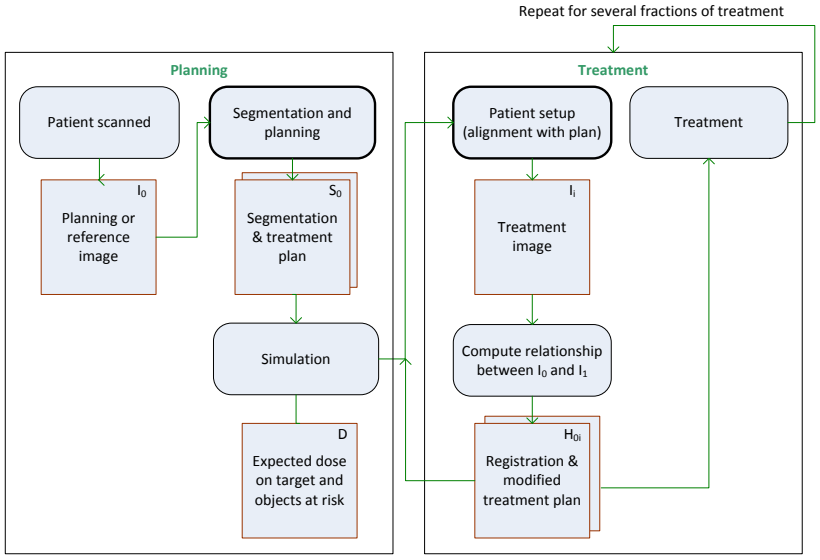

Fig. 131. Workflow for adaptive radiotherapy. Main components are planning and treatment. The MGR applications described here could be used in the segmentation, planning, and treatment setup phases.

Adjusting an EBRT plan to account for treatment-time anatomic shape change is called "adaptive radiotherapy" (ART) (Yan D 2000)(Wong JR 2005). ART is based on mappings between planning images and images collected at treatment-time. Evaluating the effects of such mappings on the expected dose delivered to the various important anatomic structures in the scene is a complex spatial task; yet it is typically limited to 2D views and a few quantitative summary measurements such as the volume overlaps of important regions before and after a registration. Planning in the presence of such anatomic change or potential patient setup error can account for considerable clinical time.

The next subsections demonstrate how MGR methods can be used to improve the particular tasks of segmentation, planning, and patient setup. From a high level, the MGR improvements allow the clinician to do 3D work in a regionally appropriate image modality, in the presence of potential error, and in a setting that ties the planning image to the patient in the world. The three clinical application views described are for **3D cross-modal segmentation**, **interactive planning under error**, and **patient setup verification**. Each section includes a short overview of the problem being addressed, the view itself, and a discussion of what MGR methods were used to improve comprehension for the task.

Appendix: Implementing the Planning Under Error View Using mgrView on page 152, walks the reader through the implementation of the planning project using the mgrView library, detailing how to preprocess input data for mgrView's file loaders and how to create a new shader program and integrate it with the library.

The examples shown here are interactive mock-ups or vignettes. At present, mgrView does not provide a complete solution for allowing a clinician to interactively segment, plan, or update patient positioning: it is limited to providing views on data. Integrating mgrView functions into a framework such as 3D Slicer could provide many of the editing capabilities required for a complete solution.

5.2.2 3D Cross-Modal Segmentation View

Identifying the boundaries of target and at-risk regions is a crucial task in preparing a patient image for planning. A clinician usually delineates these boundaries slice by slice. Many structures are difficult to discern in a CT image; for example, prostate tissue is relatively homogeneous

and in CT images has no clear boundary with some neighboring structures. The combination of difficult to identify structures and views limited to axial slices leads to *ad hoc* segmentation heuristics like "find a particular bony landmark, then count down three slices and assume the prostate starts there".

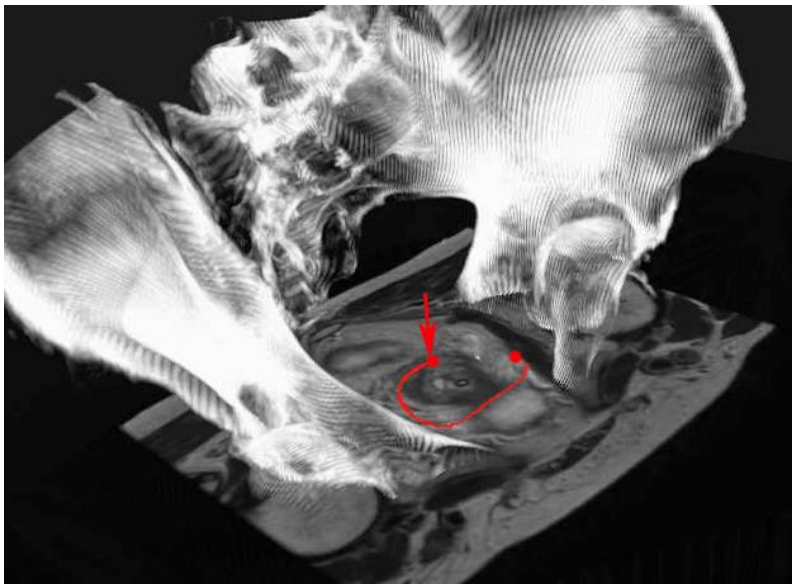

Fig. 132. 3D segmentation in mixed modes. Volume rendered structures from the CT image provide global context while the clinician can segment on a slice drawn from a corresponding MRI. Fig. 134 shows how the CT values near the prostate had been corrupted by artifacts from the metal fiducial marker visible in the center of the prostate region.

The goal of this vignette is to provide a clinician with both local detail and 3D context near a particular anatomic region to support a segmentation task.

The MGR vignette shown in Fig. 132 addresses each of these issues. First, while the user can still draw on a 2D slice, the slice is presented in its volume rendered 3D context and can take a continuous range of orientations. If a neighboring structure such as the bladder has already been identified, the slice can be oriented and positioned automatically according to local model coordinates and directions. An example might be positioning and orienting the slice to naturally emphasize the most proximal points between the bladder and the likely position of the prostate.

MGR Methods Required

- Model-coordinate driven clipping
- Regional color mapping from an alternate modality

Second, while CT is the most important imaging modality for radiotherapy planning because it directly measures the tissue properties required for computing dose distribution, MRI can also be a useful imaging modality despite its potential geometric inaccuracies. In this case, the CT values of homogeneous regions such as the prostate have been swapped out to allow the clinician to pick landmarks or delineate boundaries with stronger tissue type differentiation from MRI (see Fig.

133) in the region of interest but with the familiar context of the CT data elsewhere.

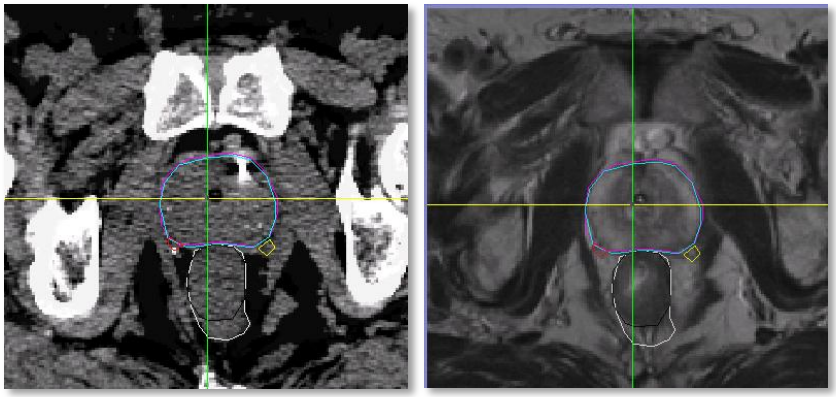

Fig. 133 Standard slice-by-slice view used during segmentation; the colored contours are the region boundaries drawn on this slice. The CT image on the left shows very little tissue differentiation between the circled prostate region and its neighbors compared to the MR slice on the right.

One particular example of the utility of this is when metal fiducial markers embedded in the prostate for tracking create CT reconstruction artifacts that obscure the target region, as in Fig. 134. Here, MGR's special capabilities for integrating multiple image sources has been used to replace the CT values on the entire working slice with data from a fused MRI. More complex potential solutions could be imagined using MGR's color mapping methods. One particularly interesting application might be using the marker positions identified in the MRI to identify untrustworthy regions in the CT so that they can be replaced with an appropriate reference solid texture. Such solid texture "in-painting" for gas bubbles in the rectum based on the methods of (Cheung, Frey and Jojic 2005) was originally discussed by Joshua Levy in an unpublished report from 2005.

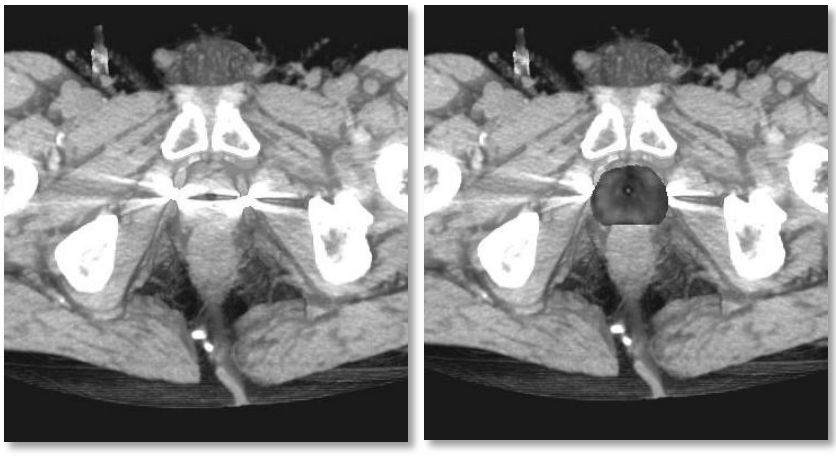

Fig. 134 The prostate region in the CT-only volume rendering on the left is obscured by the artifacts from the fiducial markers. The hybrid rendering on the right preserves the clear tissue distinction in the target region.

5.2.3 Interactive Planning Under Error Vignette

Radiotherapy planning and plan evaluation is usually done in part slice-by-slice (Fig. 135) and in part with projections such as the "beams eye view" to verify that the target regions can be "seen" and the nearby objects at risk are "hidden". Evaluation is also based in part on summary values such as dose volume histograms that measure radiation dose to target and at-risk structures.

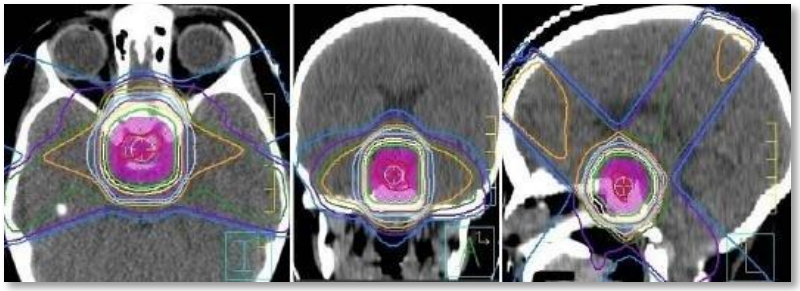

Fig. 135 A common 2D dose evaluation visualization showing isodose contours projected onto individual slices. 2D views can be quite useful for understanding local tissue types, but they are not necessarily optimal for understanding the 3D spatial relationship between the expected dose and the target region. (Image from (Mosleh-Shirazi, et al. 2004))

The vignette proposed provides several directions of visualization extensions to the basic means of evaluating radiotherapy plans described above.

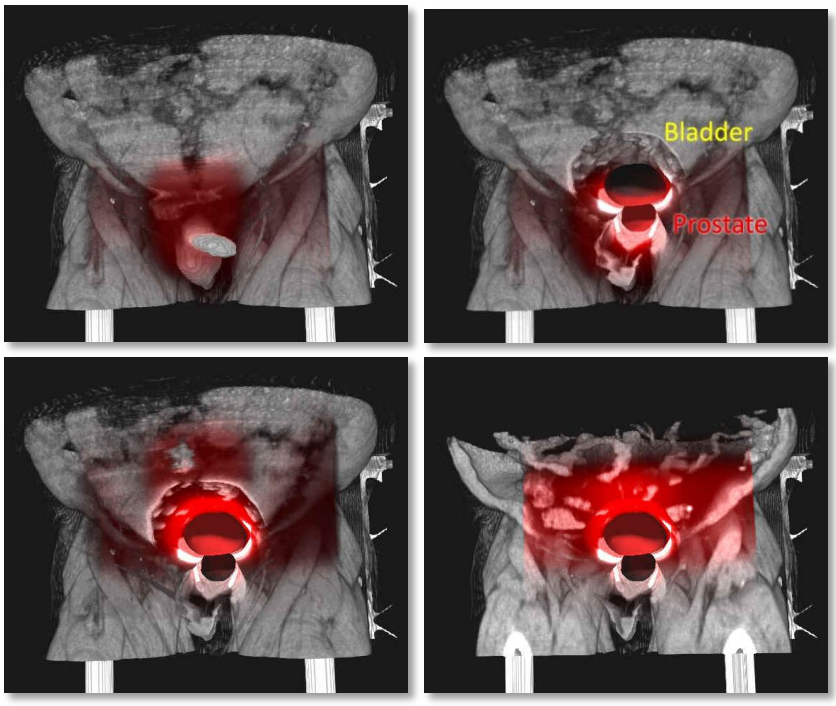

Fig. 136 Effect of error on expected dose. Top left, dose distribution overlaid near the surface where the A/P beam enters the target region. Top right, unoccluded view of the prostate target region below the at-risk bladder with expected dose overlay. Bottom left, a small rotation applied to the patient leaves the prostate cold. Bottom right, further clipping reveals the effect of the altered dose distribution on nearby unsegmented structures.

The goal of this vignette is to enable 3D evaluation of a radiotherapy plan with respect to multiple anatomic regions in the presence of a range of possible setup errors or possible internal shape changes.

MGR Methods Required

- Model-coordinate driven clipping
- Interactive volumetric animation
- Dynamic importance rendering
- Regional color mapping from alternate imaging modality
- Model-coordinate implied regions

As with the segmentation vignette described in the previous subsection, this view is 3D, but MGR techniques can be used to slice the data into 2D cross sections that can be oriented with respect to natural directions (*i.e.*, along, across) for local regions for local evaluation of dose vs. shape. Moreover, MGR provides for a dose overlay that modulates both the identified and unidentified anatomic structures. This enables the clinician to understand the dose with respect to locality – not just how much dose is received per object, but where it is received throughout the object. Regions can be individually clipped away to expose hidden interior structures, and the dose can be visualized both inside and on nearby regions to understand what is happening to unsegmented but discernible structures in the scene.

Finally, the primary novelty of the mgrView vignette for this application is the addition of interactive control over variable error parameters that apply a global rigid transform or local deformable transform to the patient while leaving the dose field in place. This is implemented as described in section 4.1, *Volumetric Animation*. This effect can be used to simulate and understand the effects of possible errors in setup, gantry realignements, patient motion, or internal anatomic change such as the bladder filling and emptying. Such evaluation may suggest minor adjustments to the beams to make them more robust to possible changes, or, when evaluating a plan against the registration to a real daily image, a decision to replan the patient. This same effect with multiple daily images could be used in segmentation as well to allow the clinician to identify or edit a boundary on any step of the series of registered images.

See the *Appendix: Implementing the Planning Under Error View Using mgrView* beginning on page 152 for a complete walkthrough of implementing such a vignette using the mgrView library.

This method could be further extended by adding appearance effects from atlas texture mapping. A basic problem is that dose volume histograms summarize dose to explicitly identified objects only. The method as outlined above additionally displays the dose distribution with respect to *discernible* but non-segmented structures, such as the spine.[24] Applying MGR's model-based atlas color mapping methods,

[24] "You can have the best summary statistics in the world, but it doesn't matter if you packed all your error into the spine." – *Ira Kalet, Univ. of Washington*

one could additionally design a display of a dose distribution with respect to *hidden but likely* radio-sensitive structures such as nerves, lymph levels, or internal structures by pulling relevant regional textures from atlas sources.

5.2.4 Patient Setup Validation Vignette

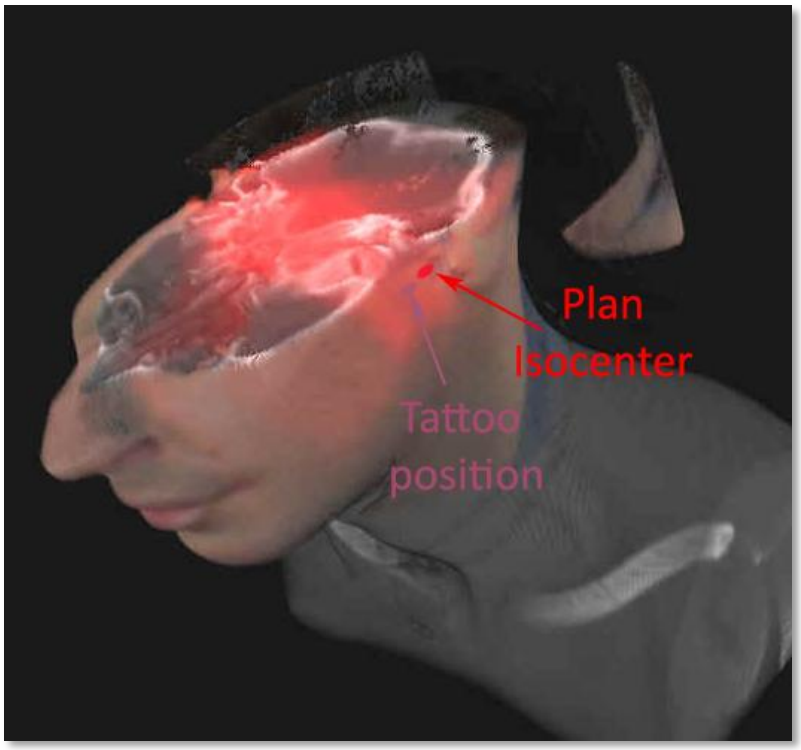

Fig. 137. A rendering showing the patient's alignment tattoo mapped back onto the planning image with dose overlay to provide feedback regarding the suitability of the world-to-plan registration. In this case, the tattoo is not in the position expected by the plan.

Patient setup is the problem of aligning the real-world patient with a virtual procedure plan. This must be done reproducibly over many days for fractionated treatment. Usually the only information available for this task are visual cues such as skin tattoos that can be lined up with lasers in the room to at least align the patient's target region with the planned dose isocenter.

The goal of this vignette is to help estimate the position of deep features based on surface features.

MGR Methods Required
- Patient photo mapping

MGR can be used in dual complementary capacities to address this problem. First, augmenting views from cameras in the treatment room with a projection of the proposed planning image (recall Fig. 31 of RANDO) could provide the same kind of visual alignment estimate as tattoos, but for the entire plan rather than just for the isocenter. Second, using the methods described in section 3.3, *2D Color Transfer*

from Patient Photos, to augment the 3D planning image with the photograph from the treatment room (Fig. 137) provides several additional guidance cues.

- Tie the plan to the patient and catch blunders such as left/right symmetry transforms or planning the wrong patient
- Confirm that the target implied by the tattoos does indeed align with the dose isocenter
- Visualize where a radiation beam should fall on the patient without having to make mental estimations about the actual surface features shown on the CT. This addresses shortcomings in the practice of "light field setup" which provided no good link between the expected light field positions and the patient's surface appearance.

Beyond photography, the same methods could be applied to visualize registrations between a planning image and various other data sources. One intriguing setup source might be a reference *surface* such as is collected by the VisionRT (www.visionrt.com) stereo camera setup system (Fig. 138), which could be used for patient setup based on geometry. Another potential setup source might be non-visible light photography such as thermographic (heat) images (Fig. 139), which could be used for patient setup based on veins or other warm, near-surface landmarks. Applying this vignette to serial patient images taken at treatment time would provide a means to track skin reactions with respect to the expected dose distribution and the initial treatment photograph.

Fig. 138 A VisionRT surface (green) aligned with the corresponding CT skin isosurface (purple).

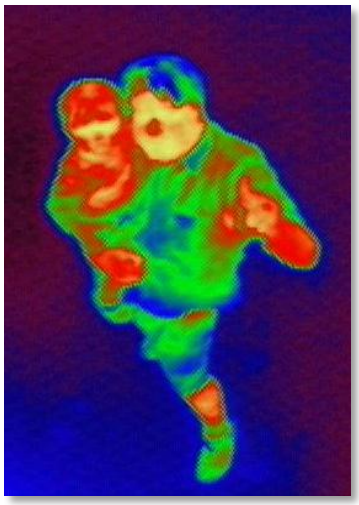

Fig. 139 Thermographic image of the author holding his oldest son at age 18 months, taken at The Tech Museum in San Jose. Thermography can show near surface features.

5.3 Enhanced Endoscopy from Multiple Modalities

The introduction of this section is paraphrased from a related RC1 grant application written principally by Dr. Julian Rosenman and submitted through UNC Hospital's Department of Radiation Oncology.

5.3.1 Clinical Goals

There are two main ways of determining the anatomic extent of a tumor: 3D imaging (*e.g.*, CT, MRI, or PET) and direct visualization via endoscopy. In the head and neck, nasopharyngoscopy is a minimally invasive diagnostic procedure in which a small camera called an endoscope is inserted into the patient's nasal passage and passed down through the pharynx. An attached cable serves to control the position and orientation of the camera, to provide light at the target via optical fiber, and to transmit a video signal back to a monitor. The endoscopic rig may include additional instruments such as a biopsy needle or pincer for collecting tissue samples at a target site.

Computed visualizations of 3D images and direct endoscopic evaluations are both particularly important when tumors are entangled with multiple critical normal tissues. Nasopharyngoscopy enables a clinician to see the mucosal surfaces of the cavities but not the internal tissues. 3D imaging provides information about the deep infiltration of the tumor, but it does not provide a visualization of the mucosal surfaces. So, for example, a patient with a submucosal abnormality on CT may have normal-appearing mucosa. Therefore the physician may require additional guidance from a 3D planning image to direct a biopsy. Or conversely, a CT may give incomplete information about the extent of a tumor's infiltration into the submucosal region. Therefore the physician may require additional guidance from direct visualization to estimate a target region for treatment planning. However, there is presently no easy way to *visually* register the information from these two assessment tools with each other beyond simply displaying the probe's world position on a slice from a CT.

Virtual endoscopy has become accepted standard of care for certain clinical procedures such as colonoscopy (as early as the 1990s, see (Robb 1996), (Nain D. 2001)), but simulated views (Fig. 140 left) lack color information and cannot be used to guide actual biopsy. Real views

(Fig. 140 right), by contrast, obviously lack targeting information from any planning that has been done.

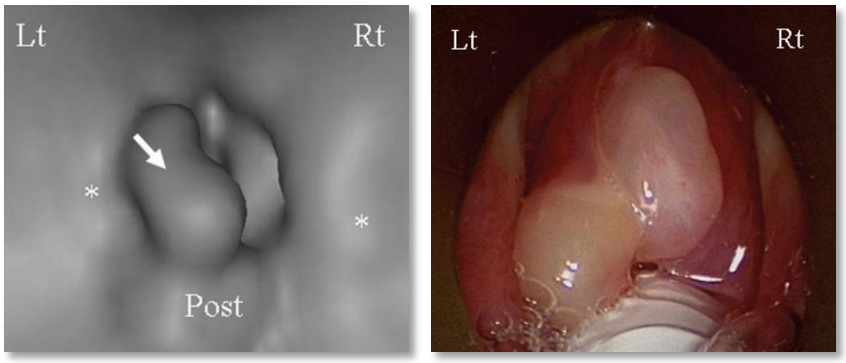

Fig. 140 Left, virtual nasopharyngoscopy and right, corresponding image from real procedure.

MGR methods can be used to create hybrid views from simulated and real views. There are two proposed applications for MGR in this domain. The first is to provide **online guidance for the biopsy** by integrating hidden features identified in 3D imaging directly into the endoscopic view. The second is the complementary task of integrating video from the endoscopy back into the 3D rendering to enhance **open field of view virtual endoscopy** with actual endoscopic images. This vignette could serve roles both for online guidance by indicating the probe position relative to the target region in 3D and for offline review. Taken together, these two vignettes would provide a powerful update to the standard guidance method of projecting the probe position onto the slice of a planning CT.

These techniques are expected to be of particular interest in virtual nasopharyngoscopy, which is an open area with little active research due at least in part to the large number of both discernible and inferred (lymph levels) critical anatomic structures and their complex spatial interrelationships. The mockup figures in the next two sections are accordingly taken from that domain.

5.3.2 Online Biopsy Guidance

Current methods for providing 3D visualizations suitable for augmenting endoscopic views are either limited to only a few poorly defined surfaces (*i.e.*, the surface-only virtual colonoscopy model), or they take months to prepare and so are not applicable to the treatment of an individual patient (*i.e.*, the VoxelMan model).

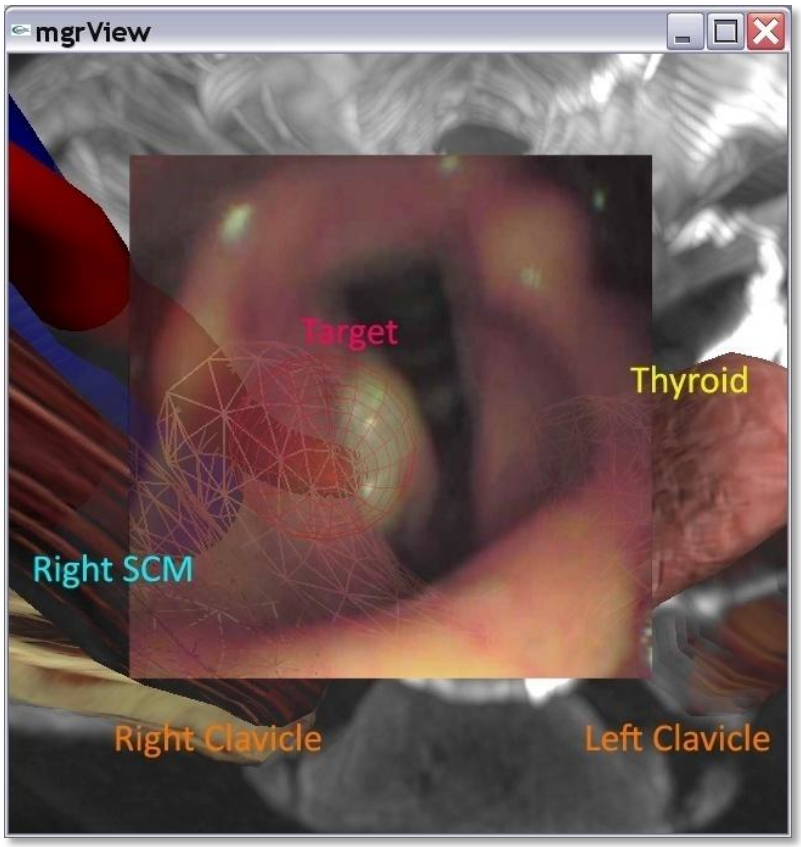

Fig. 141 A mockup of an mgrView "guided tour" 2D endoscopic display showing a sample scope view embedded in a 3D planning image with target and nearby "beyond-the-wall" structures overlaid. Fig. 142 shows the complementary 3D view.

The goal of this vignette is to augment the endoscopic view with information about the patient's anatomy in the hidden "beyond-the-wall" region relative to the camera.

Given a partial segmentation of a planning image, MGR can synthesize overlays for the camera video feed to show both target objects discernible in the planning images and other clinically relevant objects and regions implied by prior anatomic knowledge.

MGR Methods Required

- Combining 2D and 3D images
- Regional volume textures
- Model coordinate implied regions

The mgrView rendering engine runs fast enough to integrate this extra data in real time into the endoscopic view and overlay an annotated textbook-like "guided tour" of the nearby hidden features as the endoscope is advanced or endoscopic surgical instruments are utilized. It is expected that using this hybrid annotated view will improve diagnostic and therapeutic outcomes for endoscopy and endoscopic surgery for patients with head and neck cancer, in particular.

Method Outline

1. Collect a planning image, segment important objects and assign textures similarly to the head and neck project shown throughout chapter three, *Model Guided Appearance for Medical Images.*

2. Register the endoscope position and orientation to the CT image and render an MGR virtual view from the point of view of a corresponding simulated camera.
3. Combine the views according to various levels of virtualization, *e.g.*, camera only, camera + labels, camera + hidden objects, or virtual only.

5.3.3 Enhanced Open Field of View Virtual Endoscopy

Endoscopic guidance is frequently restricted to projecting the probe position onto a 2D slice of the planning image. This can confound the user's ability to understand the spatial relationship between the probe and nearby clinically relevant objects.

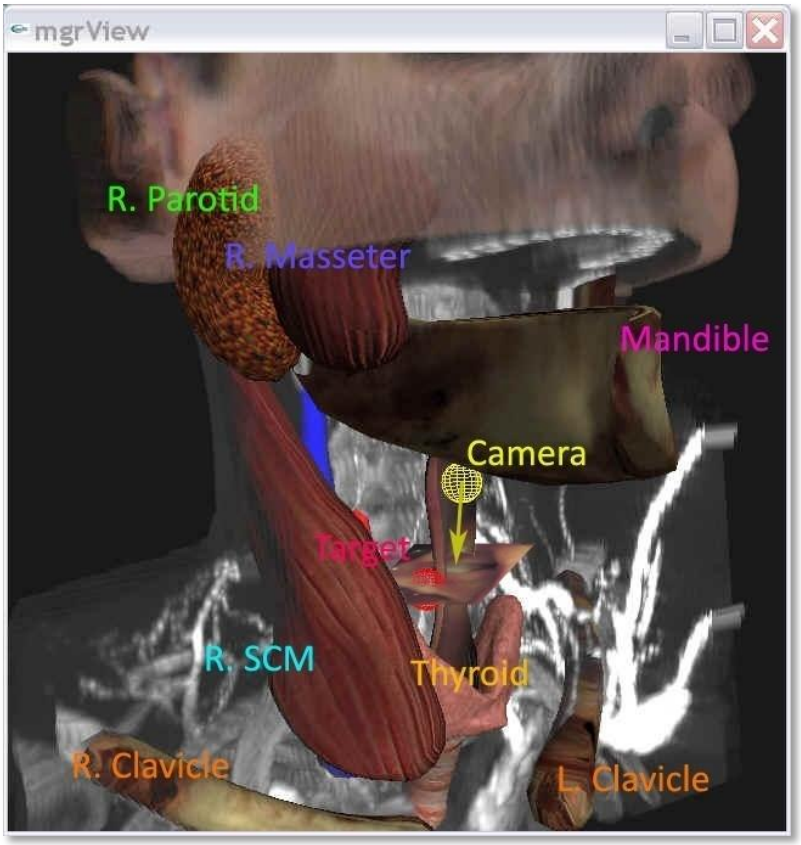

Fig. 142 A mockup of an mgrView open field of view virtual endoscopy enhanced with photomapping and online guidance information. The probe position relative to a target region is shown in 3D based on online probe position measurements. Color images collected by the endoscope are dynamically overlaid onto the CT.

The goals of this vignette are to provide an unobstructed indicator for the relative positions of the probe and beyond the wall structures for online guidance and to put the endoscopic images back into a 3D context for review.

Addressing the first goal, mgrView can easily take online probe position data and render a camera proxy into an open field of view endoscopic vignette as discussed previously. This can provide much needed 3D context for the spatial relationship between the probe and any target regions. This mapping additionally gives MGR all the information that it needs to project the current probe image back onto the CT image according to the methods described in Chapter 3. This photo-map enhanced image can then be used retrospectively for a color-correct virtual endoscopy, of either the fly-along-the-tube or the open field of view type that enables viewing from any point of view the physician desires. In either case, the virtual view could be further augmented by the same "beyond the wall" structures discussed for the previous vignette.

MGR Methods Required

- Photomapping
- Color mapping from atlas or synthetic sources
- Interactive volumetric animation
- Importance rendering
- Model coordinate implied regions

Method Outline

1. Use the methods from section 4.2, *Fast Importance Rendering*, to focus a view on the endoscopic path (an "open field of view virtual endoscopy")
2. Use online camera position information to and add geometry proxies for the camera and nearby clinical target regions to the scene
3. Use a virtual camera to project each frame into the planning CT coordinates and color the local CT data according to the endoscopic view using the methods described in section 3.3.3, *Rendering From Planar Images*
4. Collect the entire color volume for use in offline for color-correct virtual endoscopy

5.4 Evaluating MGR Methods

The aforementioned vignettes have shown a number of areas where MGR methods can be used to provide additional comprehension to particular clinical applications. Prototyping a full application would be a useful next step in evaluating MGR methods' strengths and in identifying and ameliorating its weaknesses.

6 Conclusions

Traditional medical volume visualization has focused on two main methods: slice-wise cut-planes that provide good local detail but sacrifice 3D understanding, and direct volume rendering (DVR) that provides good global comprehension but sacrifices the ability to interpret local values and comprehension of interior structures.

The goal of **Model Guided Rendering** (MGR) is to maintain the global comprehension from volume rendering while adding back some of the lost local detail and obscured structures. There are already a few methods that achieve such results, but they are either patient-specific but limited to a particular task, such as virtual colonoscopy, or generalizable but limited to a particular exhaustively micro-segmented atlas data set, such as VoxelMan.

MGR is an attempt to bridge the gap between generality in the task and subject domains. MGR can leverage a broad range of input data to produce flexible, high quality images. But it requires only limited manual intervention, so it can be applied to **a broad variety of patient-specific clinical tasks**. In particular, MGR is intended to support "Netterly" and other styles of volume rendering for specific clinical target patients. MGR is designed to be a display layer extension that sits on top of deformable model based segmentation technologies currently being actively developed by research entities like UNC and by commercial entities like Morphormics.

The key observation is that traditional volume rendering is limited by working in *voxel-coordinates* of a single source image. There is simply not enough information in a single CT image to address such complex questions as "What am I seeing here?" This is exacerbated in 3D visualization where the most common tool available for image interpretation is a fragile transfer function.

MGR addresses this by **relating information in multiple sources on a region-by-region basis** with *object-coordinates* from a few automatically segmented regions. Relying on increasingly powerful automatic image processing algorithms to define the image relationships frees the user from reliance on overly simple single-modality transfer functions to pick out and interpret 3D structures. Given an interpretation of type and orientation of a few important structures, relevant local detail can be added back into the scene based on a patient data collection when appropriate. Patient data collections are becoming increasingly complex and can comprise images from multiple 2D and 3D modalities, serial imaging studies, registration fields, and various spatial distributions, such as dose calculations for external beam radiotherapy (EBRT) planning. Additionally, atlas textures from color anatomic sections (the Visible Human) or synthetic sources may be available. MGR provides a framework that allows all of these sundry sources to play a part in the rendering pipeline.

6.1.1 Chapter Organization

This chapter has two sections: first, a **review of the dissertation thesis and claims** in the context of the materials presented; second, some meditations on the positive and negative **potentials of MGR visualization**, including **dealing with uncertainty**, **non-clinical applications**, and **extending MGR**.

6.2 Thesis & Claims Revisited

This dissertation supports the following thesis.

Image segmentation via **medial shapes provides an effective basis for guiding context-appropriate shading** in 3D medial image display by supporting regional color mapping from library or synthesized **solid textures**, **cross-modal images**, and **atlas data sources**. Precomputing a global "**scene catalog**" that collects multiple local medial-to-world and world-to-medial transforms enables these techniques in an **interactive object-order volume rendering framework**. This framework additionally extends other perception-enhancing effects such as **importance rendering** and **volume deformation** to dynamic scenes.

There are two parts to the proof of this thesis. First, the **methodology** claims restated in Table 3 have been presented. Second, several novel vignette projects enabled by these methods have been provided as **results**.

6.2.1 Methodology

Chapter three, *Model Guided Appearance for Medical Images*, presents a methodology for regional combination of data sources on the basis of coordinate systems provided by a sparse set of deformable shape models fit to important regions in the scene. The techniques discussed in this chapter for assigning scene appearance are summarized in the first and second claims of novel methodologies from Table 3.

Section 3.1, *Creating a Scene Catalog* presents a fast method for computing such **regional world-to-model (x2u)** and **model-to-world (u2x)** data mappings using m-reps (claim 2.1) in an object-order rendering setting by using programmable shaders to exploit graphics hardware acceleration (claim 2.2). The mappings created are themselves suitable for doing such transformations quickly on graphics hardware. These methods are described with respect the **m-rep shape model** because it implies a volume-filling coordinate system with intuitive directions such as along, around, and through the object. However, MGR could be integrated with other segmentation tools if given a method to infer a volume filling coordinate system.

Methodological Claims

1. Method for using medial coordinates to guide context-appropriate shading in medical images by regional color mapping from several different kinds of data sources
   - 1.1. Method for mapping and lighting library or patient-specific synthetic solid textures
   - 1.2. Method for mapping from 2D data sources such as patient photographs
   - 1.3. Method for mapping from 3D data sources such as cross-modal images or atlas data sources
2. Method for generating such renderings at interactive rates on relatively modest hardware by precomputing a "scene catalog" data structure and manipulating it in an object-order rendering framework
   - 2.1. Algorithms for computing world-to-medial ("x2u") and medial-to-world ("u2x") maps from a set of segmentations by medial shapes and a data structure for collecting these mappings together
   - 2.2. Algorithms for using the scene catalog in various ways through programmable shader hardware to do the mappings described in (1)
3. Refactored versions of important state-of-the-art volume rendering methods such as importance rendering and volume deformation that allow these techniques to be applied in dynamic scenes
   - 3.1. Object-order implementations for global and local volume deformation and for importance rendering based on ranked surfaces

Table 3 Claims revisited.

Section 3.2, *Simple Texturing for Volumes*, describes how to use regional mappings to **apply solid textures** in world space, which requires only an object label, or in model space, which requires the complete object coordinate system (claim 1.1). Section 3.3, *2D Color Transfer from Patient Photos*, details a variant of a world-space mapping that can be used **to push patient photographs into the 3D scene** (claim 1.2). Section 3.4, *3D Color Transfer*, details **cross-image mapping**, a variant of model-space mapping that uses a model to world and an inverse world to model mapping from another data set to transfer 3D image data from a different image or atlas into the scene using object-relative coordinates (claim 1.3).

The fourth chapter, *Model Guided Composition for Medical Images*, adds fast methods for scene composition and focus. **Global and local volume morphing** (the first part of claim 3.1) is discussed in section 4.1, *Volumetric Animation*, and **importance rendering using fast shadows and importance stenciling** (the second part of claim 3.1) is discussed in section 4.2, *Fast Importance Rendering*.

6.2.2 Results

Several clinical application "vignettes" are presented throughout the text as demonstrations both of MGR's methods and the kinds of novel visualizations enabled by MGR.

MGR's most general intended application is supporting high quality 3D volume rendering for particular target patients. Motivating examples throughout the methodology chapters are taken from a prototype "atlas quality head and neck" project that suggests how various types of **scene-catalog** (claim 1) driven **solid texturing** and **color mapping** (claim 2) could be used to increase understanding of multiple region types and orientations in volume rendering.

The fifth chapter, *Bringing MGR to the Clinic*, both overviews the types of medical imaging applications in which MGR could potentially play a role, and provides several specific examples that demonstrate how MGR methods are uniquely suited to address some important but currently difficult clinical tasks.

Section 5.2 presents application vignettes taken from radiotherapy planning. The first part shows how MGR's methods for **regional color mapping from alternate modalities** (claim 1.3) could be used to provide

context and suppress artifacts during 3D image segmentation. The second part shows how **dynamic importance rendering**, **volumetric animation** (claim 3.1), and **model-coordinate derived regions** could aid in understanding the effects of shape change or setup error on a 3D dose distribution. The third part shows how MGR's **photo mapping** method (claim 1.2) provides a means for evaluating the placement of patient setup tattoos relative to planning expectations.

The example application vignettes from endoscopic guidance are presented in section 5.3. The complementary view vignettes presented use MGR **photo mapping**, **solid texturing**, (claims 1.1 and 1.2) and **importance rendering** (claim 3.1) to provide a concise 3D navigation system for guiding a probe to hidden target anatomy.

In summary, the potential uses of MGR's special capabilities are demonstrated in the following ways:

- Assigning cross-modal or atlas solid textures according to model coordinates (claims 1.1, 1.3, and 2) is demonstrated in the head and neck atlas project shown throughout chapter 3, the cross-modal segmentation vignette in section 5.2.3, and the 3D endoscopy vignette in section 5.3.3.
- Assigning textures from photographs or video (claim 1.2) is demonstrated in the head and neck atlas project, the patient setup vignette in section 5.2.4, and the augmented endoscopy view in section 5.3.2.
- Applications for dynamic importance clipping and volume animation (claim 3) are demonstrated in the planning with error view in section 5.2.3, and the 3D endoscopy view in section 5.3.3.

The "radiotherapy planning under error" project is described in detail in an appendix that serves as an example of how to go about designing and building a new project using the mgrView library. This serves to show that the methods are not only potentially *useful* but also potentially *usable* as implemented in mgrView.

mgrView uses fast object-order algorithms for the core MGR methods described here and provides as well a framework for unifying many other state of the art methods for scene composition, such as global deformations and geometry based opacity modulation. Frame rates achieved by mgrView for the methods and scenes discussed in this dissertation are summarized in Table 4. mgrView includes a minimal

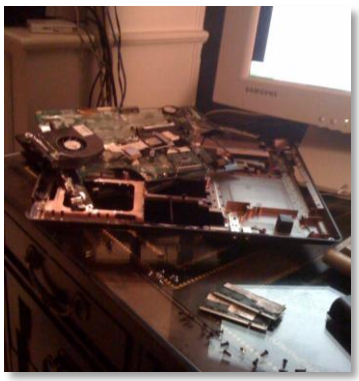

Fig. 143 The mgrView library achieves frame rates between 10 and 20 fps on a target laptop for most of the example scenes shown throughout this document. However, running mgrView did eventually overheat and crack the graphics accelerator in the disassembled laptop above.

windowing environment and user interface, but it is also intended to be embedded in other clinical tools, such as the Plan-UNC (PLUNC) radiotherapy planning system.

	NVIDIA GeForce 6200 (Desktop)		NVIDIA Quadro NVS 160M (Laptop)	
mgrView Rendering Task	**100 planes**	**200 planes**	**100 planes**	**200 planes**
Standard DVR with intensity windowing	12	8	20+	20
MGR Methods				
1. Regional Volume Texture	7	4	20+	10
2. Photo Mapping	7	4	15	11
3. 3D Color Transfer	7	4	20+	10
4. Global Deformation	4	2	15	10
5. Local Deformation	7	4	20+	12
6. Dynamic Importance Rendering	6	3	20+	15
Complex Scenes				
Radiotherapy Planning (methods 3,5,6)	4	2	15	9
Virtual Endoscopy (methods 1,2,3,6)	5	3	20+	15

Table 4 Summary of the best frame rates in frames per second (fps) achieved using the mgrView software library for various rendering tasks and scenes described in this dissertation. Methods 1 and 3 use the same code mechanism, so they have identical results.

6.3 Potentials of Model Guided Rendering

While the vignettes described here are encouraging, clearly there is a need to build a complete system to evaluate the *clinical* potentials of MGR and identify capabilities that are still required.

That said, there are several intriguing potential uses for MGR that may or may not have clinical relevance but are themselves worth further consideration. This section reviews several directions that have been considered for additional focus.

6.3.1 Dealing with Uncertainty

Assigning synthetic textures or atlas colors to a particular target patient begs the point that this potentially adds spurious information to the scene. Worse, the segmentation or registration being used to guide the rendering may itself have some level of uncertainty, so visual estimates in some areas may be both incorrect and in the wrong place. Worse yet, mgrView itself makes a variety of simplifying assumptions about the data that are problematic, such as linearizing multiple time step warp fields or using standard m-rep surfacing algorithms with radius 0 medial spokes for unstable approximations of object coordinates on the medial sheet. Thus, in the worst case, visual estimates in some areas may actually be based on completely incorrect assumptions. If such images are misused for diagnosis, the results could be dangerously misleading.

To this I would respond that at least the non-implementation-specific sources of uncertainty will be present in any volume visualization because assumptions about the character of the data will *always* be made. CT reconstruction is subject to major artifacts when algorithmic assumptions are broken; yet no one misdiagnoses a patient as having a mouth full of high density material configured in starbusts centered around her fillings. Classical volume rendering is subject to fragility of parameters that easily allows noise to be emphasized and signal suppressed. MGR bases its assumptions on a prerequisite robust segmentation, which actually gives the user significantly more explicit control over which assumptions will be made with regard to a particular scene. An expert user can tune segmentation or registration methods and set thresholds when designing the project for how credible interpretation or atlas sources must be to be included in the view.

Indeed, this very uncertainty could provide an additional channel for MGR to use in guiding how particular anatomic regions should be displayed. An MGR style has been proposed that would use a non-photorealistic (NPR) sketch texture such as in Fig. 144 where information was uncertain, a more "realistic" texture where structures were more confidently understood, and a blend of the two methods to show varying levels of certainty in the interpretation of the underlying data. A measurement of local uncertainty could be provided by a tool such as (Levy, et al. 2007) (Fig. 145), which uses deformable shape object statistics to automatically flag of suspicious local properties such as abnormal shape or intensity distributions.

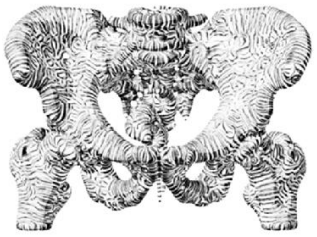

Fig. 144 Surface sketch rendering for anatomic shapes from (Interrante, Fuchs and Pizer 1997).

However, to some degree, it is still somewhat too early to discuss explicit measures on uncertainty given the youth of the system and the challenges involved in simply gauging and eliminating the implementation-specific sources of error in the presence of presumed perfect data.

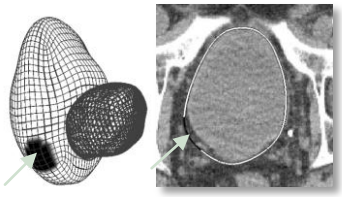

Fig. 145 Display of surface non-credibility from (Levy, et al. 2007). The dark region on the larger mesh is the area indicated on the slice shown on the right that has likely been improperly segmented.

One of my favorite observations from visualization literature is from (Simpson, et al. 2006) in regard to visualizing uncertainty of image-to-world registrations. After establishing a complex visualization designed to show uncertainty in indicating a target region for a simulated biopsy, the debriefed physician subject commented that all he needed to know was that the registration *was* uncertain and that then he would just "use a bigger drill bit" to take the sample. He didn't need to know where or by how much. Edward Chaney, a research and former clinical physicist in radiation oncology, has made similar comments about the current disconnect between the increasing accuracy of image segmentation and the limited precision of most external beam radiotherapy delivery devices. Given uncertainty in an image segmentation, the solution in external beam radiotherapy is usually to simply "use a bigger margin". That is, detailed understanding of uncertainty does not, *per se*, lead to more careful targeting; rather it leads to relaxing the precision of the targeting method.

6.3.2 Non-Clinical Applications of MGR: Understanding Principal Warps

There are a variety of volume visualization tasks where multiple data sources or relationships among multiple data sources need visualizations. An example of a task that requires visualization of multidimensional volumetric animation is evaluation of an image space defined according to "principal warps". mgrView's method for volumetric animation according to deformation fields was originally developed to aid in evaluating statistical analysis of fluid registration from an atlas to a number of target images. The mgrView interface provided a user interface to control how much of which displacement field was applied to which image. By applying a weighted combination of warps to a source image, a new "image" could be produced.

Fig. 146 Matlab's single-slice "brain" phantom function called with randomly sampled parameters.

The idea of sampling from statistics of dense registration fields to produce new images was originally developed in (Chen, et al. 2002) as a means for synthesizing a database of computational image phantoms for the region near the kidney. Segmented image phantoms are necessary for validating segmentation or planning tools. Typically they are produced by creating a parametric representation of a set of structures and then sampling from a distribution on those parameters to generate novel data sets (Fig. 146). In the case of statistics of dense registration fields, the parametric representation is how much of each of the basis warps found by principle component analysis (PCA) are applied to the source image to deform it and its corresponding segmentation into a new target image. The main problem with this method is that the implied image space is mostly full of anatomically unlikely or otherwise illegal images and only sparsely populated with useful instances. For example, PCA is not guaranteed to preserve diffeomorphism in its decomposition, so arbitrary combinations of principal warps can easily tear or fold the space. Thus, the parameters cannot be randomly sampled to produce a mapping from the atlas to a "new" image without risking grossly illegal results, as seen in Fig. 147.

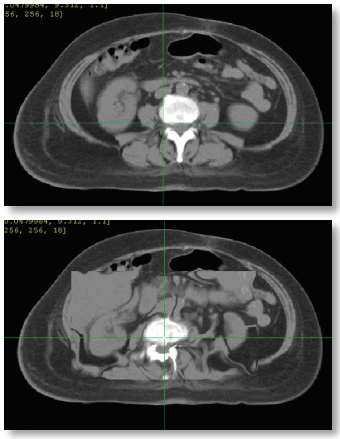

Fig. 147 Top, a slice from a source image and bottom, the same slice under an obviously unlikely sampled registration.

However, using the mgrView deformable animation tool, warp combinations could be sampled and applied in an interactive fashion to animate over a continuous range of potential "new" images. Simple visual inspection could serve to eliminate the large number of obviously illegal candidates and focus closer inspection on likely candidates. Parameters of credible images could be saved with the push of a button.

6.3.3 Extending MGR

I am confident that the increasing power of automatic medical image segmentation will lead to important improvements to 3D clinical visualization over the next ten years. MGR is one possible approach for integrating data from segmentation into rendering, but it requires considerable further development to reach the kind of robustness needed for general clinical applications.

In particular, continued development of the MGR framework is strongly dependent on two supporting areas – one is **anatomic texturing**, the other is using **physical models** to produce believable general deformations.

- Associated texture synthesis algorithms are already under development by Ilknur Kabul. However, there is a substantial amount of additional work to do in the area of creating a reference library of solid textures from anatomic photography and various found sources. Integrating information from truly multivariate data sources, such as diffusion tensor imaging, could also fall into this project of extending MGR's available appearance models.
- Using physical models for organ-by-organ deformation would be a valuable tool for designing intuitive scene compositions. As Fig. 148 shows, cutting and moving objects out of the way rather than simply ghosting them out can be a very effective means of showing the 3D positions of hidden internal anatomy without losing the global context. As previously mentioned, physical modeling is another area where having m-rep models fit to important structures in the scene can provide an uncomplicated framework for approaching this otherwise complex problem.

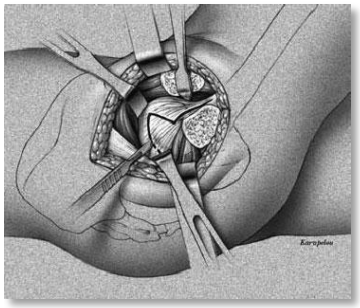

Fig. 148 A medical illustration that simulates a physical procedure with retractors provides an intuitive understanding of the 3D positions of the internal anatomic structures. Contours of the hidden bones are also sketched on the surface. (www.conservativehipsolutions.com)

These supporting technologies are somewhat tangential to MGR's core rendering capabilities; they are methods to compute additional data sources to drive a scene. In terms of algorithmic extensions to MGR's core capabilities, there are two projects that I personally think are intriguing: **Ray Traced MGR** and **Augmented Reality MGR**.

6.3.4 Ray Traced MGR

As machines get faster, MGR's goal of interactive rate rendering for complex scenes becomes more possible to achieve in a ray casting or image-order setting. Integrating MGR methods with a high quality offline ray tracing renderer such as POV-Ray (www.povray.org) would provide dramatically better lighting models than can be achieved with object-order DVR. Furthermore, many implementation details of mgrView that are necessarily complex due to the fast object-order framework would be quite straightforward in a ray casting framework.

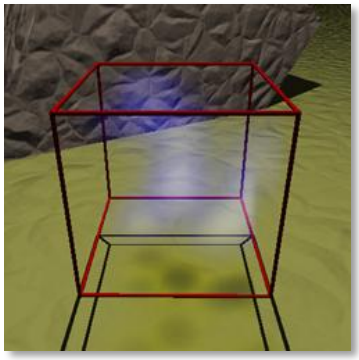

Fig. 149 (Bourke 2003) describes how to use POV-Ray to render volume data with a variant of the Gaussian "splat" method discussed in Chapter 2. Note the soft shadows of the semi-transparent volume cast on the ground.

Such an interface need not be particularly closely tied to the rendering engine. For example, Fig. 13 from the introduction demonstrating the shape change project was actually a prototype rendering done in VolView by preprocessing the image and using VolView's native transfer function controls. The importance mask was precomputed for a single point of view, and the marked intervening voxels in the target image were simply set to 0 intensity so that they could be suppressed. The halos were achieved by assigning otherwise unused value ranges to the voxels intersected by the segmentation surfaces so that they could be controlled independently of the rest of the transfer function.

An alternative to creating an MGR interface for a software ray tracer is to map MGR methods for scene appearance and composition onto a hardware-accelerated ray tracing solution. Software ray tracing is typically done serially, and each ray is independently processed. The three possible directions for hardware accelerated ray tracing are all methods for parallelizing a simplified ray following algorithm.

1. Map MGR ray tracing implementations onto **graphics hardware** using shaders or a language like nVidia's CUDA. Several sources including (Kruger and Westermann 2003) and (Quammen 2006) describe straightforward mappings of volume ray *casting* (no secondary rays) onto graphics hardware using a fragment shader to follow the entire ray that will contribute to each pixel value. (Purcell, et al. 2005) proposes a graphics hardware algorithm for triangulated surface ray *tracing*, that is, casting secondary rays for shadows, reflections, and antialiasing. Since MGR makes extensive use of both volumetric and surface data, it would require a hybrid approach.

2. Map MGR ray tracing implementations onto **parallelizable general purpose processors**, such as IBM's Cell architecture. No volume rendering software for this architecture currently exists, but I have proposed an algorithm where each processor is responsible for a small set of rays and image data is streamed between them in a ring fashion. Given a suitable preordering, it is likely that each voxel will only be loaded onto each processor one time. This could be extremely powerful as the render time is expected to be bound to the absolute minimum constraint, that is, the amount of time required to access each voxel from memory exactly once. The method would scale in both speed and target resolution with the number of dedicated processors.

   MGR methods that can reference additional data sources as if they were extra channels of the base image (*e.g.*, deformation fields or scene maps) could be easily integrated into this setting. However, most of MGR's methods rely on spatially incoherent texture lookups (*e.g.*, solid textures, photo mapping, or color mapping), and it is not clear how such data could be efficiently interleaved into the voxel-stream.

3. Map MGR ray tracing implementations onto a custom circuit design for a **field programmable gate array** (FPGA) or an application specific integrated circuit (ASIC). Custom ray tracing circuits for triangles have been proposed, most notably by (Woop 2005) as the "ray processing unit" or RPU. The challenges presented in extending this idea to realize a volume ray tracing circuit and moreover to support MGR's various needs are considerable and would likely be laborious.

   However, the result could be of considerable importance in the related clinical field of dose calculation. The volume ray casting algorithm *per se* need not accumulate a 2D buffer; it could easily accumulate a 3D buffer according to a distribution kernel – which is exactly the framework required for high precision dose calculation according to superposition/convolution. Using the same hardware engine to compute both visualizations and dose calculations for radiotherapy seems quite poetic. It seems that such dose calculations should be difficult to compute on a gpu because of the fast scattered reads and writes to the 3D accumulations buffer, but (Jacques, et al. 2008) has recently implemented exactly this.

6.3.5 Augmented Reality MGR

One of the major goals of MGR is to provide a global 3D understanding without losing too much of the fine detail available in 2D slice-by-slice renderings. Virtual or augmented reality (VR or AR, first described in practice by (Sutherland 1968)) provides a powerful tool for enhancing 3D comprehension by actually preserving the 3-dimensionality of the displayed objects. Adding a simple stereo display would be an intentionally straightforward extension of mgrView's windowing system, and a fully tracked AR display has also been considered. However, the question naturally arises of how useful stereo display, VR, or AR actually is in clinical tasks.

Clinical stereo display would be an easily achievable extension for mgrView. But while it seems obvious that providing even a limited 3D display such as a stereo view would ameliorate any 3D planning task, there are few promising examples of stereo providing significant utility in a clinical setting. The earliest reference to stereo views for volume rendering was (State, Balu and Fuchs 1994), which proposed a kind of limited single axis ("bunker view") stereo volume visualization that could be pre-rendered to give the illusion of interactive display. Subsequently (Zuiderveld, van Ooijen, et al. 1996) described using an SGI system and an object-order volume renderer to produce fully interactive stereo views, but comments about stereo capability were mostly in regard to the shutter glasses hurting the subjects' eyes and to subject motion sickness.[25] A more recent example of stereo enabled planning is (Maupu, et al. 2005), which presents a method for planning liver shunting ("TIPS") using a stereo display of anatomic surfaces; however, this paper is also conspicuously silent on whether the improved depth understanding adds anything to the physicians ability to work in 3D. Elsevier's recent *Netter's Interactive 3D Anatomy* (Fig. 150) uses stereo views of anatomy for a medical education application. However, it appears to be limited to surface-only rendering of static anatomy.

Fig. 150 Marketing image from Elsevier's *Netter's Interactive 3D Anatomy* making the likely spurious implication that multiple people could sit around a table and interact with a 3D hologram.

Fully head-tracked 3D display has seen very few proposed clinical applications, likely because of the extensive hardware requirements.

[25] An interesting observation from this paper is that the clinical staff at a major university hospital circa 1996 found the volume visualization interesting and useful but that they simply had not been aware that such technology existed.

UNC has produced a few notable exceptions. A very early paper by (Chung 1992) establishes an elaborate virtual reality experiment to test the effect of virtual reality for the same kinds of radiotherapy targeting tasks I discussed in section 5.2, *MGR Applications in Adaptive Radiotherapy* – and concludes that immersive VR is little more effective than using a joystick.[26] On the other hand, it is possible that Chung's conclusions would have been different had MGR methods been available. For instance, one might imagine that the path planning task might have been easier using something like MGR's importance rendering algorithm to focus on the target through the occluders.

Other notable exceptions are methods for the online augmented-reality-guided biopsy (State, et al. 1996) and AR guided radiofrequency ablation (Fuchs, State, et al. 2008). This work can be seen as related to the enhanced endoscopic view suggested in section 5.3, *Enhanced Endoscopy from Multiple Modalities*; however, State does not annotate views or mix modalities beyond showing the ultrasound data projected back to its original position in the patient. I am intrigued by the idea of using MGR methods in a framework like State's to provide a head tracked AR view of a patient plan projected onto the patient herself for evaluating patient setup.

Interacting with the Scene

The other important aspect of virtual reality is the opportunity to present the user with more sophisticated means of interaction than a simple mouse. It is quite difficult to interact with the many degrees of freedom available in a volume rendering. Some earlier literature has experimented with the idea of working with more intuitive controls, but the idea does not seem to have gained significant traction.

Fig. 151 The doll interface from (Hinckley, et al. 1997).

(Pierce, Stearns and Pausch 1999) discusses the use of "voodoo dolls"[27] in virtual environments and (Ebert, et al. 1996) describes a two-handed stereotactic control for 'minimally immersive' interaction with volumetric data such as CT images. (Preim, et al. 2001) and (Goble, et al. 1995) describe two-handed interfaces for rotating and specifying

[26] As an aside, the most interesting proposal in the paper is the 'orbital' head tracked display mode, which is shown to be more useful than 'walk around' interactions.

[27] This is a great term that can be used for any physical proxy for interacting with a virtual object, *e.g.* "Using Voodoo Dolls for Patient Setup".

resections when planning oncological surgery. One of my favorite ideas for a controller is from (Hinckley, et al. 1997), which describes using a doll's head prop and a piece of plexiglass to control a cut plane orientation and position relative to a head CT. The method is described as being incredibly intuitive and universally easy to adopt (Fig. 151).

It seems to be well worth thinking about supporting such a two-handed controller in mgrView. Fig. 152 top shows an augmented reality image from an experiment using an ARToolKit marker as an optical six degree of freedom ("6-DOF") mouse to control a display. ARToolKit uses computer vision techniques to identify a square marker in a camera image and estimate the corresponding relative pose of the camera. That camera pose is converted into an OpenGL-type 4 x 4 view matrix that can be used to project virtual objects into the view. The project shown was a prototype for a patient-surface-to-CT alignment task that could be integrated with mgrView. The bottom image shows the author's daughter controlling an mgrView display and specifying a clip plane using a two-handed version of the same mechanism.

Fig. 152 Top, the author using ARToolKit (HIT Lab 2007) to intuitively manipulate a 3D object. Bottom, the author's daughter at 4 months using a two-handed version of the same mechanism to manipulate and clip the mgrView scene previously shown in Fig. 17.

APPENDIX: IMPLEMENTING THE PLANNING UNDER ERROR VIEW USING MGRVIEW

This appendix describes how to combine and extend library implementations of Model Guided Rendering (MGR) methods to address the tasks described in section 5.2.3, *Interactive Planning Under Error* . It is intended in part as a starting place for a reader who is attempting to recreate the example views from this dissertation or is interested in evaluating or extending the mgrView library. It is also in part a demo of mgrView's flexibility and ease of use for rapid prototyping[28]. This section assumes more familiarity with C++, OpenGL, and related UNC research software than the rest of the text does. Non-engineering oriented readers are welcome to skip over this material.

mgrView is a four-thousand line C++/OpenGL2.0 library implementing MGR's core functionality. It provides an interactive rate volume rendering framework for MGR's core methods and has only modest hardware requirements. The library includes some simple file I/O and an extensible default windowing and UI based on GLUT/GLUI (Radamacher, Stewart and Baxter 2006), but it is also intended to be embedded in other software environments, such as UNC's in-house clinical radiotherapy planning system, Plan-UNC (PLUNC). mgrView is available through the Computational Oncology lab at UNC's Department of Radiotherapy. See http://titan.radonc.unc.edu/~derek/mgrView for details.

mgrView provides one possible object-order implementation of MGR's general methods. However, core MGR functions – creating and referencing scene maps, 2D and 3D color mapping, volumetric animation, and importance rendering – have been designed to require small, relatively independent calculations at each operation, so they could be easily transferred to alternate parallelized frameworks (see section 6.3.4, *Ray Traced MGR*, for additional thoughts on this).

Overview

The general goal of the view is to create an interactive 3D volume animation of the effects of various sources of spatial error on dose distribution in both segmented and non-segmented regions for radiotherapy treatment planning. The mgrView application described combines data from several sources that are routinely generated during planning: a base planning CT image, the dose distribution, and a deformable registration that suggests possible patient motion, and an alternate imaging modality view of the region near the target. The application allows the user to smoothly control the effects of rigid and deformable registration error applied to the base image while holding the dose fixed. The volume can be clipped or importance rendered to focus on the target region.

[28] These goals were identified after talking to a neuroradiologist at Duke University Hospital who was excited about a newly introduced visualization system that let her work on non-axis aligned cut planes. When I expressed surprise that this had only recently arrived at the clinic, she suggested that the radiologists were at the mercy of complex software and attendant specialized hardware. Hence, mgrView has been developed with the goal of enabling high quality rendering for complex scenes without requiring either intimate understanding of the algorithms or access to the powerful hardware typically required for interactive-rate ray casting volume rendering.

Fig. 153 shows mgrView's basic class organization and calls out the tasks required to implement this project. In short, a variety of data must be preprocessed for import (step 2), and then a GLSL shader with the desired functionality crafted by combining pre-built components (step 4).

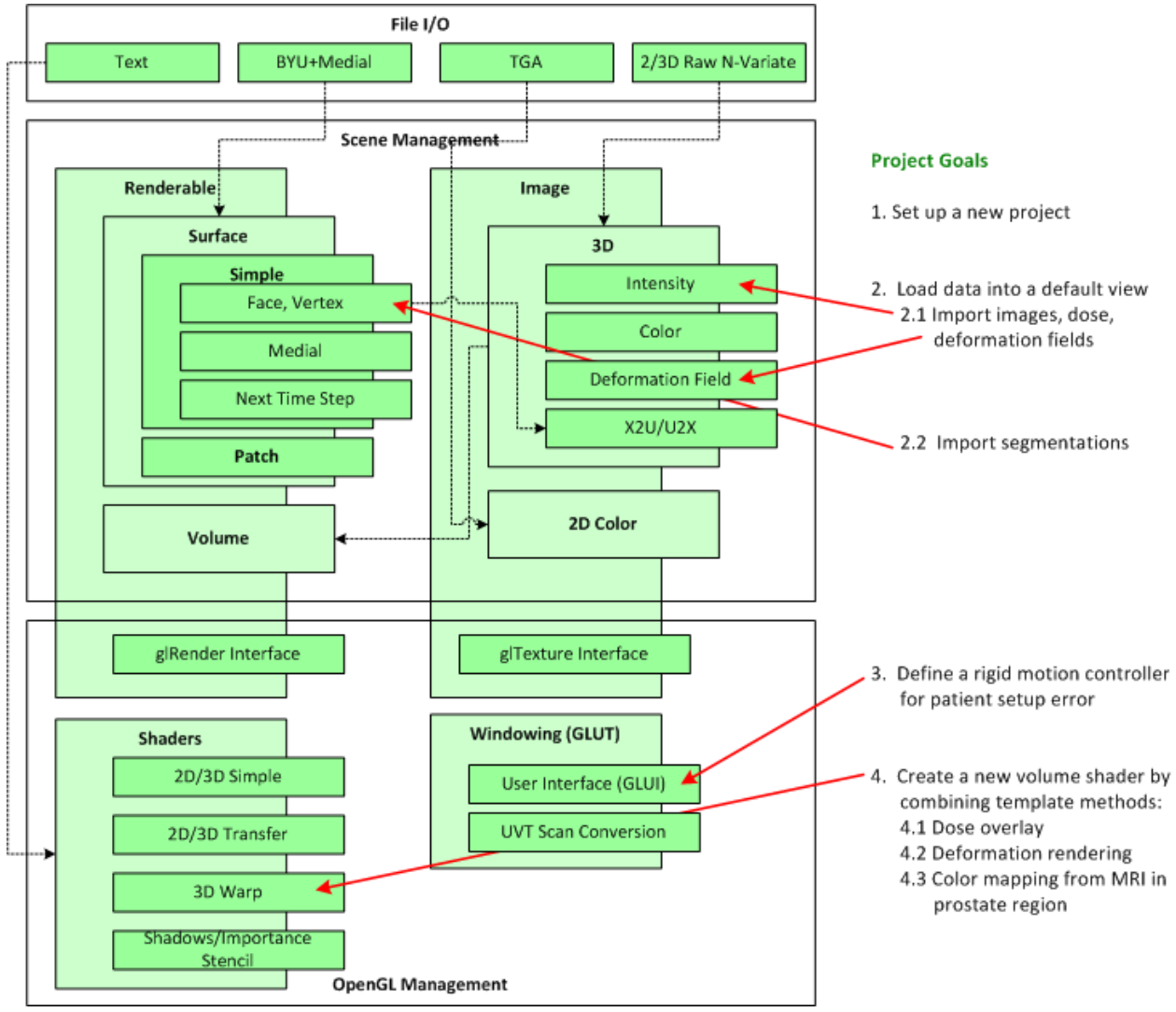

Fig. 153 mgrView's class organization with tasks for this project marked.

mgrView is organized in three basic sections: renderable objects such as tiles or volume data, images (sampled n-dimensional m-variate data fields), and OpenGL related functionality, such as shaders and windows. Each mgrView object can be attached to the other components to create a scene graph of images, shapes, and appearance descriptions descending from the world root object. mgrView's base functionality can be extended in any of these three component areas. For example, a patch surface type might extend the basic surface type as long as it implements the required glRender api. In this project the OpenGL-related functionality is extended by adding a shader that combines animation, dose overlay, and color mapping.

Step 1: Set Up a New Project

An mgrView project is usually a single file that initializes a few global path variables, loads the scene data, sets up scene relationships, and passes control to the mgrView rendering engine at the end of the main() function. Program 20 shows a basic mgrView program that loads the planning image only from this project and attaches it as the source channel for a volume renderable object. The directive to include "mgrv.h" gives the program access to the mgrView library functions. "data_dir" is a global variable that points to a directory with atlas as well as project textures. "project_dir" is the subdirectory where mgrView will expect to find any project-specific files, such as 6600.plan.pim in this example. All the sample data for this program can be found in mgrView's /data/6600.pelvis directory.

```
// New mgrView Project
#include "mgrv.h"
data_dir = "../../data"; project_dir = "6600.pelvis";
int main( argv, argc ) {
  // Load data
  mgrImage I_0   = mgrvLoadImage( "6600.plan.pim", vec3(1.0), 512, 512, 80 );
  mgrVolume v =mgrVolume( &I_0 );  // Setup volume based on I_0
  // Start renderer
  mgrvMainLoop();
}
```

Program 20 A single-image default scene.

The call to mgrvMainLoop() passes control to the rendering engine. At each frame the rendering engine starts at the scene root and passes over each object, updating OpenGL's state and switching shaders to reflect interrelationships between the images, shapes, and appearances. The object-order volume rendering core works as described in Chapter 2. A marching order is identified by comparing the dot product of the view direction with the volume axis directions, and then back-to-front ordered planes are sent down the graphics pipeline. mgrView's standard volume fragment program processes each image sample by simply looking up the CT intensity and windowing it. Because the volume is rendered in back-to-front order, every fragment is promoted to a pixel and composited into the frame buffer.

Step 2: Import Data

Step 2.1: Load Images, Dose, and Deformation Fields

mgrView's image class includes a simple binary file loader, mgrvLoadImage(), that can read any image format that stores grid data with channel (rgb) changing fastest, then x, then y, then z, *i.e.*, as ordered slices with each entry containing *n* variables. The image class does some simple processing such as pre-computing the image gradient or calculating a spatial transform into the unit cube. Any rigid spatial transforms for images are done in mgrView using the OpenGL texture matrix stack. The essential issue for converting gridded data into the unit cube is in understanding that the data extents – the voxel size times voxel count in each dimension, will be normalized to the range (0,1) in all three

dimensions. Because the image typically has a different height than its width or breadth, the z-direction coordinates must be stretched by the height-to-width ratio to maintain that relationship. Finally, mgrView loads the image into texture RAM and discards the original data unless directed to preserve it for further processing. To render an image directly, the image must be assigned as the "source0" channel of a volume-renderable object.

This project requires that images be bound to several additional predefined channels as well.

- A planning image, $\mathbf{I}_0$, is loaded into the "source0" channel.
- A dose grid file, **D**, that contains the expected dose at each sample for a particular plan is loaded into the "dose" channel.
- A dense registration field, **H**, is loaded into the "registration" channel and used to simulate internal shape change.
- An alternate modality source image for color mapping can be included in the "source1" channel. In this example, a corresponding MRI is used. In order to distinguish the alternate modality image from any treatment CT images, this additional source image will be called $\mathbf{I}_S$.

Refer back to Fig. 131 to see how each of these sources fit into the radiotherapy workflow. Program 21 shows mgrView code to load each data source.

```
// Load Image Data
mgrImage I_0   = mgrvLoadImage( "6600.plan.pim", vec3(0.1), 512,512,80, 1,
                                MGR_USHORT_DATA, MGR_SKIP_HEADER_FLAG );
mgrImage D  = mgrvLoadImage( "6600.dose.raw", vec3(0.1), 128,128,40, 1, MGR_FLOAT_DATA );
          D.SetOrigin( vec3( 0.3, 0.3, 0.3 );
mgrImage H  = mgrvLoadImage( "6600.plan2day1.raw", vec3(0.1), 128,128,40, 3,
                                MGR_FLOAT_DATA, MGR_SUBTRACT_IDENTITY_FLAG);
          H.SetOrigin( vec3( 0.3, 0.3, 0.3 );
mgrImage I_s   = mgrvLoadImage( "6600.mri.pim", vec3(0.1), 512, 512, 80, 1
                                MGR_USHORT_DATA, MGR_SKIP_HEADER_FLAG );
// Setup Scene
mgrVolume v =mgrVolume( &I_0 );  // Setup volume based on I_0
v.AttachImage( MGR_DOSE_IM, &D);  // Assign data channels
v.AttachImage( MGR_REG0_IM, &H);
v.AttachImage( MGR_SOURCE1_IM, &I_s );
```

Program 21 Using mgrView to load relevant images and segmentations for this scene.

In this project, the CT and MRI images are provided by PLUNC in .pim or "plan-image" format. Plan-images are stored as a header followed by an array of 16 bit unsigned integers. mgrView does not read image headers, so the image properties – in particular the voxel size, voxel count in each dimension, the number of variables, and the data format (*e.g.*, short, float) must be determined externally and passed as arguments to the loader. By default the loader assumes univariate unsigned short data. However, dose and displacement fields in particular are usually stored as floats.

The file reader also takes optional special action flags, such as skipping over header information before beginning to read[29]. Other common formats that use a header + gridded data arrangement, such as Pablo's raw3, Analyze's nii/hdr+raw and UNC's meta-image mha/mhd+raw formats (Chandra and Ibanez 2001) can be read similarly. Header information can be read from PLUNC images using the "plan-image-info" program, from Analyze images using Matlab's Analyze toolbox, and from raw3 by displaying the first few bytes of the file as text.

Many medical images are stored in DICOM format ("Digital Imaging and Communications in Medicine", see the website dicom.nema.org), which can be quite complex to parse. DICOM images can be read by our clinical modeling software, Pablo, and then saved in its own raw3 format which can be read as described above. Slice-wise volumes, such as the Visible Human color atlas slices can be converted to a single 3D color image using the Matlab "imgs2raw.m" script included in mgrView's /scripts directory. Converting a 3D image into a texture unit on the gpu requires that the samples be taken with a uniform slice thickness.

The displacement field for this project has been generated using UNC's in-house registration software, ImMap (described in (Foskey, et al. 2005)), a planning and treatment image are registered to each other with a rigid transform, **R**, which can be accounted for by adjusting the patient setup, and with a dense vector field, **H**, which captures residual shape change, such that $\mathbf{I}_1 = \mathbf{H}(\mathbf{R}(\mathbf{I}_0))$. The particular registration method is unimportant; mgrView's volume deformation animation would work with any sampled method for generating space-filling non-rigid mappings (*e.g.* for adaptive radiotherapy, (Lu W 2004),(Wang H 2004), (Mohan R 2005), (Freedman D 2005), or (Pekar, McNutt and Kaus 2004)).

In this example, the source and target images have been resampled into the same space, so **R** can be ignored. The displacement field itself is provided as a so-called "h-field", which contains target destinations in voxel- rather than world-coordinates. This use of the word "displacement" differs from the interpretation used by some registration packages. The special action flag "MGR_SUBTRACT_IDENTITY_FLAG" instructs mgrView's image class to convert h-field data to a vector field in unit unit cube coordinates before loading it as a texture unit. H-field data from ImMap is stored as mhd+raw files and can be read as 3-variate float data as described above – but it requires an additional origin position, since the displacement field is almost always computed only for a restricted region of interest. The origin must be read out of the header file and manually applied to the target image when it is loaded with the function mgrImage::SetOrigin().

As an aside, mgrView currently does not support data with more than four variables per entry, such as diffusion tensor images (DTI) that store the elements of a 3x3 symmetric matrix at every voxel. One possible method for addressing this in the future may be by splitting such data across multiple texture units.

[29] By counting backwards from the end the appropriate number of bytes and reading from there. This trick works with *many* different data formats.

Step 2.2: Importing Shapes into mgrView

Planning images are normally segmented as part of planning, either by hand or by deformable shape models. If a planning image is segmented by hand, medial models are fit to the resulting binary label images of each object. In either case, this results in a set of m-reps, $\mathbf{M}_{0i}$, one for each anatomic region. For this project, these shapes are read as shown in Program 22. If a shape is not explicitly identified as a child of another object, it is attached directly to the world root object in the scene. Shapes can also be assigned to other objects' data channels – in this example, the prostate shape is used as the target channel of the volume object.

```
// Load Shape Data
mgrShape* Mpros_0 = mgrvLoadShape( "6600.prostate.byu" );
mgrShape* Mrect_0  = mgrvLoadShape( "6600.rectum.byu" );
mgrShape* Mblad_0 = mgrvLoadShape( "6600.bladder.byu" );
v.AttachShape( MGR_LABEL_SURF, &Mpros_0 )
```

Program 22 Using mgrView to load relevant segmentations for this scene.

mgrView has no native m-rep type or m-rep reader but instead represents medial shapes as surfaces with extended properties. mgrView's surface class is a standard vertex and face list with some additional type information. Each vertex object must have at least a world position and a normal direction to be valid. If explicit surface normals are not provided, they will be computed automatically when the object is created by averaging the connected face normals. MGR model-coordinate functionality is provided in mgrView's surface class by assigning additional attributes to each the vertex object. A vertex may optionally include an object coordinate (*uv*, with t assumed to be 1 on the surface) and a medial position. If the vertices of a surface contain both of these attributes, mgrView will automatically generate the U2X and X2U maps for the surface and link them to the surface through their respective channels. Exporting these maps to a volume shader enables the shader to make model-coordinate relative decisions.

A vertex object can additionally include a pointer to another vertex that tracks values at the next time step for the surface. This allows the surface to be animated in multiple steps over time by interpolating between world positions. If the start and end vertices contain a medial position entry, the entire spoke can be interpolated over time, which would provide the basis for a new intermediate volumetric coordinate system. The current implementation is too slow to recompute intermediate U2X and X2U tables interactively, but this should be possible with optimization and slightly upgraded graphics hardware with more support for internal frame buffer objects.

M-rep models only explicitly identify a few surface vertices at the spoke tips. As discussed previously, higher resolution surfaces can be interpolated by various mechanisms with different smoothness and speed constraints. Because MGR methods do not require interactive surface generation, it is left to an off-line process, such as the surfacing algorithm in UNC's Pablo m-rep based segmentation tool[30]. The

[30] Pablo is discussed earlier in *M-Rep Software* on page 34. For academic access to Pablo see the UNC MIDAG website http://midag.cs.unc.edu; for commercial access see Morphormics http://www.morphormics.com.

pre-generated surface can be saved and loaded into mgrView as a simple tile geometry file. mgrView includes a byu file reader for simple tile geometry, this format choice largely motivated by convenience in exchanging data with other software in our lab. "movie.byu" is a file format for storing animations that originated at Brigham Young University. mgrView uses only the geometry part of the byu file format. byu geometry files follow a very simple format: a header with the number of vertices on the first line and the number of faces on the second, a list of three floats for every vertex, and a list of three integer vertex indices for every face. The final integer in each face triplet must be preceded by a negative sign. A formatting sample from a test data set, "cube.byu," is shown in Example 1. Complete documentation of the byu format exists in several sources online, https://people.sc.fsu.edu/~burkardt/data/byu/byu.html was used as a reference for mgrView's file reader. A useful program for converting other common surface file formats to byu is IVCON (http://orion.math.iastate.edu/burkardt/g_src/ivcon/ivcon.html).

```
  1   8   6  24   0
  1   6
 1.00000E+00 1.00000E+00 1.00000E+00
 1.00000E+00 1.00000E+00-1.00000E+00
 1.00000E+00-1.00000E+00 1.00000E+00
 1.00000E+00-1.00000E+00-1.00000E+00
-1.00000E+00 1.00000E+00 1.00000E+00
-1.00000E+00 1.00000E+00-1.00000E+00
-1.00000E+00-1.00000E+00 1.00000E+00
-1.00000E+00-1.00000E+00-1.00000E+00
  1   3   4  -2   5   7   8  -6   1   5   6  -2   3   7   8  -4
  1   5   7  -3   2   6   8  -4
```

Example 1 Sample byu format for a cube.

mgrView will instantiate a simple surface for any byu geometry file it is asked to read. Extended surface properties, as discussed above – explicit normals, object coordinates, and medial positions – can be included by adding additional specially named byu files to the data/surfs directory for the project. These files must be named as follows:

1. shape_name.byu
2. shape_name-normals.byu
3. shape_name-uvt.byu
4. shape_name-mpos.byu

These specially named files embed extended vertex data triplets in the vertex list portion of the file, and the face list is ignored. If only a single file is found, it is assumed to be the surface file. Only the surface vertices are strictly required to render a shape. The normals file is optional; it is provided to allow for higher order offline surface normal computation. If the normals file is absent, the vertex normal directions are computed as either the differences to the medial mesh (*i.e.*, the spoke directions) if the medial file exists, or they can be computed directly from a surface by averaging face normal directions as required.

Providing (*uvt*) coordinates of each vertex enables mgrView to render surfaces with object-centric texture directions as described previously in section 3.2, *Simple Texturing for Volumes*. Further providing the medial positions of each vertex enables mgrView to compute a volumetric coordinate system (the U2X and X2U lookup tables) for the shape. This allows mgrView to use the color mapping methods described in section 3.4, *3D Color Transfer*. A complete set of all four files can be generated in Pablo by saving a model as a byu surface file, contracting all of the atom radii to 0, and resaving a byu medial positions file. Supplemental normal and model coordinate files are generated automatically.

An obvious future extension to the code is to add loaders and surface generators for native m-rep registry files (.m3d). Alternately, a more compact version of this format, such as an extended .obj file supporting additional "position on the medial surface" parameters for each vertex would be useful.

Reparameterizing Models from Pablo

As an implementation aside, the m-rep parameterization in UNC's Pablo segmentation software is not a minimal representation. That is, it parameterizes a spherical topology (the version of m-reps used in these projects do not support objects with holes) with three parameters, (u'v'ϕ), where u' and v' count the medial sampling grid for atom-implied vertices but have fractional values for interpolated vertices, and ϕ indicates sidedness and varies around the "crest" regions. To reduce this to two parameters so that any coordinate on the medial sheet (along and across) can fit into the red and green channels alone, reserving blue for distance from the medial sheet (through), mgrView converts m-rep (u'v'ϕ) to a (latitude, longitude) representation, called, confusingly enough, (*uv*) again. Recall from section 3.1, *Creating a Scene Catalog*, that MGR's medial parameterization is a "shrink-wrap" cylindrical mapping, with v taken to run from one boundary edge to the other at v=0.5, then across the bottom of the object on the interval v=(0.5,1.0].

There are two different conversion algorithms in mgrView: one for "slab" type objects that have n x m medial samples with n,m>1, and another for "tube" type objects that have n x 1 medial samples with n>1. For slabs, ϕ = 1 on the anterior side, ϕ = -1 on the posterior side, and ϕ = 0 on the "crest" vertices. For tubes, v' is always 1 and ϕ counts around the object from -1 to 1. For tubes the conversion is quite simple, Pablo's v' is thrown out and the new v is simply equal to the read ϕ. For slabs the conversion is somewhat more complex and requires special cases for multiple different geometric regions (anterior or posterior faces, right A/P crest, left A/P crest, top crest, bottom crest). In both cases, integer u and v values are normalized to the (0,1) range. See the documentation of the mgrSurface::ReparameterizeSlab() function for additional implementation details.

Step 3: Add a Rigid Motion Controller

mgrView's image class automatically associates a 4 x 4 matrix transform with its image data. Whenever mgrView binds an image to a texture unit, this transform is applied to any texture coordinates intended for that texture unit. Such texture transforms serve many purposes, such as scaling and moving coordinate in the unit cube into coordinates for a region of interest or rotating a solid texture to align it with the "along" direction of an anatomic region. In this case, attaching a rigid motion user interface (UI) to the base image's transform will allow the clinician to simulate setup error by manipulating translation and rotation parameters interactively.

mgrView uses GLUI as a UI for rapid prototyping. mgrView's user interface class includes customized GLUI widgets for manipulating the parameters of several library types, including surface and volume renderables, clip planes, images, and transforms. The line in Program 23 requests that the window controller add a UI transform widget for the CT image. The resulting UI is shown in Fig. 154.

```
// Add Transform UI Control for Rigid Motion
mgrw->UI->AddTransformControl( &I_0.transform,
"Rigid error" );
```

Program 23 Adding a rigid motion controller to the CT image

The built-in deformation animation based on the displacement field is controlled by a global "time" parameter that is already included as a (0,1) slider in the default UI. This is sufficient for this view, but managing a weighted combination of deformation fields would require additional interface controls.

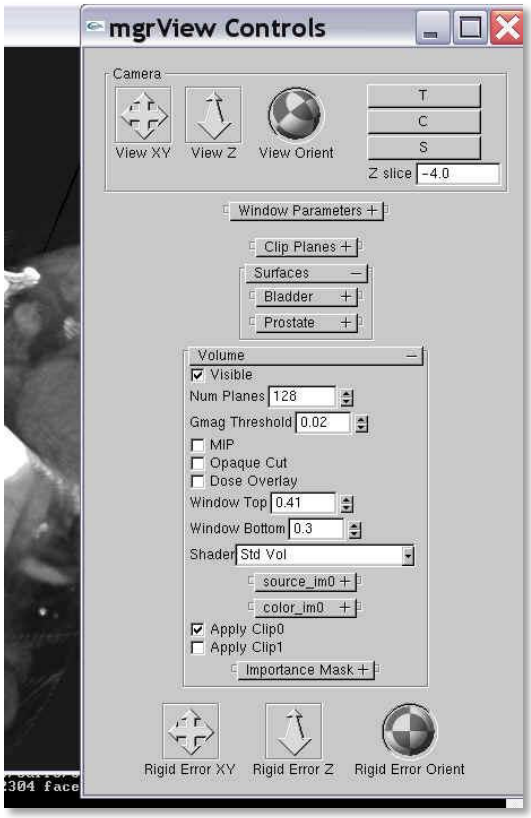

Fig. 154 mgrView's GLUI-based scene control interface with a rigid error controller.

Step 4: Create a New Shader

Volume animation and color mapping are provided by mgrView's built-in features, but combining the two and adding the dose overlay requires developing a custom shader program. There are four steps to adding the new shader to mgrView:

1. Add the new shader's name and dependencies to mgrView's shader controller
2. Modify the default volume rendering shader to add a dose overlay
3. Modify the shader to add deformation animation
4. Modify the shader to support color mapping from an alternate modality

Step 4.1: Add the new shader's name and dependencies to mgrView's shader controller

mgrView's shader object consists of a program id and a set of required texture channels. When a shader is invoked, it is passed a pointer to a particular renderable and the shader object's job is to check that all the required texture channels are present and to export the texture names and any supplemental parameters to the actual OpenGL hardware shader program. A global controller manages loading and compiling the shader programs and keeps a list of the available shaders identified by a unique name such as "MGR_STD_VOL_SHADER". The first step to adding a new shader to mgrView is to add its name to the MGRenum section of the mgrv.h file as shown in Program 24.

Second, the shader's name, parameters, and dependencies need to be listed in the mgrView::mgrLoadShaders() function found in mgrShaders.cpp, as shown in Program 25. In this case, the new shader uses the standard vertex program ("vstd"), it has a specialized fragment program in a file, the UI name should be "Dose + Err Vol", and it is defined for volume objects. The final argument is a bitwise-OR of flags for each of the required texture channels and GLSL uniforms and attributes that must be exported to the shader when it is invoked.

```
enum MGRenum {
  // Shader Names
  MGR_STD_VOL_SHADER          = 0,   ...
  // Add new shader name for ART Evaluation Project
   MGR_DOSE_ERR_VOL_SHADER
}
```

Program 24 Adding a new shader to mgrView's list of internal shader names in mgr.h

```
int mgrLoadShaders() {
      GLhandleARB vstd = mgrShader::LoadVertexProgram( "std.vert.glsl" );
      /* New shaders take the following parameters:
      mgrShader( enumerated mgrView shader name,
      vertex program handle or filename,
      fragment program handle or filename,
      string for display in UI,
      shader type (MGR_VOLUME_TYPE, MGR_SURFACE_TYPE),
      bitwise-or'd flags for reqd objects (MGR_SOURCE_IM0f|etc.))*/
      …
      // New shader for ART Evaluation Project
      new mgrShader(  MGR_DOSE_ERR_VOL_SHADER,
            vstd, "dose+err.vol.frag.glsl",
            "Dose + Err Vol", MGR_VOLUME_TYPE,
            MGR_SOURCE_IM0f | MGR_DEFORMATION_IM1f |
            MGR_DOSE_IMf, MGR_TIME_UNIFORMf );
}
```

Program 25 Adding the new shader's parameters to the mgrLoadShaders function in mgrShaders.cpp

Adding the shader name and parameters to the loader will cause mgrView's setup routine to automatically try to read the required files and compile the shader program This new shader is still missing its fragment program, so copy the "std.vol.frag.glsl" program from mgrView's /common

directory and rename it to “dose+err.vol.frag.glsl”. Now when mgrView is run, the new shader should load and its name “Dose + Err Vol” will be automatically be available on the shader drop down of the default volume object control. It is not very interesting yet though, since it merely duplicates the functionality of the standard volume shader object.

Step 4.2: Modify the default volume rendering shader to add a dose overlay

This section explains first how to extend the standard volume fragment program by referencing an additional channel for a dose overlay. When the new shader was set up in the last step, a dose channel was flagged as required. This means that the shader will require that the volume have a dose channel assigned (recall Program 21) and that the shader will export that channel’s OpenGL texture unit binding as a named uniform variable to the hardware fragment program. Variable names for all the predefined channels are defined in the common mgr.common.glsl header. In this case, the base image channel is called “source_im0” and the dose channel is called “dose_im”.

The standard volume fragment program is only three lines – it reads an intensity from source_im0, it calls the library intensity_window() function, and it sets the output fragment color accordingly. To add a dose overlay, the program needs to additionally read the dose value and modulate the red channel accordingly. This is shown in Program 26. C0 is a constant that can vary to intensify or suppress the strength of the modulation. Note that the rigid transformation controller added previously will only affect the texture coordinate used for source_im0 and not the texture coordinate used for dose_im, so the dose overlay will remain fixed.

```
const float C0 = 0.5;  // Dose modulation strength
void main() {
  // Standard volume fragment program
  float source_intensity = texture3d( source_im0, texCoord0.xyz ).r;
  float windowed_intensity = intensity_window(source_intensity);
  // Dose modulation
  float dose_intensity = texture3d( dose_im, texCoord1.xyz ).r;
  vec4 modulated_color = vec4(   source_intensity + dose_intensity*C0, source_intensity,
                                 source_intensity, source_intensity*C0 );
  modulated_color = max( modulate_color, vec4(1.) );
  // Set output
  gl_FragColor = modulated_color;
}
```

Program 26 Adding dose modulation to the standard volume fragment program.

Step 4.3: Modify the shader to add deformation animation

This example uses only a single displacement field to simulate possible internal shape change, but extending this project to support multiple registration fields, such as those between the planning image and several fraction images or using a basis of “principal warps” as described in section 6.3.2, *Non-Clinical Applications of MGR*, is not particularly difficult beyond adding several more data channels. However, each additional texture channel lookup does contribute to an overall slowdown of the system.

The object-order algorithm for an animating volume is detailed in section 4.1.2, *Rendering Images Under Local Deformation*. mgrView's local deformation shader requires both the source_im0 channel and a disp_im channel that can be used to compute the world-space offset for any world position. Program 27 shows how to integrate this functionality into the dose+error shader. The global time variable is used to weight how much of the displacement to apply.

One implementation caveat is that each image may potentially be from a different region of interest and thus may have a different mapping between world coordinates and texture coordinates. Therefore, the developer must know the order in which the mgrView shader is going to bind the required texture channels to determine the mapping between source, dose, and deformation and texCoord0, texCoord1, and texCoord2. This order can be determined by looking at the texture channel ordering in the function mgrShader::ExportUniforms().

```
const float C0 = 0.5;  // Dose modulation strength
void main() {
  // Interpolate a local registration vector
  vec3 offset = texture3d( disp_im, texCoord2.xyz ).xyz;
  float source_intensity = texture3d( source_im0, texCoord0.xyz + time*offset ).r;
  float windowed_intensity = intensity_window(source_intensity);
  // Dose modulation
  float dose_intensity = texture3d( dose_im, texCoord1.xyz ).r;
  vec4 modulated_color = vec4(   windowed_intensity + dose_intensity*C0, windowed_intensity,
                                 windowed_intensity, windowed_intensity*C0 );
  modulated_color = max( modulate_color, vec4(1.) );
  // Set output
  gl_FragColor = modulated_color;
}
```

Program 27 Adding local deformation to the dose+error shader.

Step 4.4: Modify the shader to support color mapping from an alternate modality

The last step in building the dose+error shader is to add color mapping from the MRI channel. This particular example is done as a 'world-space mapping' of the CT and MR images; that is, the images are pre-aligned by resampling them into the same space. This means that mgrView does not require model coordinates to transfer the images; it only needs to determine the region where the transfer should occur. When the fragment is determined to be within the prostate region, the MRI channel is queried at the same coordinates and its values replace the CT values. Tinting or increasing the relative opacity of the region as shown in the previous Fig. 60 can also call attention to the local detail. Program 28 shows the completed dose+error GLSL shader.

Finally, in the main program, the new shader is assigned as the initial appearance for the volume object by calling the volume object's member function SetShader() with the shader's name as an argument. Program 29 shows the completed mgrView project program. The final line before the call to mgrvMainLoop() assigns the newly created shader as the default appearance for the volume object. Other built-in shaders can be still selected by a dropdown in the UI.

```
const float C0 = 0.5;  // Dose modulation strength
void main() {
  // Interpolate a local registration vector
  vec3 offset = texture3d( disp_im, texCoord2.xyz ).xyz;
  // Determine fragment membership
  float label_value = texture3d( label_im0, texCoord0.xyz + time*offset ).r;
  if (label_value==0.) {
     // Not in target region, so use CT
     float source_intensity = texture3d( source_im0, texCoord0.xyz + time*offset ).r;
     float final_intensity = intensity_window(source_intensity); }
  else
     // In target region, so use MR
     float final_intensity = texture3d( source_im1, texCoord0.xyz + time*offset ).r;
  // Dose modulation
  float dose_intensity = texture3d( dose_im, texCoord1.xyz ).r;
  vec4 modulated_color = vec4(   final_intensity + dose_intensity*C0, final_intensity,
                                 final_intensity, final_intensity*C0 );
  modulated_color = max( modulate_color, vec4(1.) );
  // Set output
  gl_FragColor = modulated_color;}
```

Program 28 Adding local deformation to the dose+error shader.

```
#include “mgrv.h”
data_dir = "data"; project_dir = "6600.pelvis";
int main( argv, argc ) {
  // Load Image Data
  mgrImage I_0   = mgrvLoadImage( “6600.plan.pim”, vec3(0.1), 512,512,80, 1,
                                  MGR_USHORT_DATA, MGR_SKIP_HEADER_FLAG );
  mgrImage D  = mgrvLoadImage( “6600.dose.raw”, vec3(0.1), 128,128,40, 1,MGR_FLOAT_DATA );
            D.SetOrigin( vec3( 0.3, 0.3, 0.3 );
  mgrImage H  = mgrvLoadImage( “6600.plan2day1.raw”, vec3(0.1), 128,128,40, 3,
                               MGR_FLOAT_DATA, MGR_SUBTRACT_IDENTITY_FLAG);
            H.SetOrigin( vec3( 0.3, 0.3, 0.3 );
  mgrImage I_s   = mgrvLoadImage( “6600.mri.pim”, vec3(0.1), 512, 512, 80, 1
                                  MGR_USHORT_DATA, MGR_SKIP_HEADER_FLAG );
  // Setup Scene & Bind Data Channels
  mgrVolume v =mgrVolume( &I_0 );  // Setup volume based on I_0
  v.AttachImage( MGR_DOSE_IM, &D);
  v.AttachImage( MGR_REG0_IM, &H);
  v.AttachImage( MGR_SOURCE1_IM, &I_s );
  // Load Shape Data
  mgrShape* Mpros_0 = mgrvLoadShape( “6600.prostate.byu” );
  mgrShape* Mrect_0  = mgrvLoadShape( “6600.rectum.byu” );
  mgrShape* Mblad_0 = mgrvLoadShape( “6600.bladder.byu” );
  mgrShape* Mpros_s = mgrvLoadShape( “6600.mri.prostate.byu” );
  // Add Transform UI Control for Rigid Motion
  mgrw->UI->AddTransformControl( &I_0.transform, “Rigid error” );
  // Set initial shader
  v.SetShader( MGR_DOSE_ERR_VOL_SHADER);
  // Start program
  mgrvMainLoop();}
```

Program 29 The completed mgrView project program for the dose under error view.

The view can be focused by user defined clip planes or by selecting any shape in the scene as a target for mgrView's built-in importance rendering capability. mgrView implements two different functions for importance rendering: one for static shapes and a different one for dynamic shapes as described in the section *Dynamic Importance & ROIs* on page 105. In this case, because the target regions shift according to the deformation field, dynamic importance rendering can be enabled by attaching **H** to each target's reg0_im channel and calling the function mgrView::mgrvEnableDeformationImportanceRendering(). However, this requires that the graphics hardware accelerator supports vertex texture fetch ("vtf") so that the vertex program can offset each vertex position according to a texture lookup from the registration field.

REFERENCES

"Definition of Oblate Spheroidal Coordinates." In *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, by M. and Stegun, I. A. (Eds.) Abramowitz, 752. New York: Dover, 1972.

Ackerman, M.J. "The Visible Human Project." *Proceedings of the IEEE* 86, no. 3 (March 1998): 504 - 511.

Besl, P J, and N D McKay. "A Method for Registration of 3-D Shapes." *IEEE Transactions on Pattern Analysis and Machine Intelligence* (IEEE Computer Society) 14, no. 2 (1992): 239-256.

Blinn, J. F., and M. E. Newell. "Texture and reflection in computer generated images." *Commun. ACM* 19, no. 10 (October 1976): 542-547.

Blum, Harry. *A Transformation for Extracting New Descriptors of Shape.* Edited by Weiant Wathen-Dunn. Cambridge: MIT Press, 1967.

Bookstein, F L. "Principal Warps: Thin-Plate Splines and the Decomposition of Deformations." *IEEE Trans. Pattern Anal. Mach. Intell.* (IEEE Computer Society) 11, no. 6 (1989): 567-585.

Borland, David, John P. Clarke, Julia R. Fielding, and Russell M. Taylor II. "Volumetric Depth Peeling for Medical Image Display." *Proc. SPIE Int. Soc. Opt. Eng.* 2006.

Bouguet, J-Y. *Matlab Camera Calibration Toolbox.* 2008. http://www.vision.caltech.edu/bouguetj/calib_doc/.

Bourke, Paul. *Using POVRay as a volume renderer.* 2003. http://local.wasp.uwa.edu.au/~pbourke/miscellaneous/df3/.

Brechbuhler, C., G. Gerig, and O. Kubler. "Parameterization of closed surfaces for 3-D shape." *Computer Vision, Graphics, and Image Processing: Image Understanding* 65 (1995).

Broadhurst, Robert E, Joshua Stough, Stephen M Pizer, and Edward L Chaney. "A statistical appearance model based on intensity quantile histograms." *ISBI.* IEEE, 2006. 422-425.

Bruckner, S. "Exploded Views for Volume Data." *IEEE Transactions on Visualization and Computer Graphics* 12, no. 5 (2006): 1077-1084.

Bruckner, S., P. Kohlmann, A. Kanitsar, and M.E Groller. "Integrating volume visualization techniques into medical applications." *Biomedical Imaging: From Nano to Macro, 5th IEEE International Symposium on (ISBI).* 2008. 820-823.

Bruckner, S., S. Grimm, A. Kanitsar, and M. E. and Groller. "Illustrative context-preserving exploration of volume data." *IEEE Transactions on Visualization and Computer Graphics* 12, no. 6 (2006): 1559–1569.

Bullitt, E., and S.R. Aylward. "Volume rendering of segmented image objects." *Medical Imaging, IEEE Transactions on* 21, no. 8 (August 2002): 998-1002.

Cabral, Brain, and Leith Leedom. "Imaging Vector Fields Using Line Integral Convolution." *Proceedings of SIGGRAPH.* ACM Press, 1993. 263--270.

Canny, J. "A computational approach to edge detection." *IEEE Trans. Pattern Anal. Mach. Intell.* 8, no. 6 (1986).

Card, D, and JL Mitchell. "Non-Photorealistic Rendering with Pixel and Vertex Shaders." In *ShaderX: Vertex and Pixel Shaders Tips and Tricks*. 2002.

Catmull, E., and J. Clark. "Recursively generated B-spline surfaces on arbitrary topological meshes." *Computer-Aided Design* 10, no. 6 (1978): 350-355.

Chandra, Parag, and Luis Ibanez. "ImageIO: design of an extensible image input/output library." *Crossroads* (ACM) 7, no. 4 (2001): 10-16.

Chen, David. "Volume Rendering Guided by Multiscale Medial Models." PhD Thesis, Department fo Computer Science, UNC, 1998.

Chen, James Z., Stephen M. Pizer, Edward L. Chaney, and Sarang Joshi. "Medical Image Synthesis via Monte Carlo Simulation." *Medical Image Computing and Computer-Assisted Intervention — MICCAI 2002.* 2002. 347-354.

Cheung, V., B. J. Frey, and N. Jojic. "Video Epitomes." *IEEE Intern. Conf. Computer Vision and Pattern Recognition (CVPR).* 2005.

Christensen, G.E., R.D. Rabbitt, and M.I. Miller. "Deformable templates using large deformation kinematics." *IEEE Transactions on Image Processing* 5, no. 10 (1996).

Christensen, GE, SC Joshi, and MI Miller. "Volumetric transformation of brain anatomy." *IEEE Transactions in Medical Imaging*, 1997.

Chung, J. C. "A comparison of head-tracked and non-head-tracked steering modes in the targeting of radiotherapy treatment beams." *Proceedings of the 1992 Symposium on interactive 3D Graphics.* 1992. 193-196.

Cline, H. E., W. E. Lorensen, and S. Ludke. "Two algorithms for the three-dimensional reconstruction of tomograms." *Medical Physics*, 1988.

Cootes, T F, C J Twining, K O Babalola, and C J Taylor. "Diffeomorphic statistical shape models." *Image Vision Comput.* (Butterworth-Heinemann) 26, no. 3 (2008): 326-332.

Cootes, T.F., G.J. Edwards, and C.J. Taylor. "Active Appearance Models." *Computer Vision — ECCV.* 1998. 484-.

Cootes, Timothy F, A Hill, Christopher J Taylor, and J Haslam. "The Use of Active Shape Models for Locating Structures in Medical Images." *IPMI '93: Proceedings of the 13th International Conference on Information Processing in Medical Imaging.* London, UK: Springer-Verlag, 1993. 33-47.

Cormack, Allen. "Representation of a Function by its Line Integrals with some Radiological Applications." *Journal of Applied Physics*, 1964.

Correa, C.D., D. Silver, and M. Chen. "Feature Aligned Volume Manipulation for Illustration and Visualization." *Visualization and Computer Graphics, IEEE Transactions on*, 2006.

Crouch, Jessica R., Stephen M. Pizer, Edward L. Chaney, and Marco Zaider. "Medially Based Meshing with Finite Element Analysis of Prostate Deformation." *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2003.* 2003.

Crow, Franklin C. "Shadow Algorithms for Computer Graphics." *Computer Graphics (SIGGRAPH '77 Proceedings).* 1977. 242-248.

Cullip, T. J., and U. Neumann. "Accelerating volume reconstruction with 3d texture hardware." Department of Computer Science at the University of North Carolina, Chapel Hill, 1993.

Dachille, Frank, Kevin Kreeger, Baoquan Chen, Ingmar Bitter, and Arie Kaufman. "High-quality volume rendering using texture mapping hardware." *HWWS '98: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware.* Lisbon, Portugal: ACM, 1998.

Damadian, R. V. "Tumor Detection by Nuclear Magnetic Resonance." *Science*, no. 171 (March 1971): 1151-1153.

Davis, Brad, Sarang Joshi, Matthieu Jomier, and Guido Gerig. "Unbiased diffeomorphic atlas construction for computational anatomy." *NeuroImage* 23, no. Supplement 1 (2004): S151-S160.

Davis, Richard E., et al. "Three-Dimensional High-Resolution Volume Rendering (HRVR) of Computed Tomography Data: Applications to Otolaryngology-Head and Neck Surgery." *The Laryngoscope* 101, no. 6 (1991): 573.

Debevec, Paul E., George Borshukov, and Yizhou Yu. "Efficient View-Dependent Image-Based Rendering with Projective Texture-Mapping." *In 9th Eurographics Rendering Workshop.* Vienna, Austria, 1998.

Delingette H., Montagnat J. "Shape and Topology Constraints on Parametric Active Contours." *Computer Vision and Image Understanding* 83, no. 2 (August 2001): 140-171.

Deschamps, Thomas. *3-D Path Extraction for Virtual Endoscopy.* 2002. http://math.lbl.gov/~deschamp/html/virtual_endoscopy.html.

Diepstraten, J., D. Weiskopf, and T. and Ertl. "Interactive cutaway illustrations." *Computer Graphics Forum* 22, no. 3 (2003).

Dong, Feng, and Gordon J. Clapworthy. "Volumetric texture synthesis for non-photorealistic volume rendering of medical data." *The Visual Computer* 21, no. 7 (2005).

Doretto, G., and S. Soatto. "Dynamic shape and appearance models." *IEEE PAMI* 28, no. 12 (2006): 2006-2019.

Drebin, Robert A, Loren Carpenter, and Pat Hanrahan. "Volume rendering." *SIGGRAPH Comput. Graph.* (ACM) 22, no. 4 (1988): 65-74.

Ebert, D. S., C. D. Shaw, A. Zwa, and C. Starr. "Two-handed interactive stereoscopic visualization." *Proceedings of the 7th Conference on Visualization '96.* 1996.

Ebert, D., and P. Rheingans. "Volume illustration: non-photorealistic rendering of volume models." *Proceedings of Visualization 2000.* 2000. 195-202.

Ebert, David S., ed. *Texturing and Modeling: A Procedural Approach.* Morgan Kaufmann Pub, 1994.

Eisenberg, Murray, and Robert Guy. "A Proof of the Hairy Ball Theorem." *The American Mathematical Monthly* 86, no. 7 (1979): 571-574.

Fischer, J., D. Bartz, and W. Strasser. "Illustrative display of hidden iso-surface structures." *Proceedings of IEEE Visualization 2005.* 2005. 663-670.

Fletcher, P.T., Conglin Lu, S.M. Pizer, and Sarang Joshi. "Principal geodesic analysis for the study of nonlinear statistics of shape." *IEEE Transactions on Medical Imaging*, 2004.

Forsyth, David, and Jean Ponce. *Computer Vision: A Modern Approach.* 2002.

Foskey, M, BC Davis, J Rosenman, L Goyal, S Chang, and Joshi S. "Automatic segmentation of intra-treatment CT images for adaptive radiation therapy of the prostate." *Medical Image Computing and Computer-Assisted Intervention (MICCAI).* 2005. 442-50.

Freedman D, Radke R, Zhang T, Jeong Y, Lovelock M, and Chen G. "Model-based segmentation of medical imagery by matching distributions." *IEEE Transactions in Medical Imaging* 24, no. 3 (2005): 281-292.

Fuchs, H, Z M Kedem, and S P Uselton. "Optimal surface reconstruction from planar contours." *Commun. ACM* (ACM) 20, no. 10 (1977): 693-702.

Fuchs, Henry, and Stephen M. Pizer. Three Dimensional Display Using a Varifocal Mirror. United States Patent 4,607,255. August 19, 1986.

Fuchs, Henry, et al. "Optimizing a Head-Tracked Stereo Display System to Guide Hepatic Tumor Ablation." *Proc. Medicine Meets Virtual Reality (MMVR).* Newport Beach, CA, 2008.

Goble, John C., Ken Hinckley, Randy Pausch, John W. Snell, and Neal F. Kassell. "Two-Handed Spatial Interface Tools for Neurosurgical Planning." *Computer* 28, no. 7 (1995): 20-26.

Gooch, A., B. Gooch, P. Shirley, and E. Cohen. "A non-photorealistic lighting model for automatic technical illustration." *in Proceedings of the 25th Annual Conference on Computer Graphics and interactive Techniques SIGGRAPH.* New York, NY: ACM Press, 1998. 447-452.

Grimson WE, Kikinis R, Jolesz FA, Black PM. "Image-guided surgery." *Scientific American* 280, no. 6 (June 1999): 62-69.

Hagen, John. "Surgical Repair of the Septate Uterus (1990)." In *Atlas of Gynocological Surgery*, by Raymond Lee. Saunders, 1992.

Hagens, Gunther von. *Bodyworlds.* 2007. http://www.bodyworlds.com/.

Han, Qiong. *Proper Shape Representation of Single- and Multi-Figure Anatomical Objects.* PhD Thesis, University of North Carolina at Chapel Hill Dept. of Computer Science, 2007.

Heidmann, T. "Real Shadows, Real Time." *Iris Universe* 18 (1991): 23-31.

Hinckley, K, R Pausch, JH Downs, D Proffitt, and NF Kassell. "The props-based interface for neurosurgical visualization." *Studies in health technology and informatics* 39 (1997): 552-62.

HIT Lab. "ARToolKit Home Page." 2007.

Horn, Berthold K, and Brian G Schunck. *Determining Optical Flow.* Cambridge, MA, USA: Massachusetts Institute of Technology, 1980.

Interrante, V., H. Fuchs, and S.M. Pizer. "Conveying the 3D shape of smoothly curving transparent surfaces via texture." *IEEE Transactions on Visualization and Computer Graphics* 3, no. 2 (April/June 1997): 98-117.

Jacques, R., R. Taylor, J. Wong, and T. McNutt. "Towards Real-time Radiation Therapy: Superposition/Convolution at Interactive Rates." *International Journal of Radiation Oncology Biology Physics* 72, no. 1 (2008): S667-S667.

Joshi, S.C., and M.I. Miller. "Landmark matching via large deformation diffeomorphisms." *IEEE Transactions on Image Processing*, 2000.

Kabul, Ilknur, Derek Merck, Stephen Pizer, and Julian Rosenman. "Model Based Texture Synthesis for Anatomical Visualization." *in submission.* 2010.

Kass, M, A Witkin, and D Terzopoulos. "Snakes: Active Contour Models." *Int J Comp Vision*, 1987.

Kelemen, A., G. Szekely, and G. Gerig. "Elastic model-based segmentation of 3-D neuroradiological data sets." *IEEE Transactions on Medical Imaging* 14, no. 10 (Oct 1999): 828-839.

Kendall, David G. "Shape Manifolds, Procrustean Metrics, and Complex Projective Spaces." *Bull. London Math. Soc.* 16 (1984): 81-121.

Kindlmann, G., R. Whitaker, T. Tasdizen, and T. Moller. "Curvature-based transfer functions for direct volume rendering: methods and applications." *Proceedings of the 14th IEEE Visualization 2003 (Vis'03).* 2003.

Kindlmann, Gordon, and Carl-Fredrik Westin. "Diffusion Tensor Visualization with Glyph Packing." *IEEE Transactions on Visualization and Computer Graphics* 12, no. 5 (September-October 2006): 1329-1336.

Kniss, J., G. Kindlmann, and C. Hansen. "Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets." *Viz*, 2001: 255-262.

Koenderink, Jan J. *Solid Shape.* The MIT Press, 1990.

Konrad-Verse, Olaf, Bernhard Preim, and Arne Littmann. "Virtual Resection with a Deformable Cutting Plane." *In Proceedings of Simulation und Visualisierung.* 2004. 203-214.

Kopf, Johannes, Chi-Wing Fu, Daniel Cohen-Or, Oliver Deussen, Dani Lischinski, and Tien-Tsin Wong. "Solid Texture Synthesis from 2D Exemplars." *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007)* 26, no. 3 (2007): 2:1--2:9.

Kruger, J., and R. Westermann. "Acceleration Techniques for GPU-based Volume Rendering." *Proceedings of the 14th IEEE Visualization 2003.* 2003.

Langen, K.M, et al. "Evaluation of ultrasound-based prostate localization for image-guided radiotherapy." *Intl. J. of Rad-Onc, Biology, Physics* 57, no. 3 (November 2003): 635-644.

Lauterbur, P.C. "Image Formation by Induced Local Interactions: Examples of Employing Nuclear Magnetic Resonance." *Nature*, no. 242 (1973): 190-191.

Levoy, M, et al. "Volume rendering in radiation treatment planning." *Visualization in Biomedical Computing, Proceedings of the First Conference on.* 1990. 22-25.

Levoy, Marc. "Volume Rendering." *IEEE Computer Graphics and Applications* (IEEE Computer Society) 10, no. 2 (1990): 33-40.

Levy, Joshua H., Robert E. Broadhurst, Surajit Ray, Edward L. Chaney, and Stephen M. Pizer. "Signaling local non-credibility in an automatic segmentation pipeline." *Proceedings of the SPIE.* 2007.

Li, W., L. Ritter, M. Agrawala, B. Curless, and D Salesin. "Interactive cutaway illustrations of complex 3D models." *ACM SIGGRAPH 2007 Papers.* San Diego, California: ACM, 2007.

Lorensen, W. E., and H. E. Cline. "Marching cubes: A high resolution 3D surface construction algorithm." Edited by M. C. Stone. *Proceedings of the 14th Annual Conference on Computer Graphics and interactive Techniques , SIGGRAPH '87.* New York, NY: ACM Press, 1987. 163-169.

Lu W, Chen M-L, Olivera G, Ruchala K, and Mackie T. "Fast free-form deformable registration via calculus of variations." *Phys Med Biol* 49 (2004): 3067-3087.

Lu, A., and D.S. Ebert. "Example-based volume illustrations." *Visualization* (IEEE), 2005: 655- 662.

Lu, Aidong, et al. "Illustrative Interactive Stipple Rendering." *IEEE Transactions on Visualization and Computer Graphics* 9, no. 2 (2003): 127-138.

Maupu, D., M.H. Van Horn, S. Weeks, and E. Bullitt. "3D stereo interactive medical visualization." *Computer Graphics and Applications, IEEE*, 2005: 67-71.

Merck, D, et al. "Training Models of Anatomic Shape Variability." *Medical Physics* 35, no. 7 (August 2008).

Merck, Derek. *Model Guided Rendering for Medical Images.* Chapel Hill: University of North Carolina, Department of Computer Science, forthcoming dissertation.

Mohan R, Zhang X, Wang H, Kang Y, Wang X, Liu H, Ang K, Kuban D, Dong L. "Use of deformed intensity distributions for on-line modification of image-guided IMRT to account for interfractional anatomic changes." *Int J Rad Oncol Biol Phys* 61, no. 4 (2005): 1258-1266.

Mosleh-Shirazi, M. Amin, Vibeke N. Hansen, Peter J. Childs, Alan P. Warrington, and Frank H. Saran. "Commissioning and implementation of a stereotactic conformal radiotherapy technique using a general-purpose planning system." *Journal of Applied Clinical Medical Physics* 5, no. 3 (2004).

Mumford, David. "The bayesian rationale for energy functionals." 1994. Geometry Driven Diffusion in Computer Vision.

Nain D., Haker S., Kikinis R., Grimson W.E.L. "An Interactive Virtual Endoscopy Tool." *Satellite Workshop Int Conf Med Image Comput Comput Assist Interv.* 2001. http://www.spl.harvard.edu/publications/item/view/684.

Netter, Frank. *Atlas of Human Anatomy.* Saunders, 2006.

nVidia. *Fast Robust Shadow Volumes.* 2004. http://developer.nvidia.com/object/fast_shadow_volumes.html.

—. *Texture Space Bump Mapping.* May 2004. http://developer.nvidia.com/object/texture_space_bump_mapping.html.

Oliver, William R., Aziz Boxwala, Julian Rosenman, Tim Cullip, Jim Symon, and Glenn Wagner. "Three-Dimensional Visualization and Image Processing in the Evaluation of Patterned Injuries: The AFIP/UNC Experience in the Rodney King Case." *American Journal of Forensic Medicine & Pathology* 18, no. 1 (1997): 1-10.

Owada, S., F. Nielsen, M. Okabe, and T. Igarashi. "Volumetric illustration: designing 3D models with internal textures." Edited by J. Marks. *ACM SIGGRAPH 2004 Papers.* Los Angeles, 2004.

Pekar, V., T. McNutt, and M. Kaus. "Automated model-based organ delineation for radiotherapy planning in prostatic region." *Int J Rad Oncol Biol Phys* 60 (2004): 973-980.

Pelizzari, CA, et al. "Volumetric visualization of head and neck CT data for treatment planning." *Int J Radiat Oncol Biol Phys* 44, no. 3 (June 1999): 693-703.

Perlin, Ken. "An Image Synthesizer." *Computer Graphics* 19, no. 3 (1985): 287-296.

Phong, Bui Tuong. *Illumination for computer-generated images.* The University of Utah, 1973.

Piatt, Joseph H. (Jr.), Binil Starly, Wei Sun, and Eric Faerber. "Application of computer-assisted design in craniofacial reconstructive surgery using a commercial image guidance system." *Journal of Neurosurgery: Pediatrics* 104, no. 1 (January 2006).

Pierce, J. S., B. C. Stearns, and R. Pausch. "Voodoo dolls: seamless interaction at multiple scales in virtual environments." *Proceedings of the 1999 Symposium on interactive 3D Graphics.* 1999. 141-145.

Pizer, S. M. "The Medical Image Display and Analysis Group at the University of North Carolina: Reminiscences and philosophy." *Medical Imaging, IEEE Transactions on* 22, no. 1 (January 2003): 2-10.

Pizer, Stephen M. *Production and Processing of Radioisotope Scans.* PhD Thesis, Harvard University, 1967.

Pizer, Stephen M., et al. "Segmentation by Posterior Optimization of M-reps: Strategy and Results." *Medical Image Analysis*, 2008.

Pommert, Andreas, et al. "Creating a high-resolution spatial/symbolic model of the inner organs based on the Visible Human." *Medical Image Analysis* 5, no. 3 (September 2001): 221-228.

Prastawa, Marcel, Elizabeth Bullitt, and Guido Gerig. "Synthetic Ground Truth for Validation of Brain Tumor MRI Segmentation." *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2005.* 2005. 26-33.

Preim, B., W. Spindler, K. J. Oldhafer, and H Peitgen. "3D-interaction techniques for planning of oncologic soft tissue operations." *Procedures of Graphics Interfaces.* Canadian Information Processing Society, 2001. 183-190.

Purcell, Timothy J., Ian Buck, William R. Mark, and Pat Hanrahan. "Ray tracing on programmable graphics hardware." *SIGGRAPH '05: ACM SIGGRAPH 2005 Courses.* 2005.

Quammen, Cory. *Volumetric Depth Peeling.* 2006. http://wwwx.cs.unc.edu/~cquammen/wp/projects/volumetric-depth-peeling/.

Radamacher, Paul, Nigel Stewart, and Bill Baxter. "GLUI User Interface." 2006.

Radon, Johann. "On the Determination of Functions from Their Integral Values along Certain Manifolds (1917)." *IEEE Transactions on Medical Imaging 1986* 5, no. 4 (December 1917): 170-176.

Raskar, R., G. Welch, K. Low, and D. Bandyopadhyay. "Shader Lamps: Animating Real Objects With Image-Based Illumination." *Proceedings of the 12th Eurographics Workshop on Rendering Techniques .* London, U.K., 2001.

Rheingans, P., and D. Ebert. "Volume illustration: nonphotorealistic rendering of volume models." *Visualization and Computer Graphics, IEEE Transactions on* 7, no. 3 (2001): 253-264.

Robb, Richard. *Virtual Endoscopy Development and Evaluation Using the Visible Human Dataset.* 1996. http://www.nlm.nih.gov/research/visible/vhp_conf/robb/robb_pap.htm.

Röntgen, Wilhelm Conrad. "On a New Kind of Rays." *Nature*, 1896.

Rost, Randi. *OpenGL Shading Language (The Orange Book).* Addison-Wesley Professional, 2006.

Rueckert, Daniel, Paul Aljabar, Rolf A. Heckemann, Joseph V. Hajnal, and Alexander Hammers. "Diffeomorphic Registration Using B-Splines." *Medical Image Computing and Computer-Assisted Intervention (MICCAI).* 2006. 702--709.

Seligmann, D. D., and S. Feiner. "Specifying composite illustrations with communicative goals." *UIST '89: Proceedings of the 2nd annual ACM SIGGRAPH symposium on User interface software and technology.* Williamsburg, Virginia: ACM, 1989. 1-9.

Shreiner, Dave, Mason Woo, Jackie Neider, and Tom Davis. *OpenGL Programming Guide: The Official Guide to Learning OpenGL (The Red Book).* Addison-Wesley Professional, 2005.

Siddiqi, K., and S. Pizer. *Medial Representations: Mathematics, Algorithms and Applications.* Springer, 2008.

Simpson, Amber L., Burton Ma, Elvis C. S. Chen, Randy E. Ellis, and A. James Stewart. "Using Registration Uncertainty Visualization in a User Study of a Simple Surgical Task ." *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2006.* 2006.

Stabin, Michael G. *Doses from Medical Radiation Sources.* July 2008. http://www.hps.org/hpspublications/articles/dosesfrommedicalradiation.html.

State, A., et al. "Towards Performing Ultrasound-Guided Needle Biopsies from within a Head-Mounted Display." Edited by K. H. Höhne and R. Kikinis. *Visualization in Biomedical Computing, Proceedings of the 4th international Conference on.* 1996. 591-600.

State, Andrei, Suresh Balu, and Henry Fuchs. *Bunker View: Limited- range head-motion-parallax visualization for complex data sets.* 1994. http://www.cs.unc.edu/~andrei/pubs/1994_VBC_bunker.pdf.

Sutherland, I. E. "A head-mounted three dimensional display." *Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I.* ACM, 1968. 757-764.

Svakhine, Nikolai, David S. Ebert, and Don Stredney. "Illustration Motifs for Effective Medical Volume Illustration." *IEEE Computer Graphics and Applications* 25, no. 3 (May/June 2005): 31-39.

Talairach, J., and P. Tournoux. *Co-Planar Stereotaxic Atlas of the Human Brain: 3-Dimensional Proportional System : An Approach to Cerebral Imaging.* Thieme Medical Publishers, 1988.

Taylor, R. M. "Practical scientific visualization examples." *SIGGRAPH Comput. Graph.* 34, no. 1 (February 2000): 74-79.

Thall, Andrew. *Deformable Solid Modeling via Medial Sampling and Displacement Subdivision.* PhD Thesis, University of North Carolina at Chapel Hill, Department of Computer Science, 2004.

Thilmann, Christoph, et al. "Correction of patient positioning errors based on in-line cone beam CTs: clinical implementation and first experiences." *Radiation Oncology* 1, no. 1 (2006).

Thirion, J.-P. "Image matching as a diffusion process: an analogy with Maxwell's demons." *Medical Image Analysis* 2, no. 3 (September 1998): 243-260.

Tietjen, Christian, Tobias Isenberg, and Bernhard Preim. "Combining Silhouettes, Surface, and Volume Rendering for Surgery Education and Planning." *Procedings of IEEE Eurographics Symposium on Visualization.* 2005.

Trouvé, Alain, and Laurent Younes. "Metamorphoses Through Lie Group Action." *Foundations of Computational Mathematics* 5, no. 2 (2005): 173-198.

Turk, Greg. "Generating Textures on Arbitrary Surfaces Using Reaction-Diffusion." *Computer Graphics* 25, no. 4 (1991): 289-298.

UNC Hospital Department of Radiation Oncology. "Plan-UNC User's Guide." 2007.

Vesalius, Andreas. *The Illustrations from the Works of Andreas Vesalius of Brussels.* Edited by J. B. deC. M. Saunders and Charles D. O'Malley. Dover, 1973.

Viola, Ivan, Armin Kanitsar, and Eduard Groller. "Importance-Driven Volume Rendering." *Vis* (IEEE Computer Society) 00 (2004): 139-146.

Wang H, Dong l, Lii M, Lee A, De Crevoisier R, Mohan R, Cox J, Kuban D, and Cheung R. "Implementation and validation of a three-dimensional deformable registration algorithm for targeted prostate cancer radiotherapy." *Int J Rad Oncol Biol Phys* 61 (2004): 725-735.

Wang, S. W., and A. E. Kaufman. "Volume Sculpting." *Proceedings of the 1995 Symposium on interactive 3D Graphics.* Monterey, California: ACM, 1995.

Weigle, C, and R Taylor. "Visualizing intersecting surfaces with nested-surface techniques." *Vis*, 2005.

Weiskopf, Daniel, Klaus Engel, and Thomas Ertl. "Interactive Clipping Techniques for Texture-Based Volume Visualization and Volume Shading." *IEEE Transactions on Visualization and Computer Graphics* 9, no. 3 (2003): 298-312.

Westover, Lee. "Footprint evaluation for volume rendering." *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques.* ACM, 1990. 367-376.

Wong JR, Grimm L, Uematsu M, Oren R, Cheng CW, Merrick S, and Schiff P. "Image-guided radiotherapy for prostate cancer by CT-linear accelerator combination prostate movements and dosimetric considerations." *Int J Radiat Oncol Biol Phys* 61 (2005): 561–569.

Woop, S., Schmittler, J., Slusallek, P. "RPU: a programmable ray processing unit for realtime ray tracing." *ACM SIGGRAPH 2005 Papers.* 2005. 434-444.

Yan D, Lockman D, Brabbins D, Tyburski L, Martinez A. "An off-line strategy for constructing a patient-specific planning target volume in adaptive treatment process for prostate cancer." *Int J Radiat Oncol Biol* 48, no. 1 (2000): 289-302.

Yan, D, DA Jaffray, and J Wong. "A model to accumulate fractionated dose in a deforming organ." *Int J Radiat Oncol Biol Phys*, June 1999.

Yaniv, Ziv, and Kevin Cleary. "Image Guided Procedures, A Review." Tech Report, Georgetown University, 2006.

Zuiderveld, K.J., et al. "Clinical evaluation of interactive volume visualization." *Proceedings of the 7th Conference on Visualization '96.* San Francisco, California, 1996. 367-370.

Zuiderveld, K.J., M. Meissner, G. Harris, J.R. Lesser, A. Persson, and M. Vannier. "End Users' Perspectives on Volume Rendering in Medical Imaging: A job well done or not over yet?" *IEEE Visualization (VIS 05).* 2005. 711 - 713.